Imagen que contiene Calendario

El contenido generado por IA puede ser incorrecto.

**Estructura e interpretación   
de programas informáticos**

Segunda edición

Harold Abelson y Gerald Jay Sussman   
con Julie Sussman

Prólogo de Alan J. Perlis

The MIT Press   
Cambridge, Massachusetts Londres, Inglaterra

McGraw-Hill Book Company   
Nueva York San Luis San Francisco Montreal Toronto

Este libro forma parte de una serie de textos escritos por profesores del Departamento de Ingeniería Eléctrica y Ciencias Informáticas del Instituto Tecnológico de Massachusetts. Fue editado y producido por The MIT Press en virtud de un acuerdo de producción y distribución conjunto con McGraw-Hill Book Company.

**Información de pedidos:**

*Los pedidos de textos para Norteamérica*  
deben dirigirse a McGraw-Hill Book Company.   
Todos los demás pedidos deben dirigirse a The MIT Press.

*Fuera de América del Norte*  
Todos los pedidos deben dirigirse a The MIT Press o su distribuidor local.

© 1996 por el Instituto Tecnológico de Massachusetts

Segunda edición

[Licencia Creative Commons](https://translate.google.com/website?sl=en&tl=es&hl=es-419&prev=search&u=http://creativecommons.org/licenses/by-sa/4.0/)  
Estructura e interpretación de programas de computadora de [Harold Abelson y Gerald Jay Sussman con Julie Sussman](https://translate.google.com/website?sl=en&tl=es&hl=es-419&prev=search&u=http://mitpress.mit.edu/sicp/) tiene licencia [Creative Commons Atribución-Compartir Igual 4.0 Internacional](https://translate.google.com/website?sl=en&tl=es&hl=es-419&prev=search&u=http://creativecommons.org/licenses/by-sa/4.0/) de MIT Press.

Este libro fue compuesto por los autores utilizando el sistema de composición tipográfica L A T E X y fue impreso y encuadernado en los Estados Unidos de América.

**Datos de catalogación en publicación de la Biblioteca del Congreso**

Abelson, Harold   
    Estructura e interpretación de programas informáticos / Harold Abelson   
y Gerald Jay Sussman, con Julie Sussman. -- 2.ª ed.   
p. cm. -- (Serie de ingeniería eléctrica y ciencias de la computación)   
    Incluye referencias bibliográficas e índice.   
    ISBN 0-262-01153-0 (MIT Press tapa dura)   
    ISBN 0-262-51087-1 (MIT Press libro de bolsillo)   
    ISBN 0-07-000484-6 (McGraw-Hill tapa dura)   
    1. Computadoras digitales electrónicas -- Programación. 2. LISP (Lenguaje de programación informática) I. Sussman, Gerald Jay. II. Sussman, Julie.   
III. Título. IV. Serie: Serie de ingeniería eléctrica y ciencias de la computación del MIT.   
QA76.6.A255 1996   
005.13'3 -- dc20 96-17756 - Cuarta impresión, 1999

**PALABRAS DEL TRADUCTOR:**

Este documento es el resultado del mínimo esfuerzo, pero lleno de las mejores intenciones. Digo esto para que nadie piense ni en un solo momento que esto requirió esfuerzos titánicos como aprender inglés avanzado y traducir palabra por palabra el resultado de mi traducción del libro original. Aunque en un principio si quería hacer lo anterior (quería comprarme hasta el libro en físico y todo), opté por todo lo contrario al ver tantos proyectos sin completar de la misma tarea en github. Lo que me llevó a tomarme muy a pecho lo que dijo Voltaire “lo mejor es enemigo de lo bueno” y no quise arriesgarme a que este trabajo terminara de la misma manera.

Por eso, traduje el SICP Online de la página web del MIT con el Traductor de Google y comencé a copiar y pegar el contenido desde inicio a fin. Eso es todo, me tomó menos de una hora. Por supuesto no es la mejor de las traducciones, tiene un montón de errores en muchos sentidos y no es muy diferente a la traducción automática de Youtube, pero a la final es un producto terminado con rango de mejora.

Ahora, el propósito de este trabajo es brindarles a todos los hermanos que compartimos el español como lengua materna, la posibilidad de leer este libro de una manera más cómoda. Soy consciente que se requiere un gran esfuerzo el entender las ideas o ejercicios tan complejos que pueden presentarse en este libro (por lo menos para mí lo son), porque este libro no es precisamente un libro para los recién iniciados en la ciencia de la computación. Por eso se recomienda acompañar la lectura con las [clases grabadas del MIT](https://www.youtube.com/watch?v=-J_xL4IGhJA&list=PLE18841CABEA24090) y con el [libro original en formato web](https://mitp-content-server.mit.edu/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html).

Otra opción a lo anterior es aprovechar las increíbles capacidades de los nuevos modelos de Inteligencia Artificial (IA) (ya queda en sus manos decidir cual usar) para que te explique a detalle lo que no has terminado de entender. Es importante que se haga una lectura a consciencia del material y resolver los ejercicios propuestos en el libro, pero habrá momentos donde la cuestión no es de volver a leer hasta entender, sino que requiere el apoyo de conocimientos que son prerrequisitos o complementarias para asimilar la lección, y esto las puede facilitar la IA.

También es importante aclarar que la tendencia es el programar con lenguaje natural, como si estuvieras conjurando hechizos (Tal cuál como se expresa en este libro), esta es una más de las proezas de la IA, ya que pueden interpretar correctamente las instrucciones y ofrecer los resultados esperados. Quisiera pensar que muy pronto todos podremos conjurar hechizos con nuestras bocas y obtener soluciones informáticas como apps, páginas web, algoritmos, videojuegos, etc. al instante.

Dios los bendiga.

Este libro está dedicado, con respeto y admiración, al espíritu que vive en la computadora.

``Creo que es sumamente importante que los que trabajamos en informática mantengamos la diversión en la informática. Cuando empezó, era muy divertida. Por supuesto, los clientes que pagaban se veían perjudicados de vez en cuando, y después de un tiempo empezamos a tomarnos en serio sus quejas. Empezamos a sentirnos realmente responsables del uso perfecto, sin errores y con éxito de estas máquinas. No creo que lo seamos. Creo que somos responsables de hacerlas funcionar, de hacerlas avanzar en nuevas direcciones y de mantener la diversión en casa. Espero que el campo de la informática nunca pierda su sentido de la diversión. Sobre todo, espero que no nos convirtamos en misioneros. No os sintáis vendedores de Biblias. El mundo ya tiene demasiados de esos. Lo que sabéis sobre informática lo aprenderán otras personas. No creáis que la clave del éxito de la informática está sólo en vuestras manos. Lo que está en vuestras manos, creo y espero, es la inteligencia: la capacidad de ver la máquina como algo más que cuando la conocisteis por primera vez, de que podéis hacerla más.''

Alan J. Perlis (1 de abril de 1922-7 de febrero de 1990)

**Contenido**

**[Prefacio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-5.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_Temp_2)**

**[Prefacio a la segunda edición](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-6.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_Temp_3)**

**[Prefacio a la primera edición](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-7.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_Temp_4)**

**[Expresiones de gratitud](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-8.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_Temp_5)**

**[1 Construcción de abstracciones con procedimientos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_1)**  
        [1.1 Los elementos de la programación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1)   
            [1.1.1 Expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1.1)   
            [1.1.2 Nomenclatura y entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1.2)   
            [1.1.3 Evaluación de combinaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1.3)   
            [1.1.4 Procedimientos compuestos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1.4)   
            [1.1.5 El modelo de sustitución para la aplicación de procedimientos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1.5)   
            [1.1.6 Expresiones condicionales y predicados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1.6)   
            [1.1.7 Ejemplo: raíces cuadradas por el método de Newton](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1.7)   
            [1.1.8 Procedimientos como abstracciones de caja negra](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1.8)   
        [1.2 Procedimientos y los procesos que generan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.2)   
            [1.2.1 Recursión lineal e iteración](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.2.1)   
            [1.2.2 Recursión de árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.2.2)   
            [1.2.3 Órdenes de crecimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.2.3)   
            [1.2.4 Exponenciación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.2.4)   
            [1.2.5 Máximo común divisor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.2.5)   
            [1.2.6 Ejemplo: Prueba de primalidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.2.6)   
        [1.3 Formulación de abstracciones con procedimientos de orden superior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.3)   
            [1.3.1 Procedimientos como argumentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.3.1)   
            [1.3.2 Construcción de procedimientos utilizando Lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.3.2)  
            [1.3.3 Procedimientos como métodos generales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.3.3)   
            [1.3.4 Procedimientos como valores devueltos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.3.4)

**[2 Construcción de abstracciones con datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_2)**  
        [2.1 Introducción a la abstracción de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.1)   
            [2.1.1 Ejemplo: Operaciones aritméticas para números racionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.1.1)   
            [2.1.2 Barreras de abstracción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.1.2)   
            [2.1.3 ¿Qué se entiende por datos?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.1.3)   
            [2.1.4 Ejercicio extendido: Aritmética de intervalos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.1.4)   
        [2.2 Datos jerárquicos y la propiedad de clausura](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.2)   
            [2.2.1 Representación de secuencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.2.1)   
            [2.2.2 Estructuras jerárquicas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.2.2)   
            [2.2.3 Secuencias como interfaces convencionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.2.3)   
            [2.2.4 Ejemplo: un lenguaje de imágenes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.2.4)   
        [2.3 Datos simbólicos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.3)   
            [2.3.1 Cita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.3.1)   
            [2.3.2 Ejemplo: Diferenciación simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.3.2)   
            [2.3.3 Ejemplo: Representación de conjuntos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.3.3)   
            [2.3.4 Ejemplo: Árboles de codificación de Huffman](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.3.4)   
        [2.4 Representaciones múltiples para datos abstractos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.4)   
            [2.4.1 Representaciones para números complejos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.4.1)   
            [2.4.2 Datos etiquetados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.4.2)   
            [2.4.3 Programación dirigida por datos y aditividad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.4.3)   
        [2.5 Sistemas con operaciones genéricas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.5)   
            [2.5.1 Operaciones aritméticas genéricas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.5.1)   
            [2.5.2 Combinación de datos de diferentes tipos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.5.2)   
            [2.5.3 Ejemplo: Álgebra simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_2.5.3)

**[3 Modularidad, objetos y estado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-19.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_3)**  
        [3.1 Asignación y estado local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.1)   
            [3.1.1 Variables de estado local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.1.1)   
            [3.1.2 Los beneficios de introducir la asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.1.2)   
            [3.1.3 Los costes de introducir la asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.1.3)   
        [3.2 El modelo de entorno de evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.2)   
            [3.2.1 Las reglas de evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.2.1)   
            [3.2.2 Aplicación de procedimientos sencillos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.2.2)   
            [3.2.3 Marcos como depósito del estado local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.2.3)   
            [3.2.4 Definiciones internas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.2.4)   
        [3.3 Modelado con datos mutables](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.3)   
            [3.3.1 Estructura de lista mutable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.3.1)   
            [3.3.2 Representación de colas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.3.2)   
            [3.3.3 Representación de tablas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.3.3)   
            [3.3.4 Un simulador de circuitos digitales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.3.4)   
            [3.3.5 Propagación de restricciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.3.5)   
        [3.4 Concurrencia: el tiempo es esencial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.4)   
            [3.4.1 La naturaleza del tiempo en sistemas concurrentes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.4.1)   
            [3.4.2 Mecanismos para controlar la concurrencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.4.2)   
        [3.5 Flujos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.5)   
            [3.5.1 Los flujos son listas retardadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.5.1)   
            [3.5.2 Flujos infinitos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.5.2)   
            [3.5.3 Explotación del paradigma de flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.5.3)   
            [3.5.4 Flujos y evaluación retardada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.5.4)   
            [3.5.5 Modularidad de programas funcionales y modularidad de objetos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.5.5)

**[4 Abstracción metalingüística](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_4)**  
        [4.1 El evaluador metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1)   
            [4.1.1 El núcleo del evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.1)   
            [4.1.2 Representación de expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.2)   
            [4.1.3 Estructuras de datos del evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.3)   
            [4.1.4 Ejecución del evaluador como un programa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.4)   
            [4.1.5 Datos como programas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.5)   
            [4.1.6 Definiciones internas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.6)   
            [4.1.7 Separación del análisis sintáctico de la ejecución](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.7)   
        [4.2 Variaciones de un esquema: evaluación diferida](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.2)   
            [4.2.1 Orden normal y orden aplicativo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.2.1)   
            [4.2.2 Un intérprete con evaluación diferida](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.2.2)   
            [4.2.3 Flujos como listas diferidas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.2.3)   
        [4.3 Variaciones de un esquema: computación no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.3)   
            [4.3.1 Amb y búsqueda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.3.1)   
            [4.3.2 Ejemplos de programas no deterministas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.3.2)   
            [4.3.3 Implementación del evaluador Amb](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.3.3)  
        [4.4 Programación lógica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.4)   
            [4.4.1 Recuperación de información deductiva](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.4.1)   
            [4.4.2 Cómo funciona el sistema de consultas Trabajos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.4.2)   
            [4.4.3 ¿La programación lógica es lógica matemática?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.4.3)   
            [4.4.4 Implementación del sistema de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.4.4)

**[5 Computación con máquinas de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-30.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_5)**  
        [5.1 Diseño de máquinas de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.1)   
            [5.1.1 Un lenguaje para describir máquinas de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.1.1)   
            [5.1.2 Abstracción en el diseño de máquinas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.1.2)   
            [5.1.3 Subrutinas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.1.3)   
            [5.1.4 Uso de una pila para implementar la recursión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.1.4)   
            [5.1.5 Resumen de instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.1.5)   
        [5.2 Un simulador de máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.2)   
            [5.2.1 El modelo de máquina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.2.1)   
            [5.2.2 El ensamblador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.2.2)   
            [5.2.3 Generación de procedimientos de ejecución para instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.2.3)   
            [5.2.4 Monitoreo del desempeño de la máquina 5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.2.4)   
        [Asignación de almacenamiento y recolección de basura](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.3)   
            [5.3.1 Memoria como vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.3.1)   
            [5.3.2 Mantenimiento de la ilusión de memoria infinita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.3.2)   
        [5.4 El evaluador de control explícito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.4)   
            [5.4.1 El núcleo del evaluador de control explícito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.4.1)   
            [5.4.2 Evaluación de secuencia y recursión de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.4.2)   
            [5.4.3 Condicionales, asignaciones y definiciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.4.3)   
            [5.4.4 Ejecución del evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.4.4)   
        [5.5 Compilación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.5)   
            [5.5.1 Estructura del compilador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.5.1)   
            [5.5.2 Compilación de expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.5.2)   
            [5.5.3 Compilación de combinaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.5.3)   
            [5.5.4 Combinación de secuencias de instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.5.4)   
            [5.5.5 Un ejemplo de código compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.5.5)   
            [5.5.6 Direccionamiento léxico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.5.6)   
            [5.5.7 Interfaz del código compilado con el evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.5.7)

**[Referencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-36.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_Temp_849)**

**[Lista de ejercicios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-37.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_Temp_850)**

**[Índice](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-38.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_chap_Temp_851)**

[**Prefacio**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_Temp_2)

Los educadores, los generales, los dietistas, los psicólogos y los padres programan. Los ejércitos, los estudiantes y algunas sociedades están programados. Un ataque a grandes problemas emplea una sucesión de programas, la mayoría de los cuales surgen en el camino. Estos programas están plagados de problemas que parecen ser específicos del problema en cuestión. Para apreciar la programación como una actividad intelectual por derecho propio, hay que recurrir a la programación informática; hay que leer y escribir programas informáticos, muchos de ellos. No importa mucho de qué tratan los programas o para qué aplicaciones sirven. Lo que importa es lo bien que funcionan y lo bien que encajan con otros programas en la creación de programas aún mayores. El programador debe buscar tanto la perfección de la parte como la adecuación del conjunto. En este libro, el uso de "programa" se centra en la creación, ejecución y estudio de programas escritos en un dialecto de Lisp para su ejecución en un ordenador digital. Al utilizar Lisp, no restringimos o limitamos lo que podemos programar, sino sólo la notación para nuestras descripciones de programas.

Nuestro trato con el tema de este libro nos involucra con tres focos de fenómenos: la mente humana, colecciones de programas de computadora y la computadora. Cada programa de computadora es un modelo, tramado en la mente, de un proceso real o mental. Estos procesos, que surgen de la experiencia y el pensamiento humanos, son inmensos en número, intrincados en detalle y en algún momento sólo se entienden parcialmente. Rara vez nuestros programas de computadora los modelan para nuestra satisfacción permanente. Por lo tanto, aunque nuestros programas son colecciones discretas de símbolos cuidadosamente hechos a mano, mosaicos de funciones entrelazadas, evolucionan continuamente: los cambiamos a medida que nuestra percepción del modelo se profundiza, se amplía, se generaliza hasta que el modelo finalmente alcanza un lugar metaestable dentro de otro modelo con el que luchamos. La fuente de la euforia asociada con la programación de computadoras es el continuo despliegue dentro de la mente y en la computadora de mecanismos expresados ​​como programas y la explosión de percepción que generan. Si el arte interpreta nuestros sueños, ¡la computadora los ejecuta bajo la apariencia de programas!

A pesar de todo su poder, el ordenador es un duro capataz. Sus programas deben ser correctos, y lo que queremos decir debe decirse con precisión en cada detalle. Como en cualquier otra actividad simbólica, nos convencemos de la verdad del programa mediante argumentos. Al propio Lisp se le puede asignar una semántica (otro modelo, dicho sea de paso), y si se puede especificar la función de un programa, por ejemplo, en el cálculo de predicados, se pueden utilizar los métodos de demostración de la lógica para elaborar un argumento de corrección aceptable. Desafortunadamente, a medida que los programas se hacen grandes y complicados, como casi siempre ocurre, la adecuación, la coherencia y la corrección de las propias especificaciones se vuelven dudosas, de modo que los argumentos formales completos de corrección rara vez acompañan a los programas grandes. Dado que los programas grandes crecen a partir de los pequeños, es crucial que desarrollemos un arsenal de estructuras de programas estándar de cuya corrección estemos seguros (las llamamos modismos) y que aprendamos a combinarlas en estructuras más grandes utilizando técnicas de organización de valor demostrado. Estas técnicas se tratan en profundidad en este libro, y comprenderlas es esencial para participar en la empresa prometeica llamada programación. Más que cualquier otra cosa, el descubrimiento y el dominio de técnicas organizativas eficaces aceleran nuestra capacidad de crear programas grandes y significativos. Por el contrario, puesto que escribir programas grandes es muy exigente, nos sentimos estimulados a inventar nuevos métodos para reducir la masa de funciones y detalles que deben incluirse en los programas grandes.

A diferencia de los programas, los ordenadores deben obedecer las leyes de la física. Si quieren funcionar rápidamente (unos pocos nanosegundos por cambio de estado) deben transmitir electrones sólo a distancias pequeñas (como máximo 40 cm ). El calor generado por la enorme cantidad de dispositivos tan concentrados en el espacio tiene que ser eliminado. Se ha desarrollado un exquisito arte de ingeniería que equilibra la multiplicidad de funciones y la densidad de dispositivos. En cualquier caso, el hardware siempre opera a un nivel más primitivo que aquel en el que nos preocupamos por programar. Los procesos que transforman nuestros programas Lisp en programas "máquina" son en sí mismos modelos abstractos que programamos. Su estudio y creación nos dan una gran cantidad de información sobre los programas organizativos asociados con la programación de modelos arbitrarios. Por supuesto, el propio ordenador puede ser modelado de esa manera. Piénselo: el comportamiento del elemento de conmutación físico más pequeño está modelado por la mecánica cuántica descrita por ecuaciones diferenciales cuyo comportamiento detallado se captura por aproximaciones numéricas representadas en programas informáticos que se ejecutan en ordenadores compuestos de ... !

No se trata de una mera cuestión de conveniencia táctica identificar por separado los tres focos. Aunque, como suele decirse, todo está en la cabeza, esta separación lógica induce una aceleración del tráfico simbólico entre estos focos cuya riqueza, vitalidad y potencial sólo son superados en la experiencia humana por la evolución de la vida misma. En el mejor de los casos, las relaciones entre los focos son metaestables. Los ordenadores nunca son lo bastante grandes ni lo bastante rápidos. Cada avance en la tecnología del hardware conduce a empresas de programación más masivas, nuevos principios organizativos y un enriquecimiento de los modelos abstractos. Todo lector debería preguntarse periódicamente «¿Con qué fin, con qué fin?», pero no se lo haga con demasiada frecuencia, no sea que deje pasar la diversión de la programación por el estreñimiento de la filosofía agridulce.

Entre los programas que escribimos, algunos (pero nunca los suficientes) realizan una función matemática precisa, como ordenar o encontrar el máximo de una secuencia de números, determinar la primalidad o encontrar la raíz cuadrada. A estos programas los llamamos algoritmos, y se sabe mucho sobre su comportamiento óptimo, en particular con respecto a los dos parámetros importantes del tiempo de ejecución y los requisitos de almacenamiento de datos. Un programador debe adquirir buenos algoritmos y expresiones idiomáticas. Aunque algunos programas se resisten a las especificaciones precisas, es responsabilidad del programador estimar y siempre intentar mejorar su rendimiento.

Lisp es un superviviente, ya que se ha utilizado durante aproximadamente un cuarto de siglo. Entre los lenguajes de programación activos, sólo Fortran ha tenido una vida más larga. Ambos lenguajes han dado respuesta a las necesidades de programación de importantes áreas de aplicación: Fortran para la computación científica y de ingeniería, y Lisp para la inteligencia artificial. Estas dos áreas siguen siendo importantes y sus programadores están tan dedicados a ellos que Lisp y Fortran podrían seguir utilizándose activamente durante al menos otro cuarto de siglo.

Cambios en Lisp. El dialecto Scheme utilizado en este texto ha evolucionado a partir del Lisp original y difiere de este último en varios aspectos importantes, incluyendo el alcance estático para la vinculación de variables y la posibilidad de que las funciones produzcan funciones como valores. En su estructura semántica, Scheme es tan parecido a Algol 60 como a los primeros Lisp. Algol 60, que nunca volverá a ser un lenguaje activo, sigue vivo en los genes de Scheme y Pascal. Sería difícil encontrar dos lenguajes que sean la moneda de cambio de dos culturas más diferentes que las reunidas en torno a estos dos lenguajes. Pascal sirve para construir pirámides: estructuras imponentes, impresionantes y estáticas construidas por ejércitos que empujan bloques pesados ​​para colocarlos en su lugar. Lisp sirve para construir organismos: estructuras imponentes, impresionantes y dinámicas construidas por escuadrones que colocan en su lugar miríadas fluctuantes de organismos más simples. Los principios de organización utilizados son los mismos en ambos casos, excepto por una diferencia extraordinariamente importante: la funcionalidad exportable discrecional confiada al programador individual de Lisp es más de un orden de magnitud mayor que la que se encuentra en las empresas Pascal. Los programas Lisp inflan las bibliotecas con funciones cuya utilidad trasciende la aplicación que las produjo. La lista, la estructura de datos nativa de Lisp, es en gran medida responsable de ese crecimiento de la utilidad. La estructura simple y la aplicabilidad natural de las listas se reflejan en funciones que son sorprendentemente no idiosincrásicas. En Pascal, la plétora de estructuras de datos declarables induce una especialización dentro de las funciones que inhibe y penaliza la cooperación casual. Es mejor tener 100 funciones operando en una estructura de datos que tener 10 funciones operando en 10 estructuras de datos. Como resultado, la pirámide debe permanecer inalterada durante un milenio; el organismo debe evolucionar o perecer.

Para ilustrar esta diferencia, compare el tratamiento del material y los ejercicios en este libro con el de cualquier texto de primer curso que utilice Pascal. No se haga ilusiones de que este es un texto digerible sólo en el MIT, peculiar de la clase que se encuentra allí. Es precisamente lo que debe ser un libro serio sobre programación Lisp, sin importar quién sea el estudiante o dónde se utilice.

Cabe señalar que este es un texto sobre programación, a diferencia de la mayoría de los libros de Lisp, que se utilizan como preparación para trabajar en inteligencia artificial. Después de todo, las preocupaciones críticas de programación de la ingeniería de software y la inteligencia artificial tienden a fusionarse a medida que los sistemas bajo investigación se hacen más grandes. Esto explica por qué hay un interés creciente en Lisp fuera de la inteligencia artificial.

Como cabría esperar de sus objetivos, la investigación en inteligencia artificial genera muchos problemas de programación importantes. En otras culturas de programación, esta avalancha de problemas genera nuevos lenguajes. De hecho, en cualquier tarea de programación de gran envergadura, un principio organizativo útil es controlar y aislar el tráfico dentro de los módulos de tareas mediante la invención del lenguaje. Estos lenguajes tienden a volverse menos primitivos a medida que uno se acerca a los límites del sistema en el que los humanos interactuamos con mayor frecuencia. Como resultado, estos sistemas contienen funciones complejas de procesamiento del lenguaje que se replican muchas veces. Lisp tiene una sintaxis y una semántica tan simples que el análisis sintáctico puede tratarse como una tarea elemental. Por lo tanto, la tecnología de análisis sintáctico casi no desempeña ningún papel en los programas Lisp, y la construcción de procesadores de lenguaje rara vez es un impedimento para la tasa de crecimiento y cambio de los grandes sistemas Lisp. Finalmente, es esta misma simplicidad de la sintaxis y la semántica la responsable de la carga y la libertad que soportan todos los programadores Lisp. Ningún programa Lisp de cualquier tamaño que supere unas pocas líneas puede escribirse sin estar saturado de funciones discrecionales. ¡Inventar y ajustar; hacer ajustes y reinventar! Brindamos por el programador Lisp que escribe sus pensamientos entre paréntesis.

|  |
| --- |
| Alan J. Perlis,  New Haven, Connecticut |

[**Prefacio a la segunda edición**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_Temp_3)

|  |
| --- |
| ¿Es posible que el software no sea como ningún otro, que esté destinado a ser descartado: que el objetivo sea siempre verlo como una burbuja de jabón?  Alan J. Perlis |

El material de este libro ha sido la base de la asignatura de informática de nivel básico del MIT desde 1980. Habíamos estado enseñando este material durante cuatro años cuando se publicó la primera edición, y han transcurrido doce años más hasta la aparición de esta segunda edición. Nos complace que nuestro trabajo haya sido ampliamente adoptado e incorporado a otros textos. Hemos visto a nuestros estudiantes tomar las ideas y los programas de este libro y convertirlos en el núcleo de nuevos sistemas y lenguajes informáticos. En la realización literal de un antiguo juego de palabras talmúdico, nuestros estudiantes se han convertido en nuestros constructores. Tenemos suerte de tener estudiantes tan capaces y constructores tan consumados.

Al preparar esta edición, hemos incorporado cientos de aclaraciones sugeridas por nuestra propia experiencia docente y los comentarios de colegas del MIT y de otros lugares. Hemos rediseñado la mayoría de los principales sistemas de programación del libro, incluido el sistema aritmético genérico, los intérpretes, el simulador de máquina de registros y el compilador; y hemos reescrito todos los ejemplos de programas para asegurar que cualquier implementación de Scheme que cumpla con el estándar IEEE Scheme (IEEE 1990) pueda ejecutar el código.

Esta edición hace hincapié en varios temas nuevos. El más importante de ellos es el papel central que desempeñan los diferentes enfoques para tratar el tiempo en los modelos computacionales: objetos con estado, programación concurrente, programación funcional, evaluación diferida y programación no determinista. Hemos incluido nuevas secciones sobre concurrencia y no determinismo, y hemos tratado de integrar este tema a lo largo del libro.

La primera edición del libro siguió de cerca el programa de estudios de nuestra asignatura de un semestre del MIT. Con todo el material nuevo de la segunda edición, no será posible cubrir todo en un solo semestre, por lo que el instructor tendrá que elegir. En nuestra propia enseñanza, a veces nos saltamos la sección sobre programación lógica (sección  [4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4) ), hacemos que los estudiantes utilicen el simulador de máquina de registros pero no cubrimos su implementación (sección  [5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2) ) y solo damos una descripción general superficial del compilador (sección  [5.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5) ). Aun así, este sigue siendo un curso intenso. Algunos instructores pueden desear cubrir solo los primeros tres o cuatro capítulos, dejando el resto del material para cursos posteriores.

El sitio web mitpress.mit.edu/sicp ofrece asistencia a los usuarios de este libro. Incluye programas del libro, ejemplos de tareas de programación, materiales complementarios e implementaciones descargables del dialecto Scheme de Lisp.

[**Prefacio a la primera edición**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_Temp_4)

|  |
| --- |
| Un ordenador es como un violín. Podemos imaginarnos a un novato probando primero un fonógrafo y luego un violín. Este último, dice, suena terrible. Ése es el argumento que hemos escuchado de nuestros humanistas y de la mayoría de nuestros científicos informáticos. Los programas informáticos son buenos, dicen, para fines particulares, pero no son flexibles. Tampoco lo es un violín, o una máquina de escribir, hasta que aprendemos a utilizarlos.  Marvin Minsky, «Por qué la programación es un buen  medio para expresar ideas mal entendidas y formuladas» |

"La estructura y la interpretación de los programas informáticos" es la asignatura de nivel inicial en informática del Instituto Tecnológico de Massachusetts. Es obligatoria para todos los estudiantes del MIT que se especializan en ingeniería eléctrica o informática, y constituye una cuarta parte del "currículum básico común", que también incluye dos asignaturas sobre circuitos y sistemas lineales y una asignatura sobre diseño de sistemas digitales. Hemos estado involucrados en el desarrollo de esta asignatura desde 1978, y hemos enseñado este material en su forma actual desde el otoño de 1980 a entre 600 y 700 estudiantes cada año. La mayoría de estos estudiantes han tenido poca o ninguna formación formal previa en informática, aunque muchos han jugado un poco con computadoras y unos pocos tienen una amplia experiencia en programación o diseño de hardware.

El diseño de esta asignatura introductoria de informática refleja dos preocupaciones principales. En primer lugar, queremos establecer la idea de que un lenguaje informático no es sólo una forma de hacer que un ordenador realice operaciones, sino que es más bien un nuevo medio formal para expresar ideas sobre metodología. Por tanto, los programas deben escribirse para que los lea la gente y sólo incidentalmente para que los ejecuten las máquinas. En segundo lugar, creemos que el material esencial que debe abordarse en una asignatura de este nivel no es la sintaxis de construcciones particulares de lenguajes de programación, ni algoritmos ingeniosos para calcular funciones particulares de manera eficiente, ni siquiera el análisis matemático de algoritmos y los fundamentos de la informática, sino más bien las técnicas utilizadas para controlar la complejidad intelectual de grandes sistemas de software.

Nuestro objetivo es que los estudiantes que completen esta asignatura tengan una buena noción de los elementos de estilo y la estética de la programación. Deben dominar las principales técnicas para controlar la complejidad en un sistema grande. Deben ser capaces de leer un programa de 50 páginas, si está escrito en un estilo ejemplar. Deben saber qué no leer y qué no necesitan entender en ningún momento. Deben sentirse seguros a la hora de modificar un programa, conservando el espíritu y el estilo del autor original.

Estas habilidades no son exclusivas de la programación informática. Las técnicas que enseñamos y utilizamos son comunes a todo el diseño de ingeniería. Controlamos la complejidad mediante la creación de abstracciones que ocultan detalles cuando es apropiado. Controlamos la complejidad mediante el establecimiento de interfaces convencionales que nos permiten construir sistemas mediante la combinación de piezas estándar y bien entendidas de una manera "mix and match". Controlamos la complejidad mediante el establecimiento de nuevos lenguajes para describir un diseño, cada uno de los cuales enfatiza aspectos particulares del diseño y resta importancia a otros.

En nuestro enfoque de este tema se basa nuestra convicción de que la «ciencia informática» no es una ciencia y que su importancia tiene poco que ver con las computadoras. La revolución informática es una revolución en la forma en que pensamos y en la forma en que expresamos lo que pensamos. La esencia de este cambio es el surgimiento de lo que podría llamarse mejor *epistemología procedimental*: el estudio de la estructura del conocimiento desde un punto de vista imperativo, en oposición al punto de vista más declarativo adoptado por las disciplinas matemáticas clásicas. Las matemáticas proporcionan un marco para tratar con precisión las nociones de «qué es». La computación proporciona un marco para tratar con precisión las nociones de «cómo».

En la enseñanza de nuestro material utilizamos un dialecto del lenguaje de programación Lisp. Nunca enseñamos formalmente el lenguaje, porque no tenemos por qué hacerlo. Simplemente lo utilizamos y los estudiantes lo aprenden en unos pocos días. Esta es una gran ventaja de los lenguajes tipo Lisp: tienen muy pocas formas de formar expresiones compuestas y casi ninguna estructura sintáctica. Todas las propiedades formales se pueden cubrir en una hora, como las reglas del ajedrez. Después de un corto tiempo nos olvidamos de los detalles sintácticos del lenguaje (porque no hay ninguno) y nos centramos en los problemas reales: averiguar qué queremos calcular, cómo descompondremos los problemas en partes manejables y cómo trabajaremos en las partes. Otra ventaja de Lisp es que admite (pero no aplica) más estrategias a gran escala para la descomposición modular de programas que cualquier otro lenguaje que conozcamos. Podemos hacer abstracciones procedimentales y de datos, podemos usar funciones de orden superior para capturar patrones comunes de uso, podemos modelar el estado local usando asignación y mutación de datos, podemos vincular partes de un programa con flujos y evaluación retrasada, y podemos implementar fácilmente lenguajes embebidos. Todo esto está embebido en un entorno interactivo con excelente soporte para el diseño, construcción, prueba y depuración de programas incrementales. Agradecemos a todas las generaciones de magos de Lisp, comenzando por John McCarthy, quienes han creado una excelente herramienta de poder y elegancia sin precedentes.

Scheme, el dialecto de Lisp que utilizamos, es un intento de reunir la potencia y la elegancia de Lisp y Algol. De Lisp tomamos el poder metalingüístico que se deriva de la sintaxis simple, la representación uniforme de los programas como objetos de datos y los datos asignados a la pila y recolectados como basura. De Algol tomamos el alcance léxico y la estructura de bloques, que son obsequios de los pioneros del diseño de lenguajes de programación que formaban parte del comité de Algol. Deseamos citar a John Reynolds y Peter Landin por sus ideas sobre la relación del cálculo lambda de Church con la estructura de los lenguajes de programación. También reconocemos nuestra deuda con los matemáticos que exploraron este territorio décadas antes de que aparecieran las computadoras en escena. Entre estos pioneros se encuentran Alonzo Church, Barkley Rosser, Stephen Kleene y Haskell Curry.

[**Expresiones de gratitud**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_Temp_5)

Nos gustaría agradecer a las muchas personas que nos han ayudado a desarrollar este libro y este plan de estudios.

Nuestro tema es un claro descendiente intelectual de "6.231", un maravilloso tema sobre lingüística de programación y cálculo lambda enseñado en el MIT a fines de la década de 1960 por Jack Wozencraft y Arthur Evans, Jr.

Tenemos una gran deuda con Robert Fano, quien reorganizó el plan de estudios introductorio de ingeniería eléctrica y ciencias de la computación del MIT para enfatizar los principios del diseño de ingeniería. Él nos guió en el inicio de esta iniciativa y escribió el primer conjunto de notas temáticas a partir de las cuales surgió este libro.

Gran parte del estilo y la estética de la programación que intentamos enseñar se desarrollaron en colaboración con Guy Lewis Steele Jr., quien colaboró ​​con Gerald Jay Sussman en el desarrollo inicial del lenguaje Scheme. Además, David Turner, Peter Henderson, Dan Friedman, David Wise y Will Clinger nos han enseñado muchas de las técnicas de la comunidad de programación funcional que aparecen en este libro.

Joel Moses nos enseñó sobre la estructuración de sistemas grandes. Su experiencia con el sistema Macsyma para el cálculo simbólico nos permitió comprender que es necesario evitar las complejidades del control y concentrarse en organizar los datos para reflejar la estructura real del mundo que se está modelando.

Marvin Minsky y Seymour Papert han dado forma a muchas de nuestras actitudes sobre la programación y su lugar en nuestra vida intelectual. A ellos les debemos la comprensión de que la computación proporciona un medio de expresión para explorar ideas que de otro modo serían demasiado complejas para abordarlas con precisión. Destacan que la capacidad de un estudiante para escribir y modificar programas proporciona un medio poderoso en el que la exploración se convierte en una actividad natural.

También estamos totalmente de acuerdo con Alan Perlis en que programar es muy divertido y que es mejor que tengamos cuidado de apoyar la alegría de programar. Parte de esta alegría se deriva de observar a los grandes maestros en acción. Tenemos la suerte de haber sido aprendices de programación a los pies de Bill Gosper y Richard Greenblatt.

Resulta difícil identificar a todas las personas que han contribuido al desarrollo de nuestro plan de estudios. Agradecemos a todos los profesores, instructores de recitación y tutores que han trabajado con nosotros durante los últimos quince años y han dedicado muchas horas extra a nuestra materia, especialmente a Bill Siebert, Albert Meyer, Joe Stoy, Randy Davis, Louis Braida, Eric Grimson, Rod Brooks, Lynn Stein y Peter Szolovits. Nos gustaría reconocer especialmente las destacadas contribuciones docentes de Franklyn Turbak, ahora en Wellesley; su trabajo en la enseñanza de pregrado estableció un estándar al que todos podemos aspirar. Agradecemos a Jerry Saltzer y Jim Miller por ayudarnos a lidiar con los misterios de la concurrencia, y a Peter Szolovits y David McAllester por sus contribuciones a la exposición de la evaluación no determinista en el capítulo 4.

Muchas personas han hecho un esfuerzo significativo para presentar este material en otras universidades. Algunas de las personas con las que hemos trabajado en estrecha colaboración son Jacob Katzenelson en el Technion, Hardy Mayer en la Universidad de California en Irvine, Joe Stoy en Oxford, Elisha Sacks en Purdue y Jan Komorowski en la Universidad Noruega de Ciencia y Tecnología. Estamos excepcionalmente orgullosos de nuestros colegas que han recibido importantes premios de enseñanza por sus adaptaciones de este tema en otras universidades, entre ellos Kenneth Yip en Yale, Brian Harvey en la Universidad de California en Berkeley y Dan Huttenlocher en Cornell.

Al Moyé se encargó de que enseñáramos este material a los ingenieros de Hewlett-Packard y de que produjéramos cintas de vídeo de estas conferencias. Nos gustaría agradecer a los talentosos instructores, en particular a Jim Miller, Bill Siebert y Mike Eisenberg, que han diseñado cursos de formación continua que incorporan estas cintas y los han impartido en universidades e industrias de todo el mundo.

Muchos educadores de otros países han trabajado mucho en la traducción de la primera edición. Michel Briand, Pierre Chamard y André Pic produjeron una edición en francés; Susanne Daniels-Herold produjo una edición en alemán; y Fumio Motoyoshi produjo una edición en japonés. No sabemos quién produjo la edición en chino, pero consideramos un honor haber sido seleccionados como tema de una traducción "no autorizada".

Resulta difícil enumerar a todas las personas que han hecho contribuciones técnicas al desarrollo de los sistemas Scheme que utilizamos con fines didácticos. Además de Guy Steele, entre los principales expertos se encuentran Chris Hanson, Joe Bowbeer, Jim Miller, Guillermo Rozas y Stephen Adams. Otros que han dedicado un tiempo considerable son Richard Stallman, Alan Bawden, Kent Pitman, Jon Taft, Neil Mayle, John Lamping, Gwyn Osnos, Tracy Larrabee, George Carrette, Soma Chaudhuri, Bill Chiarchiaro, Steven Kirsch, Leigh Klotz, Wayne Noss, Todd Cass, Patrick O'Donnell, Kevin Theobald, Daniel Weise, Kenneth Sinclair, Anthony Courtemanche, Henry M. Wu, Andrew Berlin y Ruth Shyu.

Más allá de la implementación del MIT, nos gustaría agradecer a las muchas personas que trabajaron en el estándar IEEE Scheme, incluidos William Clinger y Jonathan Rees, quienes editaron el R 4 RS, y Chris Haynes, David Bartley, Chris Hanson y Jim Miller, quienes prepararon el estándar IEEE.

Dan Friedman ha sido durante mucho tiempo un líder de la comunidad Scheme. El trabajo más amplio de la comunidad va más allá de las cuestiones de diseño del lenguaje para abarcar innovaciones educativas significativas, como el plan de estudios de secundaria basado en EdScheme de Schemer's Inc. y los maravillosos libros de Mike Eisenberg y de Brian Harvey y Matthew Wright.

Agradecemos el trabajo de quienes contribuyeron a hacer de este un libro real, especialmente a Terry Ehling, Larry Cohen y Paul Bethge de MIT Press. Ella Mazel encontró la maravillosa imagen de la portada. Para la segunda edición estamos particularmente agradecidos a Bernard y Ella Mazel por su ayuda con el diseño del libro, y a David Jones, el extraordinario mago de T E X. También estamos en deuda con aquellos lectores que hicieron comentarios penetrantes sobre el nuevo borrador: Jacob Katzenelson, Hardy Mayer, Jim Miller y especialmente Brian Harvey, quien hizo con este libro lo que Julie hizo con su libro *Simply Scheme*.

Por último, nos gustaría agradecer el apoyo de las organizaciones que han fomentado este trabajo a lo largo de los años, incluido el apoyo de Hewlett-Packard, hecho posible gracias a Ira Goldstein y Joel Birnbaum, y el apoyo de DARPA, hecho posible gracias a Bob Kahn.

[**Capítulo 1**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_1)

[**Construyendo abstracciones con procedimientos**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_1)

|  |
| --- |
| Los actos de la mente, mediante los cuales ejerce su poder sobre las ideas simples, son principalmente estos tres: 1. Combinar varias ideas simples en una compuesta, y así se forman todas las ideas complejas. 2. El segundo es reunir dos ideas, ya sean simples o complejas, y colocarlas una junto a la otra para poder verlas de inmediato, sin unirlas en una sola, mediante lo cual obtiene todas sus ideas de relaciones. 3. El tercero es separarlas de todas las demás ideas que las acompañan en su existencia real: esto se llama abstracción, y así se forman todas sus ideas generales.  John Locke, *Ensayo sobre el entendimiento humano* (1690) |

Estamos a punto de estudiar la idea de una*Proceso computacional* . Los procesos computacionales son seres abstractos que habitan las computadoras. A medida que evolucionan, los procesos manipulan otros elementos abstractos llamados...*datos* . La evolución de un proceso está dirigida por un patrón de reglas llamado*Programa* . La gente crea programas para dirigir procesos. En efecto, conjuramos los espíritus de la computadora con nuestros hechizos.

Un proceso computacional es, de hecho, muy similar a la idea que un brujo tiene de un espíritu. No se puede ver ni tocar. No está compuesto de materia en absoluto. Sin embargo, es muy real. Puede realizar trabajo intelectual. Puede responder preguntas. Puede afectar al mundo al desembolsar dinero en un banco o al controlar un brazo robótico en una fábrica. Los programas que usamos para conjurar procesos son como los hechizos de un brujo. Están cuidadosamente compuestos a partir de expresiones simbólicas arcanas y esotéricas.*lenguajes de programación* que prescriben las tareas que queremos que nuestros procesos realicen.

Un proceso computacional, en una computadora que funciona correctamente, ejecuta programas con precisión y exactitud. Por lo tanto, como el aprendiz de brujo, los programadores novatos deben aprender a comprender y anticipar las consecuencias de sus conjuros. Incluso los pequeños errores (generalmente llamados...*bichos* o*Los fallos* ) en los programas pueden tener consecuencias complejas e imprevistas.

Afortunadamente, aprender a programar es considerablemente menos peligroso que aprender brujería, ya que los espíritus con los que tratamos están convenientemente contenidos de forma segura. Sin embargo, la programación en el mundo real requiere cuidado, experiencia y sabiduría. Un pequeño fallo en un programa de diseño asistido por computadora, por ejemplo, puede provocar el colapso catastrófico de un avión o una presa, o la autodestrucción de un robot industrial.

Los ingenieros de software expertos tienen la capacidad de organizar programas de forma que puedan estar razonablemente seguros de que los procesos resultantes realizarán las tareas previstas. Pueden visualizar el comportamiento de sus sistemas con antelación. Saben cómo estructurar programas para que los problemas imprevistos no tengan consecuencias catastróficas y, cuando surgen, pueden...*Depurar* sus programas. Los sistemas computacionales bien diseñados, como los automóviles o los reactores nucleares, se diseñan de forma modular, de modo que las partes se puedan construir, reemplazar y depurar por separado.

**[Programación en Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_6)**

Necesitamos un lenguaje apropiado para describir procesos, y para ello utilizaremos el lenguaje de programación Lisp. Así como nuestros pensamientos cotidianos se expresan habitualmente en nuestro idioma nativo (como inglés, francés o japonés), y las descripciones de fenómenos cuantitativos se expresan con notaciones matemáticas, nuestros pensamientos procedimentales se expresarán en Lisp.Lisp se inventó a finales de la década de 1950 como un formalismo para razonar sobre el uso de ciertos tipos de expresiones lógicas, llamadas*Ecuaciones de recursión* , como modelo para el cálculo. El lenguaje fue concebido porJohn McCarthy y se basa en su artículo "Funciones recursivas de expresiones simbólicas y su cálculo por máquina" (McCarthy 1960).

A pesar de su inicio como un formalismo matemático, Lisp es un lenguaje de programación práctico. Un Lisp*Un intérprete* es una máquina que ejecuta procesos descritos en el lenguaje Lisp. El primer intérprete de Lisp fue implementado porMcCarthy con la ayuda de colegas y estudiantes del Grupo de Inteligencia Artificial de laLaboratorio de Investigación de Electrónica del MIT y en el Centro de Computación del MIT. [1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_7) Lisp, acrónimo de LISt Processing, se diseñó para proporcionar capacidades de manipulación de símbolos para abordar problemas de programación como la diferenciación e integración simbólica de expresiones algebraicas. Para ello, incluyó nuevos objetos de datos conocidos como átomos y listas, lo que lo distinguió notablemente de todos los demás lenguajes de la época.

Lisp no fue el resultado de un esfuerzo de diseño coordinado. En cambio, evolucionó de forma informal y experimental en respuesta a las necesidades de los usuarios y a consideraciones pragmáticas de implementación. Esta evolución informal de Lisp ha continuado a lo largo de los años, y la comunidad de usuarios de Lisp se ha resistido tradicionalmente a los intentos de promulgar una definición "oficial" del lenguaje. Esta evolución, junto con la flexibilidad y la elegancia de la concepción inicial, ha permitido que Lisp, que es el segundo lenguaje más antiguo de uso generalizado en la actualidad (solo...Fortran es más antiguo), adaptándose continuamente para incorporar las ideas más modernas sobre diseño de programas. Por lo tanto, Lisp es ahora una familia de dialectos que, si bien comparten la mayoría de las características originales, pueden diferir significativamente entre sí. El dialecto de Lisp utilizado en este libro se llamaEsquema. [2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_8)

Por su carácter experimental y su énfasis en la manipulación de símbolos,Al principio, Lisp era muy ineficiente para los cálculos numéricos, al menos en comparación con Fortran. Sin embargo, con el paso de los años, se han desarrollado compiladores de Lisp que traducen programas a código máquina capaces de realizar cálculos numéricos con una eficiencia razonable. Y para aplicaciones especiales, Lisp se ha utilizado con gran eficacia. [3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_9) Aunque Lisp aún no ha superado su antigua reputación de ser extremadamente ineficiente, ahora se utiliza en muchas aplicaciones donde la eficiencia no es la prioridad. Por ejemplo, Lisp se ha convertido en un lenguaje predilecto para los lenguajes de shell de sistemas operativos y para los lenguajes de extensión de editores y sistemas de diseño asistido por computadora.

Si Lisp no es un lenguaje convencional, ¿por qué lo usamos como marco para nuestra discusión sobre programación? Porque el lenguaje poseeCaracterísticas únicas que lo convierten en un excelente medio para estudiar importantes construcciones de programación y estructuras de datos, y para relacionarlas con las características lingüísticas que las sustentan. La más significativa de estas características es el hecho de que las descripciones de procesos de Lisp, llamadas*Los procedimientos* pueden representarse y manipularse como datos Lisp. La importancia de esto radica en que existen potentes técnicas de diseño de programas que se basan en la capacidad de difuminar la distinción tradicional entre datos «pasivos» y procesos «activos». Como descubriremos, la flexibilidad de Lisp para manejar procedimientos como datos lo convierte en uno de los lenguajes más convenientes para explorar estas técnicas. La capacidad de representar procedimientos como datos también convierte a Lisp en un lenguaje excelente para escribir programas que deben manipular otros programas como datos, como los intérpretes y compiladores que soportan los lenguajes de programación. Más allá de estas consideraciones, programar en Lisp es muy divertido.

[1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_7) El*Manual del programador de Lisp 1*apareció en 1960, y el*Manual del programador de Lisp 1.5* (McCarthy 1965) se publicó en 1962. La historia temprana de Lisp se describe en McCarthy 1978.

[2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_8) Los dos dialectos en los que se escribieron la mayoría de los programas Lisp más importantes de la década de 1970 sonMacLisp(Luna 1978;Pitman 1983), desarrollado en elProyecto MAC del MIT yInterlisp(Teitelman 1974), desarrollado enBolt Beranek y Newman Inc. y laCentro de investigación de Xerox Palo Alto.Lisp estándar portátil(Hearn 1969;Griss 1981) fue un dialecto Lisp diseñado para ser fácilmente portable entre diferentes máquinas. MacLisp generó varios subdialectos, comoFranz Lisp, que fue desarrollado en laUniversidad de California en Berkeley, yZetalisp (Moon 1981), que se basó en un procesador de propósito especial diseñado en elEl Laboratorio de Inteligencia Artificial del MIT para ejecutar Lisp de forma muy eficiente. El dialecto Lisp utilizado en este libro, llamadoEsquema (Steele 1975), fue inventado en 1975 porGuy Lewis Steele Jr. y Gerald Jay Sussman del Laboratorio de Inteligencia Artificial del MIT, y posteriormente reimplementado para uso educativo en el MIT. Scheme se convirtió en un estándar IEEE en 1990 (IEEE 1990).El dialecto Common Lisp (Steele 1982, Steele 1990) fue desarrollado por la comunidad Lisp para combinar características de los dialectos Lisp anteriores y crear un estándar industrial para Lisp. Common Lisp se convirtió en estándar ANSI en 1994 (ANSI 1994).

[3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_9) Una de esas aplicaciones especiales fue un cálculo revolucionario de importancia científica: una integración del movimiento de laSistema Solar, que amplió los resultados previos en casi dos órdenes de magnitud y demostró que la dinámica del Sistema Solar es caótica. Este cálculo fue posible gracias a nuevos algoritmos de integración, un compilador específico y una computadora específica, todos implementados con la ayuda de herramientas de software escritas en Lisp.(Abelson y otros, 1992;Sussman y Wisdom 1992).

[**1.1 Los elementos de la programación**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_1.1)

Un lenguaje de programación potente es más que un simple medio para instruir a una computadora a realizar tareas. El lenguaje también sirve como marco dentro del cual organizamos nuestras ideas sobre los procesos. Por lo tanto, al describir un lenguaje, debemos prestar especial atención a los medios que este proporciona para combinar ideas simples y formar ideas más complejas. Todo lenguaje potente cuenta con tres mecanismos para lograr esto:

* **expresiones primitivas** , que representan las entidades más simples con las que se ocupa el lenguaje,
* **medios de combinación** , por los cuales se construyen elementos compuestos a partir de otros más simples, y
* **medios de abstracción** , mediante los cuales los elementos compuestos pueden nombrarse y manipularse como unidades.

En programación tratamos con dos tipos de elementos:procedimientos yDatos. (Más adelante descubriremos que en realidad no son tan distintos). Informalmente, los datos son «cosas» que queremos manipular, y los procedimientos son descripciones de las reglas para manipularlos. Por lo tanto, cualquier lenguaje de programación potente debería ser capaz de describir datos y procedimientos primitivos, y debería contar con métodos para combinar y abstraer procedimientos y datos.

En este capítulo trataremos únicamente los temas simples.datos numéricos para que podamos centrarnos en las reglas para construir procedimientos. [4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_10) En capítulos posteriores veremos que estas mismas reglas nos permiten construir procedimientos para manipular datos compuestos también.

**[1.1.1 Expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.1.1)**

Una forma sencilla de iniciarse en la programación es examinar algunas interacciones típicas con un intérprete del dialecto Scheme de Lisp. Imagine que está sentado frente a una terminal de computadora. Escribe una *expresión* y el intérprete responde mostrando el resultado de su *evaluación* .

Un tipo de expresión primitiva que podría escribir es un número. (Más precisamente, la expresión que escribe consiste en los numerales que representan el número en base 10). Si presenta a Lisp un número

486

El intérprete responderá imprimiendo [5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_11)

*486*

Las expresiones que representan números se pueden combinar con unaexpresión que representa unaProcedimiento primitivo (como + o \* ) para formar una expresión compuesta que representa la aplicación del procedimiento a esos números. Por ejemplo:

(+ 137 349)  
*486*  
(- 1000 334)  
*666*  
(\* 5 99)  
*495*  
(/ 10 5)  
*2*  
(+ 2.7 10)  
*12.7*

Expresiones como estas, formadas pordelimitar una lista de expresiones entre paréntesis para denotaraplicación de procedimiento, se denominan *combinaciones* . El elemento más a la izquierda de la lista se denomina*operador* , y los demás elementos se llaman*operandos* . ElEl valor de una combinación se obtiene aplicando el procedimiento especificado por el operador a la*argumentos* que son los valores de los operandos.

La convención de colocar el operador a la izquierda de los operandos se conoce como*La notación de prefijo* puede resultar algo confusa al principio, ya que se aleja significativamente de la convención matemática habitual. Sin embargo, la notación de prefijo tiene varias ventajas. Una de ellas es que permite...procedimientos que pueden tomar un número arbitrario de argumentos, como en los siguientes ejemplos:

(+ 21 35 12 7)  
*75*  
  
(\* 25 4 12)  
*1200*

No puede surgir ninguna ambigüedad, porque el operador es siempre el elemento más a la izquierda y toda la combinación está delimitada por los paréntesis.

Una segunda ventaja de la notación de prefijo es que se extiende de manera directa para permitir que las combinaciones se *aniden* , es decir, tener combinaciones cuyos elementos sean en sí mismos combinaciones:

(+ (\* 3 5) (- 10 6))  
*19*

En principio, no hay límite para la profundidad de dicha anidación ni para la complejidad general de las expresiones que el intérprete de Lisp puede evaluar. Somos nosotros, los humanos, quienes nos confundimos con expresiones relativamente simples como

(+ (\* 3 (+ (\* 2 4) (+ 3 5))) (+ (- 10 7) 6))

que el intérprete evaluaría fácilmente como 57. Podemos ayudarnos escribiendo dicha expresión en la forma

(+ (\* 3  
      (+ (\* 2 4)  
         (+ 3 5)))  
   (+ (- 10 7)  
      6))

siguiendo una convención de formato conocida como*Impresión bonita* , en la que cada combinación larga se escribe de forma que los operandos estén alineados verticalmente. Las sangrías resultantes muestran claramente la estructura de la expresión. [6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_12)

Incluso con expresiones complejas, el intérprete siempre opera en el mismo ciclo básico: lee una expresión de la terminal, la evalúa e imprime el resultado. Este modo de operación se expresa a menudo diciendo que el intérprete se ejecuta en un*Bucle de lectura-evaluación-impresión* . Observe en particular que no es necesario indicar explícitamente al intérprete que imprima el valor de la expresión. [7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_13)

**[1.1.2 Nombres y entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.1.2)**

Un aspecto crítico de un lenguaje de programación es el medio que proporciona para su uso.nombres para referirse a objetos computacionales. Decimos que elEl nombre identifica a un*variable* cuya*El valor* es el objeto.

En el dialecto Scheme de Lisp, nombramos las cosas condefinir . Escritura

(definir tamaño 2)

hace que el intérprete asocie el valor 2 con el nombre size . [8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_14) Una vez que el nombre size se ha asociado con el número 2, podemos referirnos al valor 2 por su nombre:

Talla  
*2*  
(\* talla 5)  
*10*

A continuación se muestran más ejemplos del uso de define :

(define pi 3.14159)  
(define radio 10)  
(\* pi (\* radio radio))  
*314.159*  
(define circunferencia (\* 2 pi radio))  
circunferencia  
*62.8318*

Define es el método de abstracción más simple de nuestro lenguaje, ya que nos permite usar nombres simples para referirnos a los resultados de operaciones compuestas, como la circunferencia calculada anteriormente. En general, los objetos computacionales pueden tener estructuras muy complejas, y sería extremadamente incómodo tener que recordar y repetir sus detalles cada vez que quisiéramos usarlos. De hecho, los programas complejos se construyen construyendo, paso a paso, objetos computacionales de complejidad creciente. El intérprete hace que esta construcción de programas paso a paso sea particularmente conveniente porque las asociaciones nombre-objeto se pueden crear incrementalmente en interacciones sucesivas. Esta característica fomenta ladesarrollo y prueba incremental de programas y es en gran medida responsable del hecho de queUn programa Lisp normalmente consta de una gran cantidad de procedimientos relativamente simples.

Debe quedar claro que la posibilidad de asociar valores con símbolos y recuperarlos posteriormente implica que el intérprete debe mantener algún tipo de memoria que registre los pares nombre-objeto. Esta memoria se denomina*medio ambiente* (más precisamente el*entorno global* , ya que veremos más adelante que un cálculo puede involucrar varios entornos diferentes). [9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_15)

**[1.1.3 Evaluación de combinaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.1.3)**

Uno de nuestros objetivos en este capítulo es aislar cuestiones sobre el pensamiento procedimental. Por ejemplo, consideremos que, al evaluar combinaciones, el intérprete sigue un procedimiento.

* Para evaluar una combinación, haga lo siguiente:

1. Evalúa las subexpresiones de la combinación.

2. Aplique el procedimiento que es el valor de la subexpresión más a la izquierda (el operador) a los argumentos que son los valores de las otras subexpresiones (los operandos).

Incluso esta simple regla ilustra algunos puntos importantes sobre los procesos en general. Primero, observe que el primer paso dicta que, para completar el proceso de evaluación de una combinación, primero debemos realizar el proceso de evaluación en cada elemento de la combinación. Por lo tanto, la regla de evaluación es*de naturaleza recursiva* , es decir, incluye, como uno de sus pasos, la necesidad de invocar la propia regla. [10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_16)

Observe la sucintamente con la que se puede usar la idea de recursión para expresar lo que, en el caso de una combinación profundamente anidada, de otro modo se consideraría un proceso bastante complejo. Por ejemplo, evaluar

(\* (+ 2 (\* 4 6))  
   (+ 3 5 7))

requiere que la regla de evaluación se aplique a cuatro combinaciones diferentes. Podemos obtener una imagen de este proceso medianterepresentando la combinación en forma de unaárbol, como se muestra en la figura  [1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es-419&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_1.1) . Cada combinación está representada por unnodo conramas correspondientes al operador y los operandos de la combinación derivada de él. ElLos nodos terminales (es decir, nodos sin ramas derivadas de ellos) representan operadores o números. Al considerar la evaluación en términos del árbol, podemos imaginar que los valores de los operandos se filtran hacia arriba, comenzando desde los nodos terminales y luego combinándose en niveles cada vez más altos. En general, veremos que la recursión es una técnica muy eficaz para tratar con objetos jerárquicos con forma de árbol. De hecho, la forma de «filtrar valores hacia arriba» de la regla de evaluación es un ejemplo de un tipo general de proceso conocido como*acumulación de árboles* .

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 1.1:**   Representación de árbol, que muestra el valor de cada subcombinación. |

A continuación, observe que la aplicación repetida del primer paso nos lleva al punto en el que necesitamos evaluar, no combinaciones, sino expresiones primitivas como numerales, operadores integrados u otros nombres. Nos ocupamos de los casos primitivos estipulando que

* Los valores de los numerales son los números que nombran,
* Los valores de los operadores integrados son las secuencias de instrucciones de la máquina que llevan a cabo las operaciones correspondientes, y
* Los valores de otros nombres son los objetos asociados con esos nombres en el entorno.

Podemos considerar la segunda regla como un caso especial de la tercera al estipular que los símbolos como + y \* también están incluidos en el entorno global y están asociados con las secuencias de instrucciones de máquina que son sus "valores". El punto clave a notar es el papel de laEl entorno determina el significado de los símbolos en las expresiones. En un lenguaje interactivo como Lisp, no tiene sentido hablar del valor de una expresión como (+ x 1) sin especificar ninguna información sobre el entorno que proporcione un significado para el símbolo  x (o incluso para el símbolo + ). Como veremos en el capítulo 3, la noción general del entorno como un contexto en el que se lleva a cabo la evaluación desempeñará un papel importante en nuestra comprensión de la ejecución del programa.

Tenga en cuenta que la regla de evaluación dada anteriormente no maneja definiciones. Por ejemplo, evaluar (define x 3) no aplica define a dos argumentos, uno de los cuales es el valor del símbolo x y el otro es 3, ya que el propósito de define es precisamente asociar x con un valor. (Es decir, (define x 3) no es una combinación).

Estas excepciones a la regla general de evaluación se denominan *formas especiales* . Define es el único ejemplo de una forma especial que hemos visto hasta ahora, pero pronto conoceremos otras.Cada forma especial tiene su propia regla de evaluación. Los distintos tipos de expresiones (cada una con su regla de evaluación asociada) constituyen laSintaxis del lenguaje de programación. En comparación con la mayoría de los demás lenguajes de programación, Lisp tiene una sintaxis muy simple; es decir, la regla de evaluación de las expresiones se puede describir mediante una regla general simple junto con reglas especializadas para un pequeño número de formas especiales. [11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_17)

**[1.1.4 Procedimientos compuestos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.1.4)**

Hemos identificado en Lisp algunos de los elementos que deben aparecer en cualquier lenguaje de programación potente:

* Los números y las operaciones aritméticas son datos y procedimientos primitivos.
* La anidación de combinaciones proporciona un medio para combinar operaciones.
* Las definiciones que asocian nombres con valores proporcionan un medio limitado de abstracción.

Ahora aprenderemos sobre*definiciones de procedimientos* , una técnica de abstracción mucho más poderosa mediante la cual se puede dar un nombre a una operación compuesta y luego hacer referencia a ella como una unidad.

Comenzaremos examinando cómo expresar la idea de "elevar al cuadrado". Podríamos decir: "Para elevar al cuadrado algo, multiplícalo por sí mismo". Esto se expresa en nuestro idioma como

(define (cuadrado x) (\* x x))

Podemos entender esto de la siguiente manera:

(define (cuadrado x) (\* x x))   
                                         
 Para elevar algo al cuadrado, multiplícalo por sí mismo.

Tenemos aquí un *procedimiento compuesto* , al que se le ha dado el nombre de cuadrado . El procedimiento representa la operación de multiplicar algo por sí mismo. A lo que se va a multiplicar se le da un nombre local, x , que desempeña el mismo papel que un pronombre en el lenguaje natural.Al evaluar la definición se crea este procedimiento compuesto y se lo asocia con el nombre cuadrado . [12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_18)

La forma general de una definición de procedimiento es

(define (< *nombre* > < *parámetros formales* >) < *cuerpo* >)

El <nombre> *es* un símbolo que se asociará con la definición del procedimiento en el entorno . [13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_19)Los < *parámetros formales* > son los nombres utilizados dentro del cuerpo del procedimiento para hacer referencia a los argumentos correspondientes del procedimiento.< *body* > es una expresión que producirá el valor de la aplicación del procedimiento cuando los parámetros formales se reemplacen por los argumentos reales a los que se aplica el procedimiento. [14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_20) El < *name* > y los < *formal parameters* > se agrupan dentro deparéntesis, tal como estarían en una llamada real al procedimiento que se está definiendo.

Habiendo definido cuadrado , ahora podemos usarlo:

(cuadrado 21)   
*441*  
  
(cuadrado (+ 2 5))   
*49*  
  
(cuadrado (cuadrado 3))   
*81*

También podemos utilizar el cuadrado como bloque de construcción para definir otros procedimientos. Por ejemplo, *x* 2 + *y* 2 se puede expresar como

(+ (cuadrado x) (cuadrado y))

Podemos definir fácilmente un procedimiento de suma de cuadrados que, dados dos números como argumentos, produce la suma de sus cuadrados:

(define (suma de cuadrados x y)   
  (+ (cuadrado x) (cuadrado y)))   
  
(suma de cuadrados 3 4)   
*25*

Ahora podemos utilizar la suma de cuadrados como un bloque de construcción para construir procedimientos adicionales:

(define (f a)   
  (suma de cuadrados (+ a 1) (\* a 2)))   
  
(f 5)   
*136*

Los procedimientos compuestos se utilizan exactamente de la misma manera que los procedimientos primitivos. De hecho, no se puede saber, al observar la definición de suma de cuadrados dada anteriormente, si el cuadrado está integrado en el intérprete, como + y \* , o si se define como un procedimiento compuesto.

**[1.1.5 El modelo de sustitución para la aplicación del procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.1.5)**

Para evaluar una combinación cuyo operador nombra un procedimiento compuesto, el intérprete sigue un proceso muy similar al que se sigue para las combinaciones cuyos operadores nombran procedimientos primitivos, que describimos en la sección  [1.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.3) . Es decir, el intérprete evalúa los elementos de la combinación y aplica el procedimiento (que es el valor del operador de la combinación) a los argumentos (que son los valores de los operandos de la combinación).

Podemos suponer que el mecanismo para aplicar procedimientos primitivos a los argumentos está integrado en el intérprete. Para los procedimientos compuestos, el proceso de aplicación es el siguiente:

* Para aplicar un procedimiento compuesto a los argumentos, evalúe el cuerpo del procedimiento con cada parámetro formal reemplazado por el argumento correspondiente.

Para ilustrar este proceso, evaluemos la combinación

(f5)

donde f es el procedimiento definido en la sección  [1.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.4) . Comenzamos recuperando el cuerpo de f :

(suma de cuadrados (+ a 1) (\* a 2))

Luego reemplazamos el parámetro formal a por el argumento 5:

(suma de cuadrados (+ 5 1) (\* 5 2))

Así, el problema se reduce a la evaluación de una combinación con dos operandos y un operador suma de cuadrados . La evaluación de esta combinación implica tres subproblemas. Debemos evaluar el operador para obtener el procedimiento que se va a aplicar, y debemos evaluar los operandos para obtener los argumentos. Ahora (+ 5 1) produce 6 y (\* 5 2) produce 10, por lo que debemos aplicar el procedimiento de suma de cuadrados a 6 y 10. Estos valores se sustituyen por los parámetros formales x e y en el cuerpo de la suma de cuadrados , reduciendo la expresión a

(+ (cuadrado 6) (cuadrado 10))

Si utilizamos la definición de cuadrado , esto se reduce a

(+ (\* 6 6) (\* 10 10))

que se reduce por multiplicación a

(+ 36 100)

y finalmente a

136

El proceso que acabamos de describir se denomina *modelo de sustitución* de la aplicación de procedimientos. Puede tomarse como un modelo que determina el «sentido» de la aplicación de procedimientos, en lo que respecta a los procedimientos de este capítulo. Sin embargo, hay dos puntos que conviene destacar:

* El propósito de la sustitución es ayudarnos a pensar en la aplicación del procedimiento, no proporcionar una descripción de cómo funciona realmente el intérprete. Los intérpretes típicos no evalúan las aplicaciones del procedimiento manipulando el texto de un procedimiento para sustituir valores por los parámetros formales. En la práctica, la "sustitución" se logra utilizando un entorno local para los parámetros formales. Analizaremos esto con más detalle en los capítulos 3 y 4, cuando examinemos la implementación de un intérprete en detalle.
* A lo largo de este libro, presentaremos una secuencia de modelos cada vez más elaborados de cómo funcionan los intérpretes, que culminarán con una implementación completa de un intérprete y un compilador en el capítulo 5. El modelo de sustitución es solo el primero de estos modelos: una forma de comenzar a pensar formalmente sobre el proceso de evaluación. En general, cuandoEn el modelado de fenómenos en ciencia e ingeniería, empezamos con modelos simplificados e incompletos. A medida que examinamos las cosas con mayor detalle, estos modelos simples se vuelven inadecuados y deben ser reemplazados por modelos más refinados. El modelo de sustitución no es una excepción. En particular, cuando abordamos en el capítulo 3 el uso de procedimientos con "datos mutables", veremos que el modelo de sustitución se descompone y debe ser reemplazado por un modelo más complicado de aplicación de procedimientos. [15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_21)

**[Orden aplicativo versus orden normal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_22)**

Según la descripción de la evaluación dada en la sección  [1.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.3) , el intérprete primero evalúa el operador y los operandos y luego aplica el procedimiento resultante a los argumentos resultantes. Esta no es la única forma de realizar la evaluación. Un modelo de evaluación alternativo no evaluaría los operandos hasta que se necesitaran sus valores. En su lugar, primero sustituiría expresiones de operandos por parámetros hasta obtener una expresión que involucrara solo operadores primitivos y luego realizaría la evaluación. Si usáramos este método, la evaluación de

(f5)

Procedería según la secuencia de expansiones

(suma de cuadrados (+ 5 1) (\* 5 2))   
  
(+ (cuadrado (+ 5 1)) (cuadrado (\* 5 2)) )   
  
(+ (\* (+ 5 1) (+ 5 1)) (\* (\* 5 2) (\* 5 2)))

seguido de las reducciones

(+ (\* 6 6) (\* 10 10))   
  
(+ 36 100)   
  
                    136

Esto da la misma respuesta que nuestro modelo de evaluación anterior, pero el proceso es diferente. En particular, las evaluaciones de (+ 5 1) y (\* 5 2) se realizan aquí dos veces, lo que corresponde a la reducción de la expresión

(\* x x)

con x reemplazado respectivamente por (+ 5 1) y (\* 5 2) .

Este método de evaluación alternativo de "expandir completamente y luego reducir" se conoce como*evaluación de orden normal* , en contraste con el método "evaluar los argumentos y luego aplicar" que el intérprete realmente utiliza, que se llama*evaluación de orden aplicativo* . Se puede demostrar que, para aplicaciones de procedimientos que se pueden modelar mediante sustitución (incluidos todos los procedimientos de los dos primeros capítulos de este libro) y que arrojan valores legítimos, la evaluación de orden normal y de orden aplicativo producen el mismo valor. (Véase el ejercicio  [1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.5) para un ejemplo de un valor ``ilegítimo'' donde la evaluación de orden normal y de orden aplicativo no arrojan el mismo resultado.)

Lisp utiliza la evaluación en orden aplicativo, en parte debido a la eficiencia adicional que se obtiene al evitar evaluaciones múltiples de expresiones como las ilustradas con (+ 5 1) y (\* 5 2) arriba y, más significativamente, porque la evaluación en orden normal se vuelve mucho más complicada de manejar cuando dejamos el ámbito de los procedimientos que se pueden modelar por sustitución. Por otro lado, la evaluación en orden normal puede ser una herramienta extremadamente valiosa, e investigaremos algunas de sus implicaciones en los capítulos 3 y 4. [16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_23)

**[1.1.6 Expresiones condicionales y predicados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.1.6)**

El poder expresivo de la clase de procedimientos que podemos definir en este punto es muy limitado, porque no tenemos forma de hacer pruebas y realizar diferentes operaciones dependiendo del resultado de una prueba. Por ejemplo, no podemos definir un procedimiento que calcule laValor absoluto de un número probando si el número es positivo, negativo o cero y tomando diferentes acciones en los diferentes casos según la regla

Forma

El contenido generado por IA puede ser incorrecto.

Esta construcción se denomina *análisis de caso* y existe una forma especial en Lisp para notar este tipo de análisis de caso. Se denominacond (que significa "condicional") y se utiliza de la siguiente manera:

(define (abs x)   
  (cond ((> x 0) x)   
        ((= x 0) 0)   
        ((< x 0) (- x))))

La forma general de una expresión condicional es

(cond. (< *p 1* > < *e 1* >)   
      (< *p 2* > < *e 2* >)   
        
      (< *p n* > < *e n* >))

que consiste en el símbolo cond seguido depares de expresiones entre paréntesis (< *p* > < *e* >) llamados*cláusulas* . La primera expresión de cada par es una*predicado* , es decir, una expresión cuyo valor se interpreta como verdadero o falso. [17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_24)

Las expresiones condicionales se evalúan de la siguiente manera. Primero se evalúa el predicado < *p 1* >. Si su valor es falso, entonces se evalúa < *p 2* >. Si el valor de < *p 2 > también es falso, entonces se evalúa < p 3* >. Este proceso continúa hasta que se encuentra un predicado cuyo valor es verdadero, en cuyo caso el intérprete devuelve el valor del predicado correspondiente.*Expresión consecuente* < *e* > de la cláusula como valor de la expresión condicional. Si ninguna de las < *p* > resulta verdadera, el valor de la cond no está definido.

La palabra *predicado* se utiliza para procedimientos que devuelven verdadero o falso, así como para expresiones que evalúan como verdaderas o falsas. El procedimiento de valor absoluto abs hace uso de lapredicados primitivos > , < , y = . [18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_25) Estos toman dos números como argumentos y prueban si el primer número es, respectivamente, mayor que, menor que o igual al segundo número, devolviendo verdadero o falso según corresponda.

Otra forma de escribir el procedimiento de valor absoluto es

(define (abs x)   
  (cond ((< x 0) (- x))   
        (de lo contrario x)))

que podría expresarse en español como "Si *x* es menor que cero, devuelve - *x* ; de lo contrario, devuelve *x* ".Else es un símbolo especial que se puede utilizar en lugar de < *p* > en la cláusula final de una cond . Esto hace que la cond devuelva como valor el valor de la < *e* > correspondiente siempre que se hayan omitido todas las cláusulas anteriores. De hecho, cualquier expresión que siempre dé como resultado un valor verdadero se podría utilizar como < *p* > aquí.

Aquí hay otra forma de escribir el procedimiento de valor absoluto:

(define (abs x)   
  (si (< x 0)   
      (- x)   
      x))

Esto utiliza la forma especial if , un tipo restringido de condicional que se puede utilizar cuando hay precisamenteDos casos en el análisis de casos. La forma general de una expresión if es

(si < *predicado* > < *consecuente* > < *alternativa* >)

Para evaluar una expresión if , el intérprete comienza evaluando la< *predicado* > parte de la expresión. Si el < *predicado* > se evalúa como un valor verdadero, el intérprete evalúa el<consecuente> y devuelve su valor. De lo contrario *,* evalúa el< *alternativa* > y devuelve su valor. [19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_26)

Además de los predicados primitivos como < , = y > , existen operaciones de composición lógica que permiten construir predicados compuestos. Las tres más utilizadas son las siguientes:

* (y < *e 1* > ... < *e n* >)

El intérprete evalúa las expresiones < *e* > una a la vez, en orden de izquierda a derecha. Si alguna expresión < *e* > se evalúa como falsa, el valor de la expresión y es falso y el resto de las expresiones < *e* > no se evalúan. Si todas las expresiones < *e > se evalúan como valores verdaderos, el valor de la expresión* y es el valor de la última expresión.

* (o < *e 1* > ... < *e n* >)

El intérprete evalúa las expresiones < *e* > una a la vez, en orden de izquierda a derecha. Si alguna de las expresiones < *e* > se evalúa como un valor verdadero, ese valor se devuelve como el valor de la expresión or y el resto de las expresiones < *e* > no se evalúan. Si todas las expresiones < *e > se evalúan como falsas, el valor de la expresión* or es falso.

* (no < *e* >)

El valor de una expresión no es verdadero cuando la expresión < *e* > se evalúa como falsa, y falso en caso contrario.

Tenga en cuenta que and y or son formas especiales, no procedimientos, porque no todas las subexpresiones se evalúan necesariamente. Not es un procedimiento ordinario.

Como ejemplo de cómo se utilizan, la condición de que un número *x* esté en el rango 5 < *x* < 10 puede expresarse como

(y (> x 5) (< x 10))

Como otro ejemplo, podemos definir un predicado para probar si un número es mayor o igual que otro como

(define (>= x y)   
  (o (> x y) (= x y)))

o alternativamente como

(define (>= x y)   
  (no (< x y)))

**Ejercicio 1.1.**   A continuación se muestra una secuencia de expresiones. ¿Cuál es el resultado que imprime el intérprete en respuesta a cada expresión? Suponga que la secuencia se debe evaluar en el orden en que se presenta.

10   
(+ 5 3 4)   
(- 9 1)   
(/ 6 2)   
(+ (\* 2 4) (- 4 6))   
(definir a 3)   
(definir b (+ a 1))   
(+ a b (\* a b))   
(= a b)   
(si (y (> b a) (< b (\* a b)))   
    b   
    a)   
(cond ((= a 4) 6)   
      ((= b 4) (+ 6 7 a))   
      (de lo contrario 25))   
(+ 2 (si (> b a) b a))   
(\* (cond ((> a b) a)   
         ((< a b) b)   
         (de lo contrario -1))   
   (+ a 1))

**Ejercicio 1.2.**   Traduce la siguiente expresión a la forma prefijada



**Ejercicio 1.3.**   Defina un procedimiento que tome tres números como argumentos y devuelva la suma de los cuadrados de los dos números mayores.

**Ejercicio 1.4.**  Observe que nuestro modelo de evaluación permite combinaciones cuyos operadores son expresiones compuestas. Utilice esta observación para describir el comportamiento del siguiente procedimiento:

(define (a-más-abs-b a b)   
  ((si (> b 0) + -) a b))

**Ejercicio 1.5.**  Ben Bitdiddle ha inventado una prueba para determinar si el intérprete al que se enfrenta utiliza la evaluación en orden aplicativo o la evaluación en orden normal. Define los dos procedimientos siguientes:

(define (p) (p))   
  
(define (prueba x y)   
  (si (= x 0)   
      0   
      y))

Luego evalúa la expresión

(prueba 0 (p))

¿Qué comportamiento observará Ben con un intérprete que utiliza la evaluación en orden aplicativo? ¿Qué comportamiento observará Ben con un intérprete que utiliza la evaluación en orden normal? Explica tu respuesta.(Suponga que la regla de evaluación para la forma especial if es la misma independientemente de si el intérprete utiliza el orden normal o aplicativo: la expresión de predicado se evalúa primero y el resultado determina si se debe evaluar la expresión consecuente o la alternativa).

**[1.1.7 Ejemplo: raíces cuadradas por el método de Newton](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.1.7)**

Los procedimientos, como se ha explicado anteriormente, son muy parecidos a las funciones matemáticas ordinarias. Especifican un valor que está determinado por uno o más parámetros. Pero existe una diferencia importante entre las funciones matemáticas y los procedimientos informáticos: los procedimientos deben ser eficaces.

Como ejemplo, consideremos el problema de calcular raíces cuadradas. Podemos definir la función raíz cuadrada como



Esto describe una función matemática perfectamente legítima. Podríamos usarla para reconocer si un número es la raíz cuadrada de otro, o para derivar hechos sobre raíces cuadradas en general. Por otra parte, la definición no describe un procedimiento. De hecho, no nos dice casi nada sobre cómo hallar realmente la raíz cuadrada de un número dado. No servirá de nada reformular esta definición en pseudo-Lisp:

(define (sqrt x)   
  (y (y (>= y 0)   
              (= (cuadrado y) x))))

Esto no hace más que plantear la pregunta.

El contraste entre función y procedimiento es un reflejo de la distinción general entre describir propiedades de las cosas y describir cómo hacer las cosas, o, como a veces se la denomina, la distinción entreconocimiento declarativo y conocimiento imperativo. EnEn matemáticas, normalmente nos interesan las descripciones declarativas (qué es), mientras que en informática normalmente nos interesan las descripciones imperativas (cómo hacer). [20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_32)

¿Cómo se calculan las raíces cuadradas? La forma más común es utilizar el método de aproximaciones sucesivas de Newton, que dice que siempre que tengamos una estimación *y* para el valor de la raíz cuadrada de un número *x* , podemos realizar una manipulación simple para obtener una estimación mejor (una más cercana a la raíz cuadrada real) promediando *y* con *x* / *y* . [21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_33) Por ejemplo, podemos calcular la raíz cuadrada de 2 de la siguiente manera. Supongamos que nuestra estimación inicial es 1:

|  |  |  |
| --- | --- | --- |
| Adivinar | Cociente | Promedio |
|  |  |  |
| 1 | (2/1) = 2 | ((2 + 1)/2) = 1,5 |
|  |  |  |
| 1.5 | (2/1,5) = 1,3333 | ((1,3333 + 1,5)/2) = 1,4167 |
|  |  |  |
| 1.4167 | (2/1,4167) = 1,4118 | ((1,4167 + 1,4118)/2) = 1,4142 |
|  |  |  |
| 1.4142 | ... | ... |
|  |  |  |

Continuando con este proceso obtenemos aproximaciones cada vez mejores a la raíz cuadrada.

Ahora formalicemos el proceso en términos de procedimientos. Empezamos con un valor para elradicando (el número cuya raíz cuadrada estamos tratando de calcular) y un valor para la suposición. Si la suposición es lo suficientemente buena para nuestros propósitos, hemos terminado; si no, debemos repetir el proceso con una suposición mejorada. Escribimos esta estrategia básica como un procedimiento:

(define (sqrt-iter suposición x)   
  (si (¿suficientemente bueno? suposición x)   
      suposición   
      (sqrt-iter (mejora suposición x)   
                 x)))

Una suposición se mejora al promediarla con el cociente del radicando y la suposición anterior:

(definir (mejorar conjetura x)   
  (conjetura promedio (/ x conjetura)))

dónde

(definir (promedio x y)   
  (/ (+ x y) 2))

También tenemos que decir qué queremos decir con "suficientemente bueno". Lo siguiente servirá como ilustración, pero en realidad no es una prueba muy buena. (Véase el ejercicio  [1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.7) .) La idea es mejorar la respuesta hasta que esté lo suficientemente cerca como para que su cuadrado difiera del radicando en menos de una tolerancia predeterminada (aquí 0,001): [22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_34)

(define (¿suficientemente bueno? adivina x)   
  (< (abs (- (cuadrado adivinado) x)) 0,001))

Por último, necesitamos una forma de empezar. Por ejemplo, siempre podemos adivinar que la raíz cuadrada de cualquier número es 1: [23.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_35)

(define (sqrt x)   
  (sqrt-iter 1.0 x))

Si escribimos estas definiciones al intérprete, podemos usar sqrt del mismo modo que podemos usar cualquier procedimiento:

(sqrt 9)   
*3.00009155413138*  
(sqrt (+ 100 37))   
*11.704699917758145*  
(sqrt (+ (sqrt 2) (sqrt 3)))   
*1.7739279023207892*  
(cuadrado (sqrt 1000))   
*1000.000369924366*

El programa sqrt también ilustra que el lenguaje procedimental simple que hemos introducido hasta ahora es suficiente para escribir cualquier programa puramente numérico que se pueda escribir, por ejemplo, en C o Pascal. Esto puede parecer sorprendente, ya que no hemos incluido en nuestro lenguaje ningún procedimiento iterativo.(bucles) construcciones que le indican al ordenador que haga algo una y otra vez. Sqrt-iter , por otro lado, demuestra cómo se puede lograr la iteración sin usar ninguna construcción especial más allá de la capacidad habitual de llamar a un procedimiento. [24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_36)

**Ejercicio 1.6.**  Alyssa P. Hacker no entiende por qué es necesario proporcionar if como una forma especial. “¿Por qué no puedo simplemente definirlo como un procedimiento ordinario en términos de cond ?”, pregunta. La amiga de Alyssa, Eva Lu Ator, afirma que esto sí se puede hacer y define una nueva versión de if :

(define (nuevo-si predicado entonces cláusula-cláusula-sino)   
  (cond (predicado entonces cláusula-cláusula)   
        (de lo contrario cláusula-sino)))

Eva le demuestra el programa a Alyssa:

(nuevo-si (= 2 3) 0 5)   
*5*  
  
(nuevo-si (= 1 1) 0 5)   
*0*

Encantada, Alyssa usa new-if para reescribir el programa de raíz cuadrada:

(define (sqrt-iter conjetura x)   
  (nuevo-si (¿suficientemente bueno? conjetura x)   
          conjetura   
          (sqrt-iter (mejora conjetura x)   
                     x)))

¿Qué sucede cuando Alyssa intenta usar esto para calcular raíces cuadradas? Explícalo.

**Ejercicio 1.7.**   La prueba de "suficientemente bueno" que se utiliza para calcular raíces cuadradas no será muy eficaz para encontrar las raíces cuadradas de números muy pequeños. Además, en las computadoras reales, las operaciones aritméticas casi siempre se realizan con precisión limitada. Esto hace que nuestra prueba sea inadecuada para números muy grandes. Explique estas afirmaciones, con ejemplos que muestren cómo la prueba falla para números pequeños y grandes. Una estrategia alternativa para implementar la prueba de "suficientemente bueno" es observar cómo cambia la estimación de una iteración a la siguiente y detenerse cuando el cambio sea una fracción muy pequeña de la estimación. Diseñe un procedimiento de raíz cuadrada que utilice este tipo de prueba final. ¿Funciona mejor para números pequeños y grandes?

**Ejercicio 1.8.**  El método de Newton para raíces cúbicas se basa en el hecho de que si *y* es una aproximación a la raíz cúbica de *x* , entonces una mejor aproximación viene dada por el valor



Utilice esta fórmula para implementar un procedimiento de raíz cúbica análogo al procedimiento de raíz cuadrada. (En la sección  [1.3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.4) veremos cómo implementar el método de Newton en general como una abstracción de estos procedimientos de raíz cuadrada y raíz cúbica).

**[1.1.8 Procedimientos como abstracciones de caja negra](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.1.8)**

Sqrt es nuestro primer ejemplo de un proceso definido por un conjunto de procedimientos mutuamente definidos. Observe que la definición de sqrt-iter es*recursiva* ; es decir, el procedimiento se define en términos de sí mismo. La idea de poder definir un procedimiento en términos de sí mismo puede resultar inquietante; puede parecer poco claro cómo una definición tan "circular" podría tener sentido, y mucho menos especificar un proceso bien definido que debe ser llevado a cabo por una computadora. Esto se abordará con más detalle en la sección  [1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2) . Pero primero consideremos algunos otros puntos importantes ilustrados por el ejemplo de sqrt .

Observe que el problema de calcular raíces cuadradas se divide naturalmente en una serie de subproblemas: cómo determinar si una suposición es lo suficientemente buena, cómo mejorar una suposición, etc. Cada una de estas tareas se logra mediante un procedimiento separado. El programa sqrt completo puede verse como un conjunto de procedimientos (mostrados en la figura  [1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_1.2) ) que reflejan la descomposición del problema en subproblemas.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 1.2:**   Descomposición procedimental del programa sqrt . |

La importancia de esta estrategia de descomposición no radica simplemente en que se divida el programa en partes. Después de todo, podríamos tomar cualquier programa grande y dividirlo en partes: las primeras diez líneas, las siguientes diez líneas, las siguientes diez líneas, y así sucesivamente. Más bien, es crucial que cada procedimiento realice una tarea identificable que pueda usarse como módulo para definir otros procedimientos.Por ejemplo, cuando definimos el procedimiento ¿suficientemente bueno? en términos del cuadrado , podemos considerar el procedimiento del cuadrado como un``caja negra''. En ese momento no nos preocupa *cómo* calcula el procedimiento su resultado, sino sólo el hecho de que calcula el cuadrado. Los detalles de cómo se calcula el cuadrado pueden suprimirse para considerarlos en un momento posterior. De hecho, en lo que respecta al procedimiento ¿suficientemente bueno?, el cuadrado no es exactamente un procedimiento sino más bien una abstracción de un procedimiento, un llamado*Abstracción procedimental* . En este nivel de abstracción, cualquier procedimiento que calcule el cuadrado es igualmente bueno.

Por lo tanto, considerando únicamente los valores que devuelven, los dos procedimientos siguientes para elevar al cuadrado un número deberían ser indistinguibles. Cada uno toma un argumento numérico y produce el cuadrado de ese número como valor. [25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_40)

(define (cuadrado x) (\* x x))   
  
(define (cuadrado x)    
  (exp (doble (log x))))   
  
(define (doble x) (+ x x))

Por lo tanto, la definición de un procedimiento debería permitir suprimir los detalles. Es posible que los usuarios del procedimiento no hayan escrito el procedimiento ellos mismos, sino que lo hayan obtenido de otro programador como una caja negra. Un usuario no debería necesitar saber cómo se implementa el procedimiento para poder usarlo.

**[Nombres locales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_41)**

Un detalle de la implementación de un procedimiento que no debería importarle al usuario del procedimiento es la elección de nombres que haga el implementador para los parámetros formales del procedimiento. Por lo tanto, los siguientes procedimientos no deberían ser distinguibles:

(define (cuadrado x) (\* x x))   
  
(define (cuadrado y) (\* y y))

Este principio (que el significado de un procedimiento debe ser independiente de los nombres de los parámetros utilizados por su autor) parece evidente a primera vista, pero sus consecuencias son profundas. La consecuencia más simple es que los nombres de los parámetros de un procedimiento deben ser locales al cuerpo del procedimiento. Por ejemplo, usamos cuadrado en la definición de ¿suficientemente bueno? en nuestro procedimiento de raíz cuadrada:

(define (¿suficientemente bueno? adivina x)   
  (< (abs (- (cuadrado adivinado) x)) 0,001))

La intención del autor de ¿suficientemente bueno? es determinar si el cuadrado del primer argumento está dentro de una tolerancia dada del segundo argumento. Vemos que el autor de ¿ suficientemente bueno? usó el nombre guess para referirse al primer argumento y x para referirse al segundo argumento. El argumento de square es guess . Si el autor de square usó x (como se indica arriba) para referirse a ese argumento, vemos que la x en ¿suficientemente bueno? debe ser una x diferente a la de square . La ejecución del procedimiento square no debe afectar el valor de x que usa ¿ suficientemente bueno? , porque ese valor de x puede ser necesario para ¿suficientemente bueno? después de que square haya terminado de calcular.

Si los parámetros no fueran locales a los cuerpos de sus respectivos procedimientos, entonces el parámetro x en cuadrado podría confundirse con el parámetro x en ¿suficientemente bueno?, y el comportamiento de ¿suficientemente bueno? dependería de qué versión de cuadrado usáramos. Por lo tanto, cuadrado no sería la caja negra que deseábamos.

Un parámetro formal de un procedimiento tiene un papel muy especial en la definición del procedimiento, ya que no importa el nombre que tenga el parámetro formal. Ese nombre se denomina*variable ligada* , y decimos que la definición del procedimiento*vincula* sus parámetros formales. El significado de una definición de procedimiento no cambia si una variable vinculada se renombra de manera consistente a lo largo de la definición. [26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_42) Si una variable no está vinculada, decimos que es*libre* . El conjunto de expresiones para las que un enlace define un nombre se denomina*alcance* de ese nombre. En una definición de procedimiento, las variables enlazadas declaradas comoLos parámetros formales del procedimiento tienen como alcance el cuerpo del procedimiento.

En la definición de ¿suficientemente bueno? anterior, guess y x son variables ligadas, pero < , - , abs y square son libres. El significado de ¿suficientemente bueno? debería ser independiente de los nombres que elijamos para guess y x siempre que sean distintos y diferentes de < , - , abs y square . (Si renombráramos guess a abs habríamos introducido un error al*capturando* la variable abs . Habría cambiado de libre a limitada.) Sin embargo, el significado de ¿suficientemente bueno? no es independiente de los nombres de sus variables libres. Seguramente depende del hecho (externo a esta definición) de que el símbolo abs nombra un procedimiento para calcular el valor absoluto de un número. ¿Suficientemente bueno? calculará una función diferente si sustituimos cos por abs en su definición.

**[Definiciones internas y estructura de bloques](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_43)**

Hasta ahora, tenemos disponible un tipo de aislamiento de nombres: los parámetros formales de un procedimiento son locales al cuerpo del procedimiento. El programa de raíz cuadrada ilustra otra forma en la que nos gustaría controlar el uso de nombres.El programa actual consta de procedimientos separados:

(define (sqrt x)   
  (sqrt-iter 1.0 x))   
(define (sqrt-iter conjetura x)   
  (si (¿suficientemente bueno? conjetura x)   
      conjetura   
      (sqrt-iter (mejora conjetura x) x)))   
(define (¿suficientemente bueno? conjetura x)   
  (< (abs (- (conjetura al cuadrado) x)) 0.001))   
(define (mejora conjetura x)   
  (conjetura promedio (/ x conjetura)))

El problema con este programa es que el único procedimiento que es importante para los usuarios de sqrt es sqrt . Los otros procedimientos ( sqrt-iter , good-enough? y improve ) sólo les llenan la mente. No pueden definir ningún otro procedimiento llamado good-enough? como parte de otro programa para trabajar junto con el programa de raíz cuadrada, porque sqrt lo necesita. El problema es especialmente grave en la construcción de grandes sistemas por muchos programadores separados. Por ejemplo, en la construcción de una gran biblioteca de procedimientos numéricos, muchas funciones numéricas se calculan como aproximaciones sucesivas y, por lo tanto, podrían tener procedimientos llamados good-enough? y improve como procedimientos auxiliares. Nos gustaría localizar los subprocedimientos, ocultándolos dentro de sqrt para que sqrt pudiera coexistir con otras aproximaciones sucesivas, cada una con su propio procedimiento privado good-enough ?. Para hacer esto posible, permitimos que un procedimiento tengadefiniciones internas que son locales a ese procedimiento. Por ejemplo, en el problema de la raíz cuadrada podemos escribir

(define (sqrt x)   
  (define (¿suficientemente bueno? suposición x)   
    (< (abs (- (suposición al cuadrado) x)) 0.001))   
  (define (mejora suposición x)   
    (suposición promedio (/ x suposición)))   
  (define (sqrt-iter suposición x)   
    (si (¿suficientemente bueno? suposición x)   
        suposición   
        (sqrt-iter (mejora suposición x) x)))   
  (sqrt-iter 1.0 x))

Esta anidación de definiciones, llamada *estructura de bloques* , es básicamente la solución correcta para el problema de empaquetamiento de nombres más simple. Pero hay una mejor idea escondida aquí. Además de internalizar las definiciones de los procedimientos auxiliares, podemos simplificarlos. Dado que x está ligado en la definición de sqrt , los procedimientos good-enough?, improve y sqrt -iter , que están definidos internamente en sqrt , están en el alcance de x . Por lo tanto, no es necesario pasar x explícitamente a cada uno de estos procedimientos. En su lugar, permitimos que x sea unvariable libre en las definiciones internas, como se muestra a continuación. Entonces x obtiene su valor del argumento con el que se llama al procedimiento envolvente sqrt . Esta disciplina se llama*alcance léxico* . [27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_44)

(define (sqrt x)   
  (define (¿suficientemente bueno? conjetura)   
    (< (abs (- (conjetura al cuadrado) x)) 0.001))   
  (define (mejora conjetura)   
    (conjetura promedio (/ x conjetura)))   
  (define (sqrt-iter conjetura)   
    (si (¿suficientemente bueno? conjetura)   
        conjetura   
        (sqrt-iter (mejora conjetura))))   
  (sqrt-iter 1.0))

Usaremos ampliamente la estructura de bloques para ayudarnos a dividir programas grandes en partes manejables. [28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_45) La idea de la estructura de bloques se originó con el lenguaje de programaciónAlgol 60. Aparece en la mayoría de los lenguajes de programación avanzados y es una herramienta importante para ayudar a organizar la construcción de programas grandes.

[4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_10) La caracterización de los números como "datos simples" es una farsa descarada. De hecho, el tratamiento de los números es uno de los aspectos más complicados y confusos de cualquier lenguaje de programación. Algunos de los problemas típicos que se plantean son los siguientes:Algunos sistemas informáticos distinguen entre *números enteros* , como 2, y *números reales* , como 2,71. ¿El número real 2,00 es diferente del entero 2? ¿Las operaciones aritméticas que se utilizan para los números enteros son las mismas que las que se utilizan para los números reales? ¿6 dividido por 2 produce 3 o 3,0? ¿Qué número tan grande podemos representar? ¿Cuántos decimales de precisión podemos representar? ¿El rango de los números enteros es el mismo que el rango de los números reales?Más allá de estas cuestiones, por supuesto, hay una serie de cuestiones relacionadas con los errores de redondeo y truncamiento, es decir, toda la ciencia del análisis numérico. Dado que en este libro nos centramos en el diseño de programas a gran escala y no en las técnicas numéricas, vamos a ignorar estos problemas. Los ejemplos numéricos de este capítulo mostrarán el comportamiento de redondeo habitual que se observa cuando se utilizan operaciones aritméticas que conservan un número limitado de decimales de precisión en operaciones no enteras.

[5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_11) A lo largo de este libro,Cuando deseamos enfatizar la distinción entre la entrada escrita por el usuario y la respuesta impresa por el intérprete, mostraremos esta última en caracteres inclinados.

[6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_12) Los sistemas Lisp suelen ofrecer funciones que ayudan al usuario a formatear expresiones. Dos funciones especialmente útiles son una que sangra automáticamente hasta la posición de letra bonita adecuada cada vez que se inicia una nueva línea y otra que resalta el paréntesis izquierdo correspondiente cada vez que se escribe un paréntesis derecho.

[7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_13) Lisp obedece a la convención de que cada expresión tiene un valor. Esta convención, junto con la antigua reputación de Lisp como lenguaje ineficiente, es la fuente de la ocurrencia de Alan Perlis (parafraseando a Oscar Wilde) de que «los programadores de Lisp conocen el valor de todo pero el costo de nada».

[8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_14) En este libro, no mostramos la respuesta del intérprete a la evaluación de definiciones, ya que esto depende en gran medida de la implementación.

[El](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_15) capítulo 3 mostrará que esta noción de entorno es crucial, tanto para entender cómo trabaja el intérprete como para implementarlo.

[10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_16) Puede parecer extraño que la regla de evaluación diga, como parte del primer paso, que debemos evaluar el elemento más a la izquierda de una combinación, ya que en este punto ese solo puede ser un operador como+o\*que represente un procedimiento primitivo incorporado como la suma o la multiplicación. Veremos más adelante que es útil poder trabajar con combinaciones cuyos operadores son en sí mismos expresiones compuestas.

[11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_17) Las formas sintácticas especiales que son simplemente estructuras superficiales alternativas convenientes para cosas que pueden escribirse de maneras más uniformes a veces se denominan *azúcar sintáctica* , para usar una frase acuñada por Peter Landin. En comparación con los usuarios de otros lenguajes, los programadores de Lisp, por regla general, están menos preocupados por los asuntos de sintaxis. (Por el contrario, examine cualquier manual de Pascal y observe cuánto de él está dedicado a descripciones de sintaxis). Este desdén por la sintaxis se debe en parte a la flexibilidad de Lisp, que hace que sea fácil cambiar la sintaxis superficial, y en parte a la observación de que muchas construcciones sintácticas "convenientes", que hacen que el lenguaje sea menos uniforme, terminan causando más problemas de los que valen cuando los programas se vuelven grandes y complejos. En palabras de Alan Perlis, "el azúcar sintáctico causa cáncer del punto y coma".

[12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_18) Observe que aquí se combinan dos operaciones diferentes: estamos creando el procedimiento y le estamos dando el nombrecuadrado. Es posible, y de hecho es importante, poder separar estas dos nociones: crear procedimientos sin nombrarlos y dar nombres a los procedimientos que ya se han creado. Veremos cómo hacer esto en la sección [1.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.2).

[13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_19) A lo largo de este libro, vamos adescribir la sintaxis general de expresiones utilizando símbolos en cursiva delimitados por corchetes angulares --por ejemplo, <nombre> *--* para indicar los "espacios" en la expresión que deben completarse cuando dicha expresión se utiliza realmente.

[14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_20) MásEn general, el cuerpo del procedimiento puede ser una secuencia de expresiones. En este caso, el intérprete evalúa cada expresión de la secuencia por turno y devuelve el valor de la expresión final como valor de la aplicación del procedimiento.

[15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_21) A pesar de la simplicidad de la idea de sustitución, resulta sorprendentemente complicado dar una definición matemática rigurosa del proceso de sustitución. El problema surge de la posibilidad de confusión entre los nombres utilizados para los parámetros formales de un procedimiento y los nombres (posiblemente idénticos) utilizados en las expresiones a las que se puede aplicar el procedimiento. De hecho, existe una larga historia de definiciones erróneas de*sustitución*en la literatura de lógica y semántica de programación.Véase Stoy 1977 para un análisis cuidadoso de la sustitución.

[16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_23) En el capítulo 3, presentaremos*el procesamiento de flujo*, que es una forma de manejar estructuras de datos aparentemente "infinitas" mediante la incorporación de una forma limitada de evaluación de orden normal. En la sección [4.2,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2)modificaremos el intérprete de Scheme para producir una variante de orden normal de Scheme.

[17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_24) "Interpretado como verdadero o falso" significa lo siguiente: En Scheme, hay dos valores distinguidos que se denotan por las constantes #t y #f . Cuando el intérprete comprueba el valor de un predicado, interpreta #f como falso. Cualquier otro valor se trata como verdadero. (Por lo tanto, proporcionar #t es lógicamente innecesario, pero es conveniente). En este libro usaremos los nombres true y false , que están asociados con los valores #t y #f respectivamente.

[18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_25) Abstambién utilizael operador «menos» - , que, cuando se utiliza con un solo operando, como en (- x) , indica negación.

[19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_26) Una pequeña diferenciaLa diferencia entre if y cond es que la parte < *e* > de cada cláusula cond puede ser una secuencia de expresiones. Si se determina que la < *p* > correspondiente es verdadera, las expresiones < *e* > se evalúan en secuencia y el valor de la expresión final en la secuencia se devuelve como el valor de la cond . Sin embargo, en una expresión if , la < *consecuente* > y la < *alternativa* > deben ser expresiones individuales.

[20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_32) Las descripciones declarativas e imperativas están íntimamente relacionadas, como lo están también las matemáticas y la informática. Por ejemplo, decir que la respuesta que produce un programa es"Correcto" es hacer una declaración sobre el programa. Existe una gran cantidad de investigaciones destinadas a establecer técnicas paraDemostrar que los programas son correctos, y gran parte de la dificultad técnica de este tema tiene que ver con negociar la transición entre enunciados imperativos (a partir de los cuales se construyen los programas) y enunciados declarativos (que se pueden utilizar para deducir cosas). En una línea relacionada, un área actual importante en el diseño de lenguajes de programación es la exploración de los llamadosLenguajes de muy alto nivel, en los que se programa en términos de enunciados declarativos. La idea es hacer que los intérpretes sean lo suficientemente sofisticados como para que, dado el conocimiento de «qué es» especificado por el programador, puedan generar automáticamente el conocimiento de «cómo». Esto no se puede hacer en general, pero hay áreas importantes en las que se ha avanzado. Volveremos a tratar esta idea en el capítulo 4.

[21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_33) Este algoritmo de raíz cuadrada es en realidad un caso especial del método de Newton, que es una técnica general para hallar raíces de ecuaciones. El algoritmo de raíz cuadrada en sí fue desarrollado por Heron deAlejandría en el siglo I d . C. Veremos cómo expresar el método general de Newton como un procedimiento Lisp en la sección  [1.3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.4) .

[22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_34) Generalmente daremosLos nombres de los predicados terminan con signos de interrogación, para ayudarnos a recordar que son predicados. Esto es solo una convención estilística. En lo que respecta al intérprete, el signo de interrogación es solo un carácter común.

[23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_35) Observe que expresamos nuestra estimación inicial como 1.0 en lugar de 1. Esto no haría ninguna diferencia en muchas implementaciones de Lisp.Sin embargo, el esquema MIT distingue entre números enteros exactos y valores decimales, y dividir dos números enteros produce un número racional en lugar de un decimal. Por ejemplo, dividir 10 por 6 da como resultado 5/3, mientras que dividir 10,0 por 6,0 da como resultado 1,6666666666666667. (Aprenderemos cómo implementar la aritmética en números racionales en la sección  [2.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.1) ). Si comenzamos con una estimación inicial de 1 en nuestro programa de raíz cuadrada, y *x* es un entero exacto, todos los valores posteriores producidos en el cálculo de la raíz cuadrada serán números racionales en lugar de decimales. Las operaciones mixtas en números racionales y decimales siempre dan como resultado decimales, por lo que comenzar con una estimación inicial de 1,0 obliga a que todos los valores posteriores sean decimales.

[Los](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_36) lectores que estén preocupados por los problemas de eficiencia involucrados en el uso de llamadas a procedimientos para implementar la iteración deben tener en cuenta las observaciones sobre la "recursión de cola" en la sección [1.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.1).

[25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_40) Ni siquiera está claro cuál de estos procedimientos es una implementación más eficiente. Esto depende del hardware disponible. Hay máquinas para las cuales la implementación "obvia" es la menos eficiente. Consideremos una máquina que tiene extensas tablas de logaritmos y antilogaritmos almacenados de una manera muy eficiente.

[26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_42) El concepto de cambio de nombre consistente es en realidad sutil y difícil de definir formalmente. Lógicos famosos han cometido errores embarazosos en este aspecto.

[El](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_44) alcance léxico dicta que las variables libres en un procedimiento se toman como referencia a los enlaces realizados al incluir definiciones de procedimientos; es decir, se buscan enEl entorno en el que se definió el procedimiento. Veremos cómo funciona esto en detalle en el capítulo 3, cuando estudiemos los entornos y el comportamiento detallado del intérprete.

[28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_45) Definiciones incorporadasDebe aparecer primero en el cuerpo de un procedimiento. La dirección no es responsable de las consecuencias de ejecutar programas que entrelazan definición y uso.

[**1.2 Procedimientos y los procesos que generan**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_1.2)

Hemos considerado los elementos de la programación: hemos utilizado operaciones aritméticas primitivas, hemos combinado estas operaciones y hemos abstraído estas operaciones compuestas definiéndolas como procedimientos compuestos. Pero eso no es suficiente para permitirnos decir que sabemos programar. Nuestra situación es análoga a la de alguien que ha aprendido las reglas sobre cómo se mueven las piezas en el ajedrez pero no sabe nada sobre aperturas típicas, tácticas o estrategias. Al igual que el ajedrecista novato, aún no conocemos los patrones comunes de uso en el dominio. Carecemos del conocimiento de qué movimientos vale la pena hacer (qué procedimientos vale la pena definir). Carecemos de la experiencia para predecir las consecuencias de hacer un movimiento (ejecutar un procedimiento).

La capacidad de visualizar las consecuencias de las acciones que se están considerando es crucial para convertirse en un programador experto, al igual que lo es en cualquier actividad creativa y sintética. Para convertirse en un fotógrafo experto, por ejemplo, uno debe aprender a mirar una escena y saber qué tan oscura aparecerá cada región en una copia para cada posible elección de condiciones de exposición y revelado. Solo entonces se puede razonar hacia atrás, planificando el encuadre, la iluminación, la exposición y el revelado para obtener los efectos deseados. Lo mismo ocurre con la programación, donde planificamos el curso de acción que debe seguir un proceso y donde controlamos el proceso por medio de un programa. Para convertirnos en expertos, debemos aprender a visualizar los procesos generados por varios tipos de procedimientos. Solo después de haber desarrollado esta habilidad podemos aprender a construir de manera confiable programas que muestren el comportamiento deseado.

Un procedimiento es un patrón para la *evolución local* de un proceso computacional. Especifica cómo cada etapa del proceso se construye a partir de la etapa anterior. Nos gustaría poder hacer afirmaciones sobre el comportamiento general o *global* de un proceso cuya evolución local ha sido especificada por un procedimiento. Esto es muy difícil de hacer en general, pero al menos podemos intentar describir algunos patrones típicos de evolución de procesos.

En esta sección examinaremos algunas "formas" comunes para los procesos generados por procedimientos simples. También investigaremos las tasas a las que estos procesos consumen los importantes recursos computacionales de tiempo y espacio. Los procedimientos que consideraremos son muy simples. Su papel es similar al que desempeñan los patrones de prueba en fotografía: como patrones prototípicos simplificados en exceso, en lugar de ejemplos prácticos por derecho propio.

**[1.2.1 Recursión lineal e iteración](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.2.1)**

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 1.3:**   Un proceso recursivo lineal para calcular 6!. |

Comenzamos considerando la función factorial, definida por



Hay muchas formas de calcular factoriales. Una forma es utilizar la observación de que *n* ! es igual a *n* veces ( *n* - 1)! para cualquier entero positivo  *n* :



Por lo tanto, podemos calcular *n* ! calculando ( *n* - 1)! y multiplicando el resultado por *n* . Si añadimos la condición de que 1! es igual a 1, esta observación se traduce directamente en un procedimiento:

(define (factorial n)   
  (si (= n 1)   
      1   
      (\* n (factorial (- n 1)))))

Podemos utilizar el modelo de sustitución de la sección  [1.1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.5) para observar este procedimiento en acción calculando 6!, como se muestra en la figura  [1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_1.3) .

Ahora, adoptemos una perspectiva diferente sobre el cálculo de factoriales. Podríamos describir una regla para calcular *n* ! especificando que primero multiplicamos 1 por 2, luego multiplicamos el resultado por 3, luego por 4, y así sucesivamente hasta llegar *a n* . De manera más formal, mantenemos un producto en ejecución, junto con un contador que cuenta desde 1 hasta *n* . Podemos describir el cálculo diciendo que el contador y el producto cambian simultáneamente de un paso al siguiente de acuerdo con la regla.

contador de producto · producto

contador contador + 1

y estipulando que *n* ! es el valor del producto cuando el contador excede *n* .

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 1.4:**   Un proceso iterativo lineal para calcular 6!. |

Una vez más, podemos reformular nuestra descripción como un procedimiento para calcular factoriales: [29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_46)

(define (factorial n)   
  (iterador de hechos 1 1 n))   
  
(define (iterador de hechos producto contador recuento máximo)   
  (si (> contador recuento máximo)   
      producto   
      (iterador de hechos (\* contador producto)   
                 (+ contador 1)   
                 recuento máximo)))

Como antes, podemos utilizar el modelo de sustitución para visualizar el proceso de cálculo de 6!, como se muestra en la figura  [1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_1.4) .

Comparemos los dos procesos. Desde un punto de vista, no parecen muy diferentes. Ambos calculan la misma función matemática en el mismo dominio, y cada uno requiere un número de pasos proporcional a *n* para calcular *n* !. De hecho, ambos procesos incluso llevan a cabo la misma secuencia de multiplicaciones, obteniendo la misma secuencia de productos parciales. Por otra parte, cuando consideramos la"formas" de los dos procesos, encontramos que evolucionan de manera bastante diferente.

Consideremos el primer proceso. El modelo de sustitución revela una forma de expansión seguida de contracción, indicada por la flecha en la figura  [1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_1.3) . La expansión ocurre a medida que el proceso construye una cadena de*Operaciones diferidas* (en este caso, una cadena de multiplicaciones). La contracción se produce a medida que se realizan las operaciones. Este tipo de proceso, caracterizado por una cadena de operaciones diferidas, se denomina*proceso recursivo* . Para llevar a cabo este proceso es necesario que el intérprete lleve un registro de las operaciones que se realizarán posteriormente. En el cálculo de *n* !, la longitud de la cadena de multiplicaciones diferidas y, por lo tanto, la cantidad de información necesaria para llevar un registro de ella,crece linealmente con *n* (es proporcional a *n* ), al igual que el número de pasos.Un proceso de este tipo se denomina *proceso recursivo lineal* .

Por el contrario, el segundo proceso no crece ni se encoge. En cada paso, todo lo que necesitamos controlar, para cualquier *n* , son los valores actuales de las variables product , counter y max-count . A esto lo llamamos un*proceso iterativo* . En general, un proceso iterativo es aquel cuyo estado puede resumirse mediante un número fijo de*variables de estado* , junto con una regla fija que describe cómo se deben actualizar las variables de estado a medida que el proceso pasa de un estado a otro y una prueba final (opcional) que especifica las condiciones bajo las cuales el proceso debe terminar. Al calcular *n* !, la cantidad de pasos necesarios crece linealmente con *n* . Un proceso de este tipo se denomina*proceso iterativo lineal* .

El contraste entre los dos procesos se puede ver de otra manera. En el caso iterativo, las variables del programa proporcionan una descripción completa del estado del proceso en cualquier punto. Si detuviéramos el cálculo entre los pasos, todo lo que tendríamos que hacer para reanudarlo sería proporcionar al intérprete los valores de las tres variables del programa. No sucede lo mismo con el proceso recursivo. En este caso hay información adicional «oculta», mantenida por el intérprete y no contenida en las variables del programa, que indica «dónde está el proceso» en la negociación de la cadena de operaciones diferidas. Cuanto más larga sea la cadena, más información debe mantenerse. [30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_47)

Al contrastar la iteración y la recursión, debemos tener cuidado de no confundir la noción de*proceso* recursivo con la noción de un *procedimiento* recursivo . Cuando describimos un procedimiento como recursivo, nos referimos al hecho sintáctico de que la definición del procedimiento se refiere (directa o indirectamente) al procedimiento en sí. Pero cuando describimos un proceso como que sigue un patrón que es, digamos, linealmente recursivo, estamos hablando de cómo evoluciona el proceso, no de la sintaxis de cómo se escribe un procedimiento. Puede parecer inquietante que nos refiramos a un procedimiento recursivo como fact-iter como generador de un proceso iterativo. Sin embargo, el proceso realmente es iterativo: su estado es capturado completamente por sus tres variables de estado, y un intérprete necesita realizar un seguimiento de solo tres variables para ejecutar el proceso.

Una razón por la que la distinción entre proceso y procedimiento puede ser confusa es que la mayoría de las implementaciones de lenguajes comunes (incluidosLos lenguajes Ada, Pascal y C están diseñados de tal manera que la interpretación de cualquier procedimiento recursivo consume una cantidad de memoria que crece con el número de llamadas a procedimientos, incluso cuando el proceso descrito es, en principio, iterativo. En consecuencia, estos lenguajes pueden describir procesos iterativos solo recurriendo a funciones especiales.``construcciones de bucle'' como do , repeat , till , for y while . La implementación de Scheme que consideraremos en el capítulo 5 no comparte este defecto. Ejecutará un proceso iterativo en un espacio constante, incluso si el proceso iterativo se describe mediante un procedimiento recursivo. Una implementación con esta propiedad se llama*Con una implementación recursiva de cola* ,La iteración se puede expresar utilizando el mecanismo de llamada a procedimiento ordinario, de modo que las construcciones de iteración especiales son útiles solo comoazúcar sintáctico. [31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_48)

**Ejercicio 1.9.**   Cada uno de los dos procedimientos siguientes define un método para sumar dos números enteros positivos en términos de los procedimientos inc , que incrementa su argumento en 1, y dec , que lo disminuye en 1.

(definir (+ a b)   
  (si (= a 0)   
      b   
      (inc (+ (dec a) b))))   
  
(definir (+ a b)   
  (si (= a 0)   
      b   
      (+ (dec a) (inc b))))

Utilizando el modelo de sustitución, ilustre el proceso generado por cada procedimiento al evaluar (+ 4 5) . ¿Estos procesos son iterativos o recursivos?

**Ejercicio 1.10.**  El siguiente procedimiento calcula una función matemática llamada función de Ackermann.

(definir (A x y)   
  (cond ((= y 0) 0)   
        ((= x 0) (\* 2 y))   
        ((= y 1) 2)   
        (de lo contrario (A (- x 1)   
                 (A x (- y 1))))))

¿Cuáles son los valores de las siguientes expresiones?

(Un 1 10)   
  
(Un 2 4)   
  
(Un 3 3)

Consideremos los siguientes procedimientos, donde A es el procedimiento definido anteriormente:

(definir (f n) (A 0 n))   
  
(definir (g n) (A 1 n))   
  
(definir (h n) (A 2 n))   
  
(definir (k n) (\* 5 n n))

Proporcione definiciones matemáticas concisas para las funciones calculadas mediante los procedimientos f , g y h para valores enteros positivos de *n* . Por ejemplo, (kn) calcula 5 *n* 2 .

**[1.2.2 Recursión de árboles](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.2.2)**

Otro patrón común de cálculo se denomina *recursión de árbol* . Como ejemplo, considere el cálculo de la secuencia deNúmeros de Fibonacci, en los que cada número es la suma de los dos anteriores:



En general, los números de Fibonacci se pueden definir mediante la regla

Forma

El contenido generado por IA puede ser incorrecto.

Podemos traducir inmediatamente esta definición en un procedimiento recursivo para calcular los números de Fibonacci:

(define (fib n)   
  (cond ((= n 0) 0)   
        ((= n 1) 1)   
        (de lo contrario (+ (fib (- n 1))   
                 (fib (- n 2))))))

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 1.5:**   El proceso recursivo de árbol generado en computación (fib 5) . |

Considere el patrón de este cálculo. Para calcular (fib 5) , calculamos (fib 4) y (fib 3) . Para calcular (fib 4) , calculamos (fib 3) y (fib 2) . En general, el proceso evolucionado se parece a un árbol, como se muestra en la figura  [1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_1.5) . Observe que las ramas se dividen en dos en cada nivel (excepto en la parte inferior); esto refleja el hecho de que el procedimiento fib se llama a sí mismo dos veces cada vez que se lo invoca.

Este procedimiento es instructivo como recursión prototípica de un árbol, pero es una forma terrible de calcular los números de Fibonacci porque hace muchos cálculos redundantes. Observe en la figura  [1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_1.5) que todo el cálculo de (fib 3) - casi la mitad del trabajo - está duplicado. De hecho, no es difícil demostrar que el número de veces que el procedimiento calculará (fib 1) o (fib 0) (el número de hojas en el árbol anterior, en general) es precisamente *F i b* ( *n* + 1). Para tener una idea de lo malo que es esto, se puede demostrar que el valor de *F i b* ( *n* )crece exponencialmente con *n* . Más precisamente (ver ejercicio  [1.13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.13) ), *F i b* ( *n* ) es el entero más cercano a *n* / 5, donde



es el*proporción áurea* , que satisface la ecuación



Por lo tanto, el proceso utiliza un número de pasos que crece exponencialmente con la entrada. Por otro lado, el espacio requerido crece solo linealmente con la entrada, porque solo necesitamos llevar un registro de qué nodos están por encima de nosotros en el árbol en cualquier punto del cálculo. En general, el número de pasos requeridos por un proceso recursivo en árbol será proporcional al número de nodos en el árbol, mientras que el espacio requerido será proporcional a la profundidad máxima del árbol.

También podemos formular un proceso iterativo para calcular los números de Fibonacci. La idea es utilizar un par de números enteros *a* y *b* , inicializados en *F i b* (1) = 1 y *F i b* (0) = 0, y aplicar repetidamente las transformaciones simultáneas.



No es difícil demostrar que, después de aplicar esta transformación *n* veces, *a* y *b* serán iguales, respectivamente, a *F i b* ( *n* + 1) y *F i b* ( *n* ). Por lo tanto, podemos calcular los números de Fibonacci de forma iterativa utilizando el procedimiento

(define (fib n)   
  (fib-iter 1 0 n))   
  
(define (fib-iter a b conteo)   
  (si (= conteo 0)   
      b   
      (fib-iter (+ a b) a (- conteo 1))))

Este segundo método para calcular *F i b* ( *n* ) es una iteración lineal. La diferencia en el número de pasos que requieren los dos métodos (uno lineal en *n* , otro que crece tan rápido como el propio *F i b* ( *n* ) ) es enorme, incluso para entradas pequeñas.

De esto no se debe concluir que los procesos recursivos en árboles son inútiles. Cuando consideramos procesos que operan sobre datos estructurados jerárquicamente en lugar de números, encontraremos que la recursión en árboles es una herramienta natural y poderosa. [32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_51) Pero incluso en operaciones numéricas, los procesos recursivos en árboles pueden ser útiles para ayudarnos a entender y diseñar programas. Por ejemplo, aunque el primer procedimiento fib es mucho menos eficiente que el segundo, es más sencillo, siendo poco más que una traducción a Lisp de la definición de la secuencia de Fibonacci. Para formular el algoritmo iterativo fue necesario notar que el cálculo podía reformularse como una iteración con tres variables de estado.

**[Ejemplo: Contar el cambio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_52)**

Basta con un poco de ingenio para idear el algoritmo iterativo de Fibonacci. En cambio, consideremos el siguiente problema: ¿de cuántas maneras diferentes podemos dar cambio de $1,00, dadas las monedas de 50 centavos, 25 centavos, 10 centavos, 5 y 10 centavos? En términos más generales, ¿podemos escribir un procedimiento para calcular la cantidad de maneras de dar cambio a una cantidad determinada de dinero?

Este problema tiene una solución sencilla como procedimiento recursivo. Supongamos que pensamos en los tipos de monedas disponibles como si estuvieran dispuestos en algún orden. Entonces se cumple la siguiente relación:

El número de formas de cambiar la cantidad *a* usando *n* tipos de monedas es igual a

* la cantidad de formas de cambiar la cantidad *a* usando todos los tipos de monedas excepto el primero, más
* el número de formas de cambiar la cantidad *a* - *d* usando todos *los n* tipos de monedas, donde *d* es la denominación del primer tipo de moneda.

Para ver por qué esto es cierto, observe que las formas de dar cambio se pueden dividir en dos grupos: las que no utilizan ninguna de las primeras monedas y las que sí las utilizan. Por lo tanto, el número total de formas de dar cambio por una determinada cantidad es igual al número de formas de dar cambio por esa cantidad sin utilizar ninguna de las primeras monedas, más el número de formas de dar cambio suponiendo que sí utilizamos las primeras. Pero este último número es igual al número de formas de dar cambio por la cantidad que queda después de utilizar una moneda de las primeras.

De este modo, podemos reducir recursivamente el problema de cambiar una cantidad dada al problema de cambiar cantidades más pequeñas utilizando menos tipos de monedas. Considere esta regla de reducción con atención y convénzase de que podemos usarla para describir un algoritmo si especificamos los siguientes casos degenerados: [33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_53)

* Si *a* es exactamente 0, deberíamos contarlo como 1 forma de realizar el cambio.
* Si *a* es menor que 0, deberíamos contarlo como 0 formas de realizar el cambio.
* Si *n* es 0, deberíamos contarlo como 0 formas de realizar el cambio.

Podemos traducir fácilmente esta descripción en un procedimiento recursivo:

(define (contar-cambiar importe)   
  (cc importe 5))   
(define (cc importe tipos-de-monedas)   
  (cond ((= importe 0) 1)   
        ((o (< importe 0) (= tipos-de-monedas 0)) 0)   
        (de lo contrario (+ (cc importe   
                     (- tipos-de-monedas 1))   
                 (cc (- importe   
                        (primera-denominación tipos-de-monedas))   
                     tipos-de-monedas)))))   
(define (primera-denominación tipos-de-monedas)   
  (cond ((= tipos-de-monedas 1) 1)   
        ((= tipos-de-monedas 2) 5)   
        ((= tipos-de-monedas 3) 10)   
        ((= tipos-de-monedas 4) 25)   
        ((= tipos-de-monedas 5) 50)))

(El procedimiento de primera denominación toma como entrada la cantidad de tipos de monedas disponibles y devuelve la denominación del primer tipo. Aquí estamos pensando en las monedas como si estuvieran ordenadas de mayor a menor, pero cualquier orden funcionaría también). Ahora podemos responder a nuestra pregunta original sobre el cambio de un dólar:

(cambio de cuenta 100)   
*292*

Count-change genera un proceso recursivo en forma de árbol con redundancias similares a las de nuestra primera implementación de fib . (Tomará bastante tiempo calcular ese 292). Por otro lado, no es obvio cómo diseñar un mejor algoritmo para calcular el resultado, y dejamos este problema como un desafío. La observación de que unEl proceso recursivo en árbol puede ser altamente ineficiente, pero a menudo fácil de especificar y comprender, lo que ha llevado a la gente a proponer que se podría obtener lo mejor de ambos mundos diseñando un "compilador inteligente" que pudiera transformar los procedimientos recursivos en árbol en procedimientos más eficientes que calculen el mismo resultado. [34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_54)

**Ejercicio 1.11.**   Una función *f* se define por la regla de que *f* ( *n* ) = *n* si *n* < 3 y *f* ( *n* ) = *f* ( *n* - 1) + 2 *f* ( *n* - 2) + 3 *f* ( *n* - 3) si *n* > 3. Escriba un procedimiento que calcule *f* mediante un proceso recursivo. Escriba un procedimiento que calcule *f* mediante un proceso iterativo.

**Ejercicio 1.12.**  El siguiente patrón de números se llama *triángulo de Pascal* .

![Forma

El contenido generado por IA puede ser incorrecto.]()

Los números en el borde del triángulo son todos 1, y cada número dentro del triángulo es la suma de los dos números que están encima de él. [35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_57) Escriba un procedimiento que calcule elementos del triángulo de Pascal mediante un proceso recursivo.

**Ejercicio 1.13.**   Demuestre que *F i b* ( *n* ) es el entero más próximo a *n* / 5, donde = (1 + 5)/2. Pista: Sea = (1 - 5)/2. Utilice la inducción y la definición de los números de Fibonacci (véase la sección  [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) ) para demostrar que *F i b* ( *n* ) = ( *n* - *n* )/ 5.

**[1.2.3 Órdenes de crecimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.2.3)**

Los ejemplos anteriores ilustran que los procesos pueden diferir considerablemente en las tasas a las que consumen recursos computacionales. Una forma conveniente de describir esta diferencia es utilizar la noción de*orden de crecimiento* para obtener una medida bruta de larecursos requeridos por un proceso a medida que las entradas se hacen mayores.

Sea *n* un parámetro que mide el tamaño del problema, y ​​sea *R* ( *n* ) la cantidad de recursos que el proceso requiere para un problema de tamaño *n* . En nuestros ejemplos anteriores tomamos *n* como el número para el cual se debe calcular una función dada, pero hay otras posibilidades. Por ejemplo, si nuestro objetivo es calcular una aproximación a la raíz cuadrada de un número, podemos tomar *n* como el número de dígitos de precisión requeridos. Para la multiplicación de matrices podemos tomar *n* como el número de filas en las matrices. En general, hay una serie de propiedades del problema con respecto a las cuales será deseable analizar un proceso dado. De manera similar, *R* ( *n* ) puede medir el número de registros de almacenamiento interno utilizados, el número de operaciones de máquina elementales realizadas, etc. En computadoras que solo realizan un número fijo de operaciones a la vez, el tiempo requerido será proporcional al número de operaciones de máquina elementales realizadas.

Decimos que *R* ( *n* ) tiene orden de crecimiento ( *f* ( *n* )), escrito *R* ( *n* ) = ( *f* ( *n* )) (pronunciado "theta de *f* ( *n* )"), si hay constantes positivas *k* 1 y *k* 2 independientes de *n* tales que



para cualquier valor suficientemente grande de *n* . (En otras palabras, para *n* grande , el valor *R* ( *n* ) está intercalado entre *k* 1 *f* ( *n* ) y *k* 2 *f* ( *n* ).)

Por ejemplo, con el proceso recursivo lineal para calcular el factorial descrito en la sección  [1.2.1,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.1) el número de pasos crece proporcionalmente a la entrada *n* . Por lo tanto, los pasos necesarios para este proceso crecen como ( *n* ). También vimos que el espacio requerido crece como ( *n* ). Para elfactorial iterativo, el número de pasos sigue siendo ( *n* ) pero el espacio es (1), es decir, constante. [36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_59)El cálculo recursivo de Fibonacci en árbol requiere ( *n* ) pasos y espacio ( *n* ), donde es la proporción áurea descrita en la sección  [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) .

Los órdenes de crecimiento proporcionan sólo una descripción burda del comportamiento de un proceso. Por ejemplo, un proceso que requiere *n* 2 pasos, un proceso que requiere 1000 *n* 2 pasos y un proceso que requiere 3 *n* 2 + 10 *n* + 17 pasos tienen todos un orden de crecimiento de ( *n* 2 ). Por otra parte, el orden de crecimiento proporciona una indicación útil de cómo podemos esperar que cambie el comportamiento del proceso a medida que cambiamos el tamaño del problema. Para un( *n* ) proceso (lineal), duplicar el tamaño duplicará aproximadamente la cantidad de recursos utilizados.En un proceso exponencial, cada incremento en el tamaño del problema multiplicará la utilización de recursos por un factor constante. En el resto de la sección  [1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2) examinaremos dos algoritmos cuyo orden de crecimiento eslogarítmico, de modo que duplicar el tamaño del problema aumenta el requerimiento de recursos en una cantidad constante.

**Ejercicio 1.14.**   Dibuje el árbol que ilustra el proceso generado por el procedimiento  de conteo y cambio de la sección [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) para dar cambio por 11 centavos. ¿Cuáles son los órdenes de crecimiento del espacio y el número de pasos utilizados por este proceso a medida que aumenta la cantidad a cambiar?

**Ejercicio 1.15.**  El seno de un ángulo (especificado en radianes) se puede calcular utilizando la aproximación sen *x* *x* si *x* es suficientemente pequeño y la identidad trigonométrica



para reducir el tamaño del argumento de sen . (Para los fines de este ejercicio, un ángulo se considera ``suficientemente pequeño'' si su magnitud no es mayor que 0,1 radianes). Estas ideas se incorporan en los siguientes procedimientos:

(define (cubo x) (\* x x x))   
(define (p x) (- (\* 3 x) (\* 4 (cubo x))))   
(define (ángulo seno)   
   (si (no (> (ángulo abs) 0,1))   
       ángulo   
       (p (seno (/ ángulo 3,0)))))

a. ¿Cuántas veces se aplica el procedimiento p cuando se evalúa (seno 12,15) ?

b. ¿Cuál es el orden de crecimiento en el espacio y el número de pasos (en función de  *a* ) utilizados por el proceso generado por el procedimiento seno cuando se evalúa (seno a) ?

**[1.2.4 Exponenciación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.2.4)**

Consideremos el problema de calcular la exponencial de un número dado. Nos gustaría un procedimiento que tome como argumentos una base *b* y un exponente entero positivo *n* y calcule *b n* . Una forma de hacerlo es mediante la definición recursiva



lo cual se traduce fácilmente en el procedimiento

(define (expt b n)   
  (si (= n 0)   
      1   
      (\* b (expt b (- n 1)))))

Se trata de un proceso recursivo lineal, que requiere ( *n* ) pasos y ( *n* ) espacio. Al igual que con el factorial, podemos formular fácilmente una iteración lineal equivalente:

(define (expt b n)   
  (expt-iter b n 1))   
  
(define (expt-iter b contador producto)   
  (si (= contador 0)   
      producto   
      (expt-iter b   
                (- contador 1)   
                (\* b producto))))

Esta versión requiere ( *n* ) pasos y (1) espacio.

Podemos calcular exponenciales en menos pasos utilizando cuadrados sucesivos. Por ejemplo, en lugar de calcular *b* 8 como



Podemos calcularlo usando tres multiplicaciones:

![Forma

El contenido generado por IA puede ser incorrecto.]()

Este método funciona bien para exponentes que son potencias de 2. También podemos aprovechar el cuadrado sucesivo al calcular exponenciales en general si usamos la regla



Podemos expresar este método como un procedimiento:

(define (exp-rápido b n)   
  (cond ((= n 0) 1)   
        ((¿par? n) (cuadrado (exp-rápido b (/ n 2))))   
        (sino (\* b (exp-rápido b (- n 1))))))

donde el predicado para probar si un entero es par se define en términos deprocedimiento primitivo resto por

(define (par? n)   
  (= (resto n 2) 0))

El proceso desarrollado por fast-expt crece logarítmicamente con *n* tanto en el espacio como en el número de pasos. Para comprobarlo, observe que calcular *b* 2 *n* utilizando fast-expt requiere solo una multiplicación más que calcular *b n* . Por lo tanto, el tamaño del exponente que podemos calcular se duplica (aproximadamente) con cada nueva multiplicación que se nos permite. Por lo tanto, el número de multiplicaciones necesarias para un exponente de *n* crece aproximadamente tan rápido como el logaritmo de *n* en base 2. El proceso tiene un crecimiento de ( log *n* ). [37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_62)

La diferencia entre el crecimiento ( log *n* ) y el crecimiento ( *n* ) se hace sorprendente a medida que *n* se hace grande. Por ejemplo, fast-expt para *n* = 1000 requiere solo 14 multiplicaciones. [38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_63) También es posible utilizar la idea de cuadratura sucesiva para idear un algoritmo iterativo que calcule exponenciales con un número logarítmico de pasos (ver ejercicio  [1.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.16) ), aunque, como suele suceder con los algoritmos iterativos, esto no se escribe de manera tan sencilla como el algoritmo recursivo. [39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_64)

**Ejercicio 1.16.**   Diseñe un procedimiento que desarrolle un proceso de exponenciación iterativa que utilice elevaciones sucesivas al cuadrado y utilice un número logarítmico de pasos, como lo hace fast-expt . (Sugerencia: utilizando la observación de que ( *b n* /2 ) 2 = ( *b* 2 ) *n* /2 , mantenga, junto con el exponente *n* y la base *b* , una variable de estado adicional *a* , y defina la transformación de estado de tal manera que el producto *a b n* no cambie de un estado a otro. Al comienzo del proceso, *a* se toma como 1, y la respuesta se da por el valor de *a* al final del proceso. En general, la técnica para definir una transformación de estado es*La cantidad invariable* que permanece sin cambios de un estado a otro es una forma poderosa de pensar en laDiseño de algoritmos iterativos.)

**Ejercicio 1.17.**   Los algoritmos de exponenciación de esta sección se basan en realizar la exponenciación mediante multiplicaciones repetidas. De manera similar, se puede realizar la multiplicación de números enteros mediante sumas repetidas. El siguiente procedimiento de multiplicación (en el que se supone que nuestro lenguaje solo puede sumar, no multiplicar) es análogo al procedimiento expt :

(define (\* a b)   
  (si (= b 0)   
      0   
      (+ a (\* a (- b 1)))))

Este algoritmo requiere una cantidad de pasos lineal en b . Ahora supongamos que incluimos, junto con la suma, las operaciones double , que duplica un entero, y halve , que divide un entero (par) por 2. Con estas operaciones, diseñe un procedimiento de multiplicación análogo a fast-expt que utilice una cantidad logarítmica de pasos.

**Ejercicio 1.18.**   Utilizando los resultados de los ejercicios  [1.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.16) y  [1.17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.17) , diseñe un procedimiento que genere un proceso iterativo para multiplicar dos números enteros en términos de suma, duplicación y división a la mitad y utilice un número logarítmico de pasos. [40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_68)

**Ejercicio 1.19.**   Existe un algoritmo inteligente para calcular los números de Fibonacci en un número logarítmico de pasos. Recordemos la transformación de las variables de estado *a* y *b* en el proceso  fib-iter de la sección [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) : *a* *a* + *b* y *b* *a* . Llamemos a esta transformación *T* , y observemos que al aplicar *T* una y otra vez *n* veces, comenzando con 1 y 0, se obtiene el par *F i b* ( *n* + 1) y *F i b* ( *n* ). En otras palabras, los números de Fibonacci se obtienen al aplicar *T n* , la *n* ésima potencia de la transformación *T* , comenzando con el par (1,0). Ahora consideremos que *T* es el caso especial de *p* = 0 y *q* = 1 en una familia de transformaciones *T p q* , donde *T p q* transforma el par ( *a* , *b* ) según *a* *b q* + *a q* + *a p* y *b* *b p* + *a q* . Demuestre que si aplicamos una transformación de este tipo *T p q* dos veces, el efecto es el mismo que si utilizamos una única transformación *T p* ' *q* ' de la misma forma, y ​​calculamos *p* ' y *q* ' en términos de *p* y  *q* . Esto nos proporciona una forma explícita de elevar al cuadrado estas transformaciones y, por lo tanto, podemos calcular *T n* utilizando elevaciones al cuadrado sucesivas, como en el procedimiento fast-expt . Juntemos todo esto para completar el siguiente procedimiento, que se ejecuta en un número logarítmico de pasos: [41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_70)

(define (fib n)   
  (fib-iter 1 0 0 1 n))   
(define (fib-iter a b p q count)   
  (cond ((= count 0) b)   
        ((even? count)   
         (fib-iter a   
                   b   
                   < *??* >       *; calculate  p '*  
                   < *??* >       *; calculate  q '*  
                   (/ count 2)))   
        (else (fib-iter (+ (\* b q) (\* a q) (\* a p))   
                        (+ (\* b p) (\* a q))   
                        p   
                        q   
                        (- count 1)))))

**[1.2.5 Máximos divisores comunes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.2.5)**

El máximo común divisor (MCD) de dos números enteros *a* y *b* se define como el mayor número entero que divide a *a* y *b* sin dejar resto. Por ejemplo, el MCD de 16 y 28 es 4. En el capítulo 2, cuando investiguemos cómo implementar la aritmética de números racionales, necesitaremos poder calcular MCD para reducir números racionales a su mínima expresión. (Para reducir un número racional a su mínima expresión, debemos dividir tanto el numerador como el denominador por su MCD. Por ejemplo, 16/28 se reduce a 4/7). Una forma de encontrar el MCD de dos números enteros es factorizarlos y buscar factores comunes, pero existe un algoritmo famoso que es mucho más eficiente.

La idea del algoritmo se basa en la observación de que, si *r* es el resto cuando *a* se divide por *b* , entonces los divisores comunes de *a* y *b* son exactamente los mismos que los divisores comunes de *b* y *r* . Por lo tanto, podemos utilizar la ecuación



reducir sucesivamente el problema de calcular un MCD al problema de calcular el MCD de pares de números enteros cada vez más pequeños. Por ejemplo,

Forma

El contenido generado por IA puede ser incorrecto.

reduce MCD(206,40) a MCD(2,0), que es 2. Es posible demostrar que si se parte de dos números enteros positivos cualesquiera y se realizan reducciones repetidas, siempre se obtendrá un par en el que el segundo número sea 0. En ese caso, el MCD es el otro número del par. Este método para calcular el MCD se conoce como *algoritmo de Euclides* . [42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_71)

Es fácil expresar el algoritmo de Euclides como un procedimiento:

(define (mcd a b)   
  (si (= b 0)   
      a   
      (mcd b (resto a b))))

Esto genera un proceso iterativo, cuyo número de pasos crece como el logaritmo de los números involucrados.

El hecho de que el número de pasos requeridos por el algoritmo de Euclides tenga un crecimiento logarítmico guarda una relación interesante con los números de Fibonacci:

**Teorema de Lamé:** si el algoritmo de Euclides requiere *k* pasos para calcular el MCD de algún par, entonces el número más pequeño del par debe ser mayor o igual al *k-* ésimo número de Fibonacci. [43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_72)

Podemos utilizar este teorema para obtener una estimación del orden de crecimiento del algoritmo de Euclides. Sea *n* la menor de las dos entradas del procedimiento. Si el proceso tiene *k* pasos, entonces debemos tener *n* > *F i b* ( *k* ) *k* / 5. Por lo tanto, el número de pasos *k* crece como el logaritmo (en base ) de *n* . Por lo tanto, el orden de crecimiento es ( log *n* ).

**Ejercicio 1.20.**  El proceso que genera un procedimiento depende, por supuesto, de las reglas que utiliza el intérprete. Como ejemplo, considere el procedimiento iterativo de mcd dado anteriormente. Supongamos que tuviéramos que interpretar este procedimiento utilizando la evaluación de orden normal, como se discutió en la sección  [1.1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.5) . (La regla de evaluación de orden normal para if se describe en el ejercicio  [1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.5) .) Utilizando el método de sustitución (para orden normal), ilustre el proceso generado en la evaluación de (mcd 206 40) e indique las operaciones de resto que se realizan realmente. ¿Cuántas operaciones de resto se realizan realmente en la evaluación de orden normal de (mcd 206 40) ? ¿En la evaluación de orden aplicativo?

**[1.2.6 Ejemplo: Prueba de primalidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.2.6)**

En esta sección se describen dos métodos para comprobar la primalidad de un entero *n* , uno con orden de crecimiento ( *n* ) y un algoritmo «probabilístico» con orden de crecimiento ( log *n* ). Los ejercicios que se incluyen al final de esta sección sugieren proyectos de programación basados ​​en estos algoritmos.

**[Buscando divisores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_74)**

Desde la antigüedad, los matemáticos se han sentido fascinados por los problemas relacionados con los números primos, y muchas personas han trabajado en el problema de determinar formas de comprobar si los números son primos. Una forma de comprobar si un número es primo es encontrar los divisores del número. El siguiente programa encuentra el divisor integral más pequeño (mayor que 1) de un número dado  *n* . Lo hace de una manera sencilla, comprobando la divisibilidad de *n* por enteros sucesivos comenzando con 2.

(definir (divisor más pequeño n)   
  (encontrar divisor n 2))  
(define (buscar-divisor n divisor-de-prueba)   
  (cond ((> (divisor-de-prueba-cuadrado) n) n)   
        ((divide? divisor-de-prueba n) divisor-de-prueba)   
        (else (buscar-divisor n (+ divisor-de-prueba 1)))))  
(define (divide? a b)   
  (= (resto b a) 0))

Podemos comprobar si un número es primo de la siguiente manera: *n* es primo si y sólo si *n* es su propio divisor más pequeño.

(define (¿primo? n)   
  (= n (divisor más pequeño n)))

La prueba final para encontrar divisor se basa en el hecho de que si *n* no es primo, debe tener un divisor menor o igual a *n* . [44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_75) Esto significa que el algoritmo solo necesita probar divisores entre 1 y *n* . En consecuencia, la cantidad de pasos necesarios para identificar *n* como primo tendrá un orden de crecimiento ( *n* ).

**[La prueba de Fermat](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_76)**

La prueba de primalidad ( log *n* ) se basa en un resultado de la teoría de números conocido como el Pequeño Teorema de Fermat. [45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_77)

**Pequeño teorema de Fermat:** si *n* es un número primo y *a* es cualquier entero positivo menor que *n* , entonces *a* elevado a la *n* -ésima potencia es congruente con *a* módulo *n* .

(Se dice que dos números son *congruentes módulo n* si ambos tienen el mismo resto cuando se dividen por *n* . El resto de un número *a* cuando se divide por *n* también se conoce como*resto de un módulo n* , o simplemente como *un módulo n* .)

Si *n* no es primo, entonces, en general, la mayoría de los números *a* < *n* no satisfarán la relación anterior. Esto conduce al siguiente algoritmo para probar la primalidad: Dado un número *n* , elija unnúmero aleatorio *a* < *n* y calculamos el resto de *a n* módulo *n* . Si el resultado no es igual a *a* , entonces *n* ciertamente no es primo. Si es *a* , entonces hay buenas probabilidades de que *n* sea primo. Ahora escojamos otro número aleatorio *a* y probémoslo con el mismo método. Si también satisface la ecuación, entonces podemos estar aún más seguros de que *n* es primo. Al probar más y más valores de *a* , podemos aumentar nuestra confianza en el resultado. Este algoritmo se conoce como la prueba de Fermat.

Para implementar la prueba de Fermat, necesitamos un procedimiento que calcule el exponencial de un número módulo otro número:

(define (expmod base exp m)   
  (cond ((= exp 0) 1)   
        ((¿par? exp)   
         (resto (cuadrado (expmod base (/ exp 2) m))   
                    m))   
        (sino   
         (resto (\* base (expmod base (- exp 1) m))   
                    m))))

Esto es muy similar al procedimiento de exponente rápido de la sección  [1.2.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.4) . Utiliza el cuadrado sucesivo, de modo que el número de pasos crece logarítmicamente con el exponente. [46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_78)

La prueba de Fermat se realiza eligiendo al azar un número *a* entre 1 y *n* - 1 inclusive y comprobando si el resto módulo *n* de la *n* -ésima potencia de *a* es igual a *a* . El número aleatorio *a* se elige mediante el procedimientorandom , que suponemos que está incluido como primitivo en Scheme. Random devuelve un entero no negativo menor que su entrada entera. Por lo tanto, para obtener un número aleatorio entre 1 y *n* - 1, llamamos a random con una entrada de *n* - 1 y sumamos 1 al resultado:

(define (prueba de fermat n)   
  (define (pruébalo a)   
    (= (expmod a n n) a))   
  (pruébalo (+ 1 (aleatorio (- n 1)))))

El siguiente procedimiento ejecuta la prueba una cantidad determinada de veces, según lo especificado por un parámetro. Su valor es verdadero si la prueba se ejecuta correctamente todas las veces y falso en caso contrario.

(define (¿prime rápido? n veces)   
  (cond ((= veces 0) verdadero)   
        ((prueba de fermat n) (¿prime rápido? n (- veces 1)))   
        (de lo contrario falso)))

**[Métodos probabilísticos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_79)**

La prueba de Fermat difiere en su carácter de la mayoría de los algoritmos conocidos, en los que se calcula una respuesta que se garantiza que es correcta. En este caso, la respuesta obtenida solo es probablemente correcta. Más precisamente, si *n* alguna vez falla la prueba de Fermat, podemos estar seguros de que *n* no es primo. Pero el hecho de que *n* pase la prueba, aunque sea una indicación extremadamente fuerte, no es una garantía de que *n* sea primo. Lo que nos gustaría decir es que para cualquier número *n* , si realizamos la prueba suficientes veces y descubrimos que *n* siempre pasa la prueba, entonces la probabilidad de error en nuestra prueba de primalidad puede hacerse tan pequeña como queramos.

Desafortunadamente, esta afirmación no es del todo correcta. Existen números que engañan a la prueba de Fermat: números *n* que no son primos y sin embargo tienen la propiedad de que *a n* es congruente con *a* módulo *n* para todos los enteros *a* < *n* . Tales números son extremadamente raros, por lo que la prueba de Fermat es bastante confiable en la práctica. [47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_80) Hay variaciones de la prueba de Fermat que no pueden ser engañadas. En estas pruebas, como con el método de Fermat, uno prueba la primalidad de un entero *n* eligiendo un entero aleatorio *a* < *n* y verificando alguna condición que depende de *n* y *a* . (Véase el ejercicio  [1.28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.28) para un ejemplo de dicha prueba.) Por otro lado, en contraste con la prueba de Fermat, uno puede probar que, para cualquier *n* , la condición no se cumple para la mayoría de los enteros *a* < *n* a menos que *n* sea primo. Así, si *n* pasa la prueba para alguna elección aleatoria de  *a* , las probabilidades son mejores que incluso que *n* sea primo. Si *n* pasa la prueba para dos elecciones aleatorias de *a , las probabilidades de que n* sea primo son mejores que 3 de 4. Al ejecutar la prueba con más y más valores de *a* elegidos aleatoriamente , podemos hacer que la probabilidad de error sea tan pequeña como queramos.

La existencia de pruebas que permiten demostrar que la probabilidad de error es arbitrariamente pequeña ha despertado el interés por los algoritmos de este tipo, que han pasado a denominarse *algoritmos probabilísticos* . Existe una gran actividad de investigación en este ámbito, y los algoritmos probabilísticos se han aplicado con éxito en muchos campos. [48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_81)

**Ejercicio 1.21.**   Utilice el procedimiento del divisor más pequeño para hallar el divisor más pequeño de cada uno de los siguientes números: 199, 1999, 19999.

**Ejercicio 1.22.**  La mayoría de las implementaciones de Lisp incluyen una primitiva llamada runtime que devuelve un entero que especifica la cantidad de tiempo que el sistema ha estado funcionando (medido, por ejemplo, en microsegundos). El siguiente procedimiento timed-prime-test , cuando se llama con un entero *n* , imprime *n* y verifica si *n* es primo. Si *n* es primo, el procedimiento imprime tres asteriscos seguidos de la cantidad de tiempo empleado en realizar la prueba.

(define (timed-prime-test n)   
  (newline)   
  (display n)   
  (start-prime-test n (runtime)))   
(define (start-prime-test n start-time)   
  (if (prime? n)   
      (report-prime (- (runtime) start-time))))   
(define (report-prime elapsed-time)   
  (display " \*\*\* ")   
  (display elapsed-time))

Utilizando este procedimiento, escriba un procedimiento de búsqueda de primos que verifique la primalidad de los números enteros impares consecutivos en un rango especificado. Utilice su procedimiento para encontrar los tres primos más pequeños mayores que 1000; mayores que 10 000; mayores que 100 000; mayores que 1 000 000. Observe el tiempo necesario para probar cada primo. Dado que el algoritmo de prueba tiene un orden de crecimiento de ( *n* ), debe esperar que la prueba de primos alrededor de 10 000 demore aproximadamente 10 veces más que la prueba de primos alrededor de 1000. ¿Sus datos de tiempo confirman esto? ¿Qué tan bien respaldan los datos para 100 000 y 1 000 000 la predicción *n* ? ¿Su resultado es compatible con la noción de que los programas en su máquina se ejecutan en un tiempo proporcional al número de pasos necesarios para el cálculo?

**Ejercicio 1.23.**  El procedimiento de divisor más pequeño que se muestra al principio de esta sección realiza muchas pruebas innecesarias: después de verificar si el número es divisible por 2, no tiene sentido verificar si es divisible por números pares mayores. Esto sugiere que los valores utilizados para test-divisor no deberían ser 2, 3, 4, 5, 6, ... , sino 2, 3, 5, 7, 9, ... . Para implementar este cambio, defina un procedimiento next que devuelva 3 si su entrada es igual a 2 y, de lo contrario, devuelva su entrada más 2. Modifique el procedimiento de divisor más pequeño para usar (next test-divisor) en lugar de (+ test-divisor 1) . Con timed-prime-test incorporando esta versión modificada de lowest-divisor , ejecute la prueba para cada uno de los 12 primos encontrados en el ejercicio  [1.22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.22) . Dado que esta modificación reduce a la mitad el número de pasos de prueba, debe esperar que se ejecute aproximadamente el doble de rápido. ¿Se confirma esta expectativa? Si no es así, ¿cuál es la relación observada entre las velocidades de los dos algoritmos y cómo explica el hecho de que sea diferente de 2?

**Ejercicio 1.24.**   Modifique el procedimiento de prueba de primos cronometrados del ejercicio  [1.22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.22) para utilizar el método de primos rápidos (el método de Fermat) y pruebe cada uno de los 12 primos que encontró en ese ejercicio. Dado que la prueba de Fermat tiene un crecimiento de ( log *n* ), ¿cómo esperaría que se compare el tiempo necesario para probar primos cercanos a 1.000.000 con el tiempo necesario para probar primos cercanos a 1000? ¿Sus datos confirman esto? ¿Puede explicar cualquier discrepancia que encuentre?

**Ejercicio 1.25.**   Alyssa P. Hacker se queja de que nos costó mucho trabajo extra escribir expmod . Después de todo, dice, como ya sabemos cómo calcular exponenciales, podríamos haber escrito simplemente

(define (expmod base exp m)   
  (resto (exp-rápido base exp) m))

¿Tiene razón? ¿Este procedimiento serviría también para nuestro probador de cebado rápido? Explíquelo.

**Ejercicio 1.26.**   Louis Reasoner tiene grandes dificultades para realizar el ejercicio  [1.24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.24) . Su prueba fast-prime? parece ejecutarse más lentamente que su prueba prime?. Louis llama a su amiga Eva Lu Ator para que lo ayude. Cuando examinan el código de Louis, descubren que ha reescrito el procedimiento expmod para utilizar una multiplicación explícita, en lugar de llamar a square :

(define (expmod base exp m)   
  (cond ((= exp 0) 1)   
        ((¿par? exp)   
         (resto (\* (expmod base (/ exp 2) m)   
                       (expmod base (/ exp 2) m))   
                    m))   
        (sino   
         (resto (\* base (expmod base (- exp 1) m))   
                    m))))

"No veo qué diferencia podría suponer eso", dice Louis. "Sí, la veo", dice Eva. "Al escribir el procedimiento de esa manera, has transformado el proceso ( log *n* ) en un proceso ( *n* )". Explícalo.

**Ejercicio 1.27.**  Demuestre que los números de Carmichael enumerados en la nota al pie  [47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#footnote_Temp_80) realmente engañan a la prueba de Fermat. Es decir, escriba un procedimiento que tome un entero *n* y pruebe si *a n* es congruente con *a* módulo *n* para cada *a* < *n* y pruebe su procedimiento con los números de Carmichael dados.

**Ejercicio 1.28.**  Una variante de la prueba de Fermat que no se puede engañar se llama *prueba de Miller-Rabin* (Miller 1976; Rabin 1980). Esta comienza desdeuna forma alternativa del Pequeño Teorema de Fermat, que establece que si *n* es un número primo y *a* es cualquier entero positivo menor que *n* , entonces *a* elevado a la ( *n* - 1)ª potencia es congruente con 1 módulo  *n* . Para probar la primalidad de un número *n* mediante la prueba de Miller-Rabin, escogemos un número aleatorio *a* < *n* y elevamos *a* a la ( *n* - 1)ª potencia módulo  *n* utilizando el procedimiento expmod . Sin embargo, siempre que realizamos el paso de elevación al cuadrado en expmod , verificamos si hemos descubierto una ``raíz cuadrada no trivial de 1 módulo  *n* '', es decir, un número distinto de 1 o *n* - 1 cuyo cuadrado sea igual a 1 módulo  *n* . Es posible probar que si existe una raíz cuadrada no trivial de 1, entonces *n* no es primo. También es posible demostrar que si *n* es un número impar que no es primo, entonces, para al menos la mitad de los números *a* < *n* , calcular *a n* -1 de esta manera revelará una raíz cuadrada no trivial de 1 módulo  *n* . (Esta es la razón por la que la prueba de Miller-Rabin no puede ser engañada.) Modifique el procedimiento expmod para que indique si descubre una raíz cuadrada no trivial de 1, y use esto para implementar la prueba de Miller-Rabin con un procedimiento análogo a fermat-test . Verifique su procedimiento probando varios primos y no primos conocidos. Sugerencia: Una forma conveniente de hacer que expmod indique es hacer que devuelva 0.

[En un programa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_46) real probablemente usaríamos la estructura de bloques introducida en la última sección para ocultar la definición defact-iter:

(define (factorial n)   
  (define (iter producto contador)   
    (si (> contador n)   
        producto   
        (iter (\* contador producto)   
              (+ contador 1))))   
  (iter 1 1))

Evitamos hacer esto aquí para minimizar la cantidad de cosas en las que pensar a la vez.

[30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_47) Cuando analizamos la implementación de procedimientos en máquinas de registro en el capítulo 5, veremos que cualquier proceso iterativo se puede realizar "en hardware" como una máquina que tiene un conjunto fijo de registros y ninguna memoria auxiliar. Por el contrario, la realización de un proceso recursivo requiere una máquina que utilice un conjunto fijo de registros.estructura de datos auxiliar conocida como *pila* .

[31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_48) La recursión de cola ha sido durante mucho tiempoconocido como un truco de optimización del compilador. Carl Hewitt (1977) proporcionó una base semántica coherente para la recursión de cola y la explicó entérminos del modelo computacional de "paso de mensajes" que discutiremos en el capítulo 3. Inspirados por esto, Gerald Jay Sussman y Guy Lewis Steele Jr. (ver Steele 1975) construyeron un intérprete recursivo de cola para Scheme. Steele luego mostró cómo la recursión de cola es una consecuencia de la forma natural de compilar llamadas a procedimientos (Steele 1977). El estándar IEEE para Scheme requiere que las implementaciones de Schemeser recursivo en la cola.

[32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_51) Un ejemplo de esto se sugirió en la sección [1.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.3): El intérprete mismo evalúa las expresiones usando un proceso recursivo de árbol.

[33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_53) Por ejemplo, calcule en detalle cómo se aplica la regla de reducción al problema de dar cambio de 10 centavos usando monedas de un centavo y de cinco centavos.

[34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_54) Un método para hacer frente a los cálculos redundantes es organizar las cosas de modo que construyamos automáticamente una tabla de valores a medida que se calculan. Cada vez que se nos pide que apliquemos el procedimiento a algún argumento, primero verificamos si el valor ya está almacenado en la tabla, en cuyo caso evitamos realizar el cálculo redundante. Esta estrategia, conocida como*La tabulación* o *memorización* se puede implementar de una manera sencilla. La tabulación se puede utilizar a veces para transformar procesos que requieren un número exponencial de pasos (como count-change ) en procesos cuyos requisitos de espacio y tiempo crecen linealmente con la entrada. Véase el ejercicio  [3.27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.27) .

[35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_57) Los elementos del triángulo de Pascal se llaman*coeficientes binomiales*, porque la*n*-ésima consta delos coeficientes de los términos en la expansión de ( *x* + *y* ) *n* . Este patrón para calcular los coeficientesapareció en la obra fundamental de Blaise Pascal de 1653 sobre la teoría de la probabilidad, *Tratado del triángulo aritmético* . SegúnKnuth (1973), el mismo patrón aparece en *Szu-yuen Yü-chien* (``El precioso espejo de los cuatro elementos''), publicadopor el matemático chino Chu Shih-chieh en 1303, en las obras del poeta y matemático persa del siglo XII Omar Khayyam, y en las obras del matemático hindú del siglo XII Bháscara Ã chárya.

[36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_59) Estas afirmaciones ocultan una gran cantidad de simplificaciones excesivas. Por ejemplo, si contamos los pasos de un proceso como «operaciones de máquina», estamos suponiendo que el número de operaciones de máquina necesarias para realizar, por ejemplo, una multiplicación es independiente del tamaño de los números que se van a multiplicar, lo que es falso si los números son suficientemente grandes. Observaciones similares son válidas para las estimaciones de espacio. Al igual que el diseño y la descripción de un proceso, el análisis de un proceso puede llevarse a cabo en varios niveles de abstracción.

[37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_62) Más precisamente, el número de multiplicaciones requeridas es igual a 1 menos que el logaritmo en base 2 de*n*más el número de unos en la representación binaria de*n*. Este total siempre es menor que el doble del logaritmo en base 2 de*n*. Las constantes arbitrarias*k*1y*k*2en la definición de notación de orden implican que, para un proceso logarítmico, la base a la que se llevan los logaritmos no importa, por lo que todos esos procesos se describen como(log *n*).

[Quizás te preguntes por](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_63) qué a alguien le interesaría elevar números a la milésima potencia. Consulta la sección [1.2.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.6).

[39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_64) Este algoritmo iterativo es antiguo. Aparece en el*Chandah-sutra*deÃ chárya Pingala, escrito antes del año 200 a . C. Véase Knuth 1981, sección 4.6.3, para un análisis y discusión completos de este y otros métodos de exponenciación.

[40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_68) EstoEl algoritmo, que a veces se conoce como el «método campesino ruso» de multiplicación, es antiguo. Se encuentran ejemplos de su uso enPapiro Rhind, uno de los dos documentos matemáticos más antiguos que existen, escrito alrededor de 1700 a . C. (y copiado de un manuscrito aún(documento más antiguo) por un escriba egipcio llamado Ah-mose.

[41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_70) Este ejercicio fueNos lo sugirió Joe Stoy, basándose en un ejemplo de Kaldewaij 1990.

[42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_71) El algoritmo de Euclides es asíSe llama así porque aparece en *los Elementos* de Euclides (libro 7, ca. 300 a . C. ). Según Knuth (1973), puede considerarse elEl algoritmo no trivial más antiguo que se conoce. El antiguo método egipcio de multiplicación (ejercicio  [1.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.18) ) es seguramente más antiguo, pero, como explica Knuth, el algoritmo de Euclides es el más antiguo que se conoce que se ha presentado como un algoritmo general, en lugar de como un conjunto de ejemplos ilustrativos.

[43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_72) Este teorema fue demostrado en 1845 por Gabriel Lamé, unMatemático e ingeniero francés conocido principalmente por sus contribuciones a la física matemática. Para demostrar el teorema, consideramos pares ( *a k* , *b k* ), donde *a k* > *b k* , para los cuales el algoritmo de Euclides termina en *k* pasos. La prueba se basa en la afirmación de que, si ( *a k* +1 , *b k* +1 ) ( *a k* , *b k* ) ( *a k* -1 , *b k* -1 ) son tres pares sucesivos en el proceso de reducción, entonces debemos tener *b k* +1 > *b k* + *b k* -1 . Para verificar la afirmación, considere que un paso de reducción se define aplicando la transformación *a k* -1 = *b k* , *b k* -1 = resto de *a k* dividido por *b k* . La segunda ecuación significa que *a k* = *q b k* + *b k* -1 para algún entero positivo *q* . Y como *q* debe ser al menos 1, tenemos *a k* = *q b k* + *b k* -1 > *b k* + *b k* -1 . Pero en el paso de reducción anterior tenemos *b k* +1 = *a k* . Por lo tanto, *b k* +1 = *a k* > *b k* + *b k* -1 . Esto verifica la afirmación. Ahora podemos demostrar el teorema por inducción sobre *k* , el número de pasos que el algoritmo requiere para terminar. El resultado es verdadero para *k* = 1, ya que esto simplemente requiere que *b* sea al menos tan grande como *F i b* (1) = 1. Ahora, supongamos que el resultado es verdadero para todos los números enteros menores o iguales a *k* y establezcamos el resultado para *k* + 1. Sea ( *a k* +1 , *b k* +1 ) ( *ak* , *b k* )( *a k* -1 , *b k* -1 ) sean pares sucesivos en el proceso de reducción. Por nuestras hipótesis de inducción, tenemos *b k* -1 > *F i b* ( *k* - 1) y *b k* > *F i b* ( *k* ). Por lo tanto, aplicando la afirmación que acabamos de demostrar junto con la definición de los números de Fibonacci obtenemos *b k* +1 > *b k* + *b k* -1 > *F i b* ( *k* ) + *F i b* ( *k* - 1) = *F i b* ( *k* + 1), lo que completa la demostración del teorema de Lamé.

[44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_75) Si*d*es divisor de*n*, entonces también lo es*n*/*d*. Pero*d*y*n*/*d*no pueden ser ambos mayores que*n*.

[45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_77) Pierre de Fermat (1601-1665) es considerado el fundador deTeoría de números moderna. Obtuvo muchos resultados importantes en teoría de números, pero generalmente sólo anunciaba los resultados, sin proporcionar pruebas.El pequeño teorema de Fermat fue enunciado en una carta que escribió en 1640. La primera prueba publicada fue dada porEuler en 1736 (y unAnteriormente, se había descubierto una prueba idéntica en los manuscritos inéditos de Leibniz. El resultado más famoso de Fermat, conocido como el Último Teorema de Fermat, fue anotado en 1637 en su copia del libro *Aritmética* (por el matemático griego del siglo III ).Diofanto) con la observación «He descubierto una prueba verdaderamente notable, pero este margen es demasiado pequeño para contenerla». Encontrar una prueba del último teorema de Fermat se convirtió en uno de los desafíos más famosos en la teoría de números.La solución finalmente fue dada en 1995 por Andrew Wiles de la Universidad de Princeton.

[46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_78) Los pasos de reducción en los casos en que el exponente*e*es mayor que 1 se basan en el hecho de que, para cualquier entero*x*,*y*y*m*, podemos hallar el resto de*x*por*y*módulo*m*calculando por separado los restos de*x*módulo*m*e y*módulo*m, multiplicándolos y luego tomando el resto del resultado módulo*m*. Por ejemplo, en el caso en que*e*es par, calculamos el resto de*b e* /2módulo*m,*lo elevamos al cuadrado y tomamos el resto módulo*m*. Esta técnica es útil porque significa que podemos realizar nuestro cálculo sin tener que lidiar nunca con números mucho mayores que*m*. (Compare el ejercicio [1.25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.25).)

[47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_80) Números que engañan al*Los números de Carmichael* son los que se obtienen con la prueba de Fermat , y se sabe poco sobre ellos, salvo que son extremadamente raros. Hay 255 números de Carmichael por debajo de 100.000.000. Los más pequeños son 561, 1105, 1729, 2465, 2821 y 6601. Al probar la primalidad de números muy grandes elegidos al azar, la probabilidad de encontrar un valor que engañe a la prueba de Fermat es menor que la probabilidad de queLa radiación cósmica hará que la computadora cometa un error al ejecutar un algoritmo "correcto". Considerar que un algoritmo es inadecuado por la primera razón pero no por la segunda ilustra la diferencia entreMatemáticas e ingeniería.

[48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_81) Una de las aplicaciones más llamativas deLa prueba de primos probabilísticos se ha aplicado al campo de la criptografía. Aunque ahora es computacionalmente inviable factorizar un número arbitrario de 200 dígitos, la primalidad de dicho número se puede comprobar en unos pocos segundos con la prueba de Fermat. Este hecho constituye la base de una técnica para construir "códigos indescifrables" sugerida porRemachado,Shamir, yAdleman (1977). El resultado*El algoritmo RSA* se ha convertido en una técnica ampliamente utilizada para mejorar la seguridad de las comunicaciones electrónicas. Debido a esto y a otros avances relacionados, el estudio deLos números primos, que alguna vez se consideraron el epítome de un tema de las matemáticas «puras» que debía estudiarse sólo por sí mismo, ahora resultan tener importantes aplicaciones prácticas para la criptografía, la transferencia electrónica de fondos y la recuperación de información.

[**1.3 Formulación de abstracciones con procedimientos de orden superior**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_1.3)

Hemos visto que los procedimientos son, en efecto, abstracciones que describen operaciones compuestas sobre números independientemente de los números particulares. Por ejemplo, cuando

(define (cubo x) (\* x x x))

No estamos hablando del cubo de un número en particular, sino de un método para obtener el cubo de cualquier número. Por supuesto, podríamos arreglárnoslas sin definir nunca este procedimiento, escribiendo siempre expresiones como

(\* 3 3 3)   
(\* x x x)   
(\* y y y)

y nunca mencionar explícitamente el cubo . Esto nos colocaría en una seria desventaja, obligándonos a trabajar siempre en el nivel de las operaciones particulares que resultan ser primitivas en el lenguaje (la multiplicación, en este caso) en lugar de en términos de operaciones de nivel superior. Nuestros programas serían capaces de calcular cubos, pero nuestro lenguaje carecería de la capacidad de expresar el concepto de cubos. Una de las cosas que deberíamos exigir de un lenguaje de programación potente es la capacidad de construir abstracciones asignando nombres a patrones comunes y luego trabajar en términos de las abstracciones directamente. Los procedimientos proporcionan esta capacidad. Es por eso que todos los lenguajes de programación, excepto los más primitivos, incluyen mecanismos para definir procedimientos.

Sin embargo, incluso en el procesamiento numérico, nuestra capacidad para crear abstracciones estará severamente limitada si nos limitamos a procedimientos cuyos parámetros deben ser números. A menudo, se utilizará el mismo patrón de programación con varios procedimientos diferentes. Para expresar dichos patrones como conceptos, necesitaremos construir procedimientos que puedan aceptar procedimientos como argumentos o devolver procedimientos como valores. Los procedimientos que manipulan procedimientos se denominan*procedimientos de orden superior* . Esta sección muestra cómo los procedimientos de orden superior pueden servir como poderosos mecanismos de abstracción, aumentando enormemente el poder expresivo de nuestro lenguaje.

**[1.3.1 Procedimientos como argumentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.3.1)**

Considere los tres procedimientos siguientes. El primero calcula la suma de los números enteros de a a b :

(define (suma de enteros a b)   
  (si (> a b)   
      0   
      (+ a (suma de enteros (+ a 1) b))))

El segundo calcula la suma de los cubos de los números enteros en el rango dado:

(define (suma de cubos a b)   
  (si (> a b)   
      0   
      (+ (cubo a) (suma de cubos (+ a 1) b))))

El tercero calcula la suma de una secuencia de términos en la serie.



que converge a /8 (muy lentamente): [49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_90)

(define (pi-suma a b)   
  (si (> a b)   
      0   
      (+ (/ 1.0 (\* a (+ a 2))) (pi-suma (+ a 4) b))))

Estos tres procedimientos comparten claramente un patrón subyacente común. En su mayor parte son idénticos y difieren únicamente en el nombre del procedimiento, la función de a que se utiliza para calcular el término que se debe sumar y la función que proporciona el siguiente valor de a . Podríamos generar cada uno de los procedimientos completando espacios en la misma plantilla:

(define (< *nombre* > a b)   
  (si (> a b)   
      0   
      (+ (< *término* > a)   
         (< *nombre* > (< *siguiente* > a) b))))

La presencia de un patrón tan común es una prueba contundente de que existe una abstracción útil que espera ser descubierta. De hecho, hace mucho tiempo que los matemáticos identificaron la abstracción de*suma de una serie* e inventó "sigma"notación'', por ejemplo



para expresar este concepto. El poder de la notación sigma es que permite a los matemáticos tratar con el concepto de suma en sí mismo en lugar de sólo con sumas particulares; por ejemplo, para formular resultados generales sobre sumas que son independientes de la serie particular que se está sumando.

De manera similar, como diseñadores de programas, nos gustaría que nuestro lenguaje fuera lo suficientemente potente como para poder escribir un procedimiento que exprese el concepto de suma en sí mismo, en lugar de solo procedimientos que calculen sumas particulares. Podemos hacerlo fácilmente en nuestro lenguaje procedimental tomando la plantilla común que se muestra arriba y transformando las "ranuras" en parámetros formales:

(define (término suma a siguiente b)   
  (si (> a b)   
      0   
      (+ (término a)   
         (término suma (siguiente a) siguiente b))))

Observe que suma toma como argumentos los límites inferior y superior a  y  b junto con los procedimientos term y next . Podemos usar suma como lo haríamos con cualquier procedimiento. Por ejemplo, podemos usarlo (junto con un procedimiento inc que incrementa su argumento en 1) para definir sumas de cubos :

(definir (inc n) (+ n 1))  
(define (suma de cubos a b)   
  (suma cubo a inc b))

Usando esto, podemos calcular la suma de los cubos de los números enteros del 1 al 10:

(suma de cubos 1 10)   
*3025*

Con la ayuda de un procedimiento de identidad para calcular el término, podemos definir sumas enteras en términos de suma :

(definir (identidad x) x)  
  
(define (suma de enteros a b)   
  (suma identidad a inc b))

Luego podemos sumar los números enteros del 1 al 10:

(suma de enteros 1 10)   
*55*

También podemos definir pi-sum de la misma manera: [50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_91)

(define (pi-suma a b)   
  (define (pi-término x)   
    (/ 1.0 (\* x (+ x 2))))   
  (define (pi-siguiente x)   
    (+ x 4))   
  (suma pi-término a pi-siguiente b))

Usando estos procedimientos, podemos calcular una aproximación a :

(\* 8 (pi-suma 1 1000))   
*3.139592655589783*

Una vez que tenemos la suma , podemos usarla como un bloque de construcción para formular conceptos adicionales. Por ejemplo, laLa integral definida de una función *f* entre los límites *a* y *b* se puede aproximar numéricamente utilizando la fórmula



Para valores pequeños de *d x* . Podemos expresar esto directamente como un procedimiento:

(define (integral f a b dx)   
  (define (sumar-dx x) (+ x dx))   
  (\* (suma f (+ a (/ dx 2.0)) suma-dx b)   
     dx))   
(integral cubo 0 1 0.01)   
*.24998750000000042*  
(integral cubo 0 1 0.001)   
*.249999875000001*

(El valor exacto de la integral del cubo entre 0 y 1 es 1/4.)

**Ejercicio 1.29.**  La regla de Simpson es un método de integración numérica más preciso que el método ilustrado anteriormente. Utilizando la regla de Simpson, la integral de una función *f* entre *a* y *b* se aproxima como



donde *h* = ( *b* - *a* )/ *n* , para algún entero par *n* , e *y k* = *f* ( *a* + *k h* ). (Aumentar *n* aumenta la precisión de la aproximación.) Defina un procedimiento que tome como argumentos *f* , *a* , *b* y *n* y devuelva el valor de la integral, calculada utilizando la Regla de Simpson. Utilice su procedimiento para integrar el cubo entre 0 y 1 (con *n* = 100 y *n* = 1000), y compare los resultados con los del procedimiento integral que se muestra arriba.

**Ejercicio 1.30.**  El procedimiento de suma anterior genera una recursión lineal. El procedimiento se puede reescribir de modo que la suma se realice de forma iterativa. Muestre cómo hacerlo completando las expresiones faltantes en la siguiente definición:

(define (término suma a siguiente b)   
  (define (iter a resultado)   
    (si < *??* >   
        < *??* >   
        (iter < *??* > < *??* >)))   
  (iter < *??* > < *??* >))

**Ejercicio 1.31.**     
a. El procedimiento suma es sólo el más simple de un gran número de abstracciones similares que pueden ser capturadas como procedimientos de orden superior. [51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_95) Escriba un procedimiento análogo llamado producto que devuelva el producto de los valores de una función en puntos sobre un rango dado. Muestre cómo definirfactorial en términos de producto . También use el producto para calcular aproximaciones autilizando la fórmula [52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_96)



b. Si el procedimiento de su producto genera un proceso recursivo, escriba uno que genere un proceso iterativo. Si genera un proceso iterativo, escriba uno que genere un proceso recursivo.

**Ejercicio 1.32.**  a. Demuestre que la suma y el producto (ejercicio  [1.31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.31) ) son ambos casos especiales de una noción aún más general llamada acumulación que combina una colección de términos, utilizando alguna función de acumulación general:

(acumula el término de valor nulo del combinador a en el siguiente b)

Accumulate toma como argumentos las mismas especificaciones de término y rango que sum y product , junto con un procedimiento combinador (de dos argumentos) que especifica cómo se combinará el término actual con la acumulación de los términos anteriores y un valor nulo que especifica qué valor base utilizar cuando se agoten los términos. Escriba recover y muestre cómo se pueden definir tanto sum como product como llamadas simples a recover .

b. Si su procedimiento de acumulación genera un proceso recursivo, escriba uno que genere un proceso iterativo. Si genera un proceso iterativo, escriba uno que genere un proceso recursivo.

**Ejercicio 1.33.**  Puede obtener una versión aún más general de acumular (ejercicio  [1.32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.32) ) introduciendo la noción de*Filtrar* los términos que se van a combinar. Es decir, combinar solo aquellos términos derivados de valores en el rango que satisfacen una condición especificada. La abstracción filtered-accumulate resultante toma los mismos argumentos que recover, junto con un predicado adicional de un argumento que especifica el filtro. Escriba filtered-accumulate como un procedimiento. Muestre cómo expresar lo siguiente utilizando filtered-accumulate :

a. la suma de los cuadrados de los números primos en el intervalo de *a* a *b* (suponiendo que ya tienes un predicado primo escrito)

b. el producto de todos los números enteros positivos menores que *n* que son relativamente primos con respecto a  *n* (es decir, todos los números enteros positivos *i* < *n* tales que *G C D* ( *i* , *n* ) = 1).

**[1.3.2 Construcción de procedimientos mediante Lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.3.2)**

Al utilizar suma como en la sección  [1.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.1) , parece terriblemente extraño tener que definir procedimientos triviales como pi-term y pi-next solo para poder usarlos como argumentos para nuestro procedimiento de orden superior. En lugar de definir pi-next y pi-term , sería más conveniente tener una forma de especificar directamente ``el procedimiento que devuelve su entrada incrementada en 4'' y ``el procedimiento que devuelve el recíproco de su entrada multiplicada por su entrada más 2''. Podemos hacer esto introduciendo la forma especial lambda , que crea procedimientos. Usando lambda podemos describir lo que queremos como

(lambda(x) (+ x 4))

y

(lambda(x)(/1.0(\*x(+x2))))

Entonces nuestro procedimiento pi-sum se puede expresar sin definir ningún procedimiento auxiliar como

(define (pi-suma a b)   
  (suma (lambda (x) (/ 1.0 (\* x (+ x 2))))   
       a   
       (lambda (x) (+ x 4))   
       b))

Usando nuevamente lambda , podemos escribir el procedimiento integral sin tener que definir el procedimiento auxiliar add-dx :

(define (integral f a b dx)   
  (\* (suma f   
          (+ a (/ dx 2.0))   
          (lambda (x) (+ x dx))   
          b)   
     dx))

En general, lambda se utiliza para crear procedimientos de la misma manera que define , excepto queNo se especifica ningún nombre para el procedimiento:

(lambda (< *parámetros formales* >) < *cuerpo* >)

El procedimiento resultante es un procedimiento igual que uno creado usando define . La única diferencia es que no se ha asociado con ningún nombre en el entorno. De hecho,

(define (más4 x) (+ x 4))

es equivalente a

(define plus4(lambda(x) (+ x 4)))

Podemos leer una expresión lambda de la siguiente manera:

    (lambda (x) (+ x 4))   
                                                  
 el procedimiento de un argumento  x   que suma   x  y 4

Al igual que cualquier expresión que tiene un procedimiento como valor, una expresión lambda se puede utilizar como operador en una combinación como

((lambda (x y z) (+ x y (cuadrado z))) 1 2 3)   
*12*

o, de manera más general, en cualquier contexto en el que normalmente utilizaríamos un nombre de procedimiento. [53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_99)

**[Uso de let para crear variables locales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_100)**

Otro uso de lambda es la creación de variables locales. A menudo necesitamos variables locales en nuestros procedimientos además de las que se han vinculado como parámetros formales. Por ejemplo, supongamos que deseamos calcular la función



Lo cual también podríamos expresar como

Forma

El contenido generado por IA puede ser incorrecto.

Al escribir un procedimiento para calcular *f* , nos gustaría incluir como variables locales no solo *x e y* sino también los nombres de cantidades intermedias como *a* y *b* . Una forma de lograr esto es usar un procedimiento auxiliar para vincular las variables locales:

(define (f x y)   
  (define (f-auxiliar a b)   
    (+ (\* x (cuadrado a))   
       (\* y b)   
       (\* a b)))   
  (f-auxiliar (+ 1 (\* x y))    
            (- 1 y)))

Por supuesto, podríamos usar una expresión lambda para especificar un procedimiento anónimo para vincular nuestras variables locales. El cuerpo de f se convierte entonces en una única llamada a ese procedimiento:

(define (f x y)   
  ((lambda (a b)   
     (+ (\* x (cuadrado a))   
        (\* y b)   
        (\* a b)))   
   (+ 1 (\* x y))   
   (- 1 y)))

Esta construcción es tan útil que existe una forma especial llamada let para que su uso sea más conveniente. Usando let , el procedimiento f podría escribirse como

(define (f x y)   
  (sea ((a (+ 1 (\* x y)))   
        (b (- 1 y)))   
    (+ (\* x (cuadrado a))   
       (\* y b)   
       (\* a b))))

La forma general de una expresión let es

(let ((< *var 1* > < *exp 1* >)   
      (< *var 2* > < *exp 2* >)   
        
      (< *var n* > < *exp n* >))   
   < *cuerpo* >)

Lo cual puede considerarse como decir

|  |  |
| --- | --- |
| dejar | < *var 1* > tiene el valor < *exp 1* > y |
|  | < *var 2* > tiene el valor < *exp 2* > y |
|  |  |
|  | < *var n* > tiene el valor < *exp n* > |
| en | < *cuerpo* > |

La primera parte de la expresión let es una lista de pares nombre-expresión. Cuando se evalúa let , cada nombre se asocia con el valor de la expresión correspondiente. El cuerpo de let se evalúa con estos nombres enlazados como variables locales. La forma en que esto sucede es que la expresión let se interpreta como una sintaxis alternativa para

((lambda (< *var 1* >  ... < *var n* >)   
    < *cuerpo* >)   
 < *exp 1* >   
   
 < *exp n* >)

No se requiere ningún mecanismo nuevo en el intérprete para proporcionar variables locales.La expresión let es simplemente azúcar sintáctica para la aplicación lambda subyacente .

De esta equivalencia podemos ver que el alcance de una variable especificada por una expresión let es el cuerpo de la expresión let . Esto implica que:

* Let permite vincular variables de la forma más local posible al lugar donde se van a utilizar. Por ejemplo, si el valor de x es 5, el valor de la expresión

(+ (sea ((x 3))   
     (+ x (\* x 10)))   
   x)

es 38. Aquí, la x en el cuerpo del let es 3, por lo que el valor de la expresión let es 33. Por otro lado, la x que es el segundo argumento del + más externo sigue siendo 5.

* Los valores de las variables se calculan fuera de let . Esto es importante cuando las expresiones que proporcionan los valores para las variables locales dependen de variables que tienen los mismos nombres que las propias variables locales. Por ejemplo, si el valor de x es 2, la expresión

(sea ((x 3)   
      (y (+ x 2)))   
  (\* x y))

tendrá el valor 12 porque, dentro del cuerpo del let , x será 3 e y será 4 (que es la x externa más 2).

A veces podemos usar definiciones internas para obtener el mismo efecto que con let . Por ejemplo, podríamos haber definido el procedimiento f anterior como

(define (f x y)   
  (define a (+ 1 (\* x y)))   
  (define b (- 1 y))   
  (+ (\* x (cuadrado a))   
     (\* y b)   
     (\* a b)))

Sin embargo, preferimos utilizar let en situaciones como ésta y utilizar internal define sólo para procedimientos internos. [54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_101)

**Ejercicio 1.34.**   Supongamos que definimos el procedimiento

(definir (f g)   
  (g 2))

Entonces tenemos

(f cuadrado)   
*4*  
  
(f (lambda (z) (\* z (+ z 1))))   
*6*

¿Qué sucede si (perversamente) le pedimos al intérprete que evalúe la combinación (ff) ? Explíquelo.

**[1.3.3 Procedimientos como métodos generales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.3.3)**

[En la sección 1.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_1.1.4) introdujimos los procedimientos compuestos  como un mecanismo para abstraer patrones de operaciones numéricas de modo de hacerlos independientes de los números particulares involucrados. Con los procedimientos de orden superior, como el procedimiento integral de la sección  [1.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.1) , comenzamos a ver un tipo de abstracción más potente: los procedimientos utilizados para expresar métodos generales de cálculo, independientes de las funciones particulares involucradas. En esta sección analizamos dos ejemplos más elaborados (métodos generales para hallar ceros y puntos fijos de funciones) y mostramos cómo estos métodos pueden expresarse directamente como procedimientos.

**[Encontrar raíces de ecuaciones mediante el método del semiintervalo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_103)**

El *método de semiintervalo* es una técnica sencilla pero potente para hallar raíces de una ecuación *f* ( *x* ) = 0, donde *f* es una función continua. La idea es que, si se nos dan los puntos *a* y *b* tales que *f* ( *a* ) < 0 < *f* ( *b* ), entonces *f* debe tener al menos un cero entre *a* y *b* . Para localizar un cero, sea *x* el promedio de *a* y *b* y calcule *f* ( *x* ). Si *f* ( *x* ) > 0, entonces *f* debe tener un cero entre *a* y *x* . Si *f* ( *x* ) < 0, entonces *f* debe tener un cero entre *x* y *b* . Continuando de esta manera, podemos identificar intervalos cada vez más pequeños en los que *f* debe tener un cero. Cuando llegamos a un punto en el que el intervalo es lo suficientemente pequeño, el proceso se detiene. Como el intervalo de incertidumbre se reduce a la mitad en cada paso del proceso, el número de pasos necesarios crece como ( log ( *L* / *T* )), donde *L* es la longitud del intervalo original y *T* es la tolerancia de error (es decir, el tamaño del intervalo que consideraremos "suficientemente pequeño"). A continuación se presenta un procedimiento que implementa esta estrategia:

(define (busca f punto-neg punto-pos)   
  (let ((punto-medio (promedio punto-neg punto-pos)))   
    (si (¿suficientemente-cerca? punto-neg punto-pos)   
        punto-medio   
        (let ((valor-prueba (f punto-medio)))   
          (cond ((¿positivo? valor-prueba)   
                 (busca f punto-neg punto-medio))   
                ((¿negativo? valor-prueba)   
                 (busca f punto-medio punto-pos))   
                (sino punto-medio))))))

Suponemos que inicialmente se nos da la función *f* junto con puntos en los que sus valores son negativos y positivos. Primero calculamos el punto medio de los dos puntos dados. A continuación, comprobamos si el intervalo dado es lo suficientemente pequeño y, si es así, simplemente devolvemos el punto medio como nuestra respuesta. De lo contrario, calculamos como valor de prueba el valor de *f* en el punto medio. Si el valor de prueba es positivo, entonces continuamos el proceso con un nuevo intervalo que va desde el punto negativo original hasta el punto medio. Si el valor de prueba es negativo, continuamos con el intervalo desde el punto medio hasta el punto positivo. Finalmente, existe la posibilidad de que el valor de prueba sea 0, en cuyo caso el punto medio es en sí mismo la raíz que estamos buscando.

Para comprobar si los puntos finales están "suficientemente cerca", podemos utilizar un procedimiento similar al utilizado en la sección  [1.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.7) para calcular raíces cuadradas: [55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_104)

(definir (¿suficientemente cerca? x y)   
  (< (abs (- x y)) 0,001))

La búsqueda es difícil de usar directamente, porque podemos asignarle accidentalmente puntos en los que los valores de *f* no tengan el signo requerido, en cuyo caso obtendremos una respuesta incorrecta. En su lugar, utilizaremos la búsqueda mediante el siguiente procedimiento, que verifica cuál de los puntos finales tiene un valor de función negativo y cuál tiene un valor positivo, y llama al procedimiento de búsqueda en consecuencia. Si la función tiene el mismo signo en los dos puntos dados, no se puede utilizar el método de medio intervalo, en cuyo caso el procedimiento señala un error. [56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_105)

(define (método de medio intervalo f a b)   
  (let ((valor a (f a))   
        (valor b (f b)))   
    (cond ((y (¿valor a negativo?) (¿valor b positivo?))   
           (busca f a b))   
          ((y (¿valor b negativo?) (¿valor a positivo?))   
           (busca f b a))   
          (else   
           (error "Los valores no son de signo opuesto" a b)))))

El siguiente ejemplo utiliza el método del medio intervalo para aproximar como la raíz entre 2 y 4 de sen *x* = 0:

(método del medio intervalo sen 2,0 4,0)   
*3,14111328125*

He aquí otro ejemplo, utilizando el método del medio intervalo para buscar una raíz de la ecuación *x* 3 - 2 *x* - 3 = 0 entre 1 y 2:

(método del medio intervalo (lambda (x) (- (\* x x x) (\* 2 x) 3))   
                      1.0   
                      2.0)   
*1.89306640625*

**[Encontrar puntos fijos de funciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_106)**

Un número *x* se denomina *punto fijo* de una función *f* si *x* satisface la ecuación *f* ( *x* ) = *x* . Para algunas funciones *f* podemos localizar un punto fijo comenzando con una estimación inicial y aplicando *f* repetidamente,



hasta que el valor no cambie mucho. Con esta idea, podemos idear un procedimiento de punto fijo que toma como entradas una función y una estimación inicial y produce una aproximación a un punto fijo de la función. Aplicamos la función repetidamente hasta que encontramos dos valores sucesivos cuya diferencia sea menor que una tolerancia prescrita:

(definir tolerancia 0,00001)  
(define (punto fijo f primera suposición)   
  (define (¿suficientemente cerca? v1 v2)   
    (< (abs (- v1 v2)) tolerancia))   
  (define (intenta adivinar)   
    (let ((next (f suposición)))   
      (if (¿suficientemente cerca? adivinar next)   
          next   
          (intenta siguiente))))   
  (intenta primera suposición))

Por ejemplo, podemos utilizar este método para aproximar el punto fijo de la función coseno, comenzando con 1 como aproximación inicial: [57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_107)

(coseno de punto fijo 1,0)   
*.7390822985224023*

De manera similar, podemos encontrar una solución para la ecuación *y* = sen *y* + cos *y* :

(punto fijo (lambda (y) (+ (sin y) (cos y)))   
             1,0)   
*1,2587315962971173*

El proceso de punto fijo recuerda al proceso que utilizamos para hallar raíces cuadradas en la sección  [1.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.7) . Ambos se basan en la idea de mejorar repetidamente una suposición hasta que el resultado satisfaga algún criterio. De hecho, podemos formular fácilmente elCálculo de la raíz cuadrada como una búsqueda de punto fijo. Calcular la raíz cuadrada de un número *x* requiere encontrar un *y* tal que *y* 2 = *x* . Al poner esta ecuación en la forma equivalente *y* = *x* / *y* , reconocemos que estamos buscando un punto fijo de la función [58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_108) *y* *x* / *y* , y por lo tanto podemos intentar calcular raíces cuadradas como

(define (sqrt x)   
  (punto fijo (lambda (y) (/ x y))   
               1.0))

Lamentablemente, esta búsqueda de punto fijo no converge. Consideremos una suposición inicial *y* 1 . La siguiente suposición es *y* 2 = *x* / *y* 1 y la siguiente suposición es *y* 3 = *x* / *y* 2 = *x* /( *x* / *y* 1 ) = *y* 1 . Esto da como resultado un bucle infinito en el que las dos suposiciones *y* 1 e *y* 2 se repiten una y otra vez, oscilando en torno a la respuesta.

Una forma de controlar tales oscilaciones es evitar que las suposiciones cambien demasiado. Dado que la respuesta siempre está entre nuestra suposición *y* y *x* / *y* , podemos hacer una nueva suposición que no esté tan lejos de *y* como *x* / *y* promediando *y* con *x* / *y* , de modo que la siguiente suposición después de *y* sea (1/2)( *y* + *x* / *y* ) en lugar de *x* / *y* . El proceso de hacer tal secuencia de suposiciones es simplemente el proceso de buscar un punto fijo de *y* (1/2)( *y* + *x* / *y* ):

(define (sqrt x)   
  (punto fijo (lambda (y) (promedio y (/ x y)))   
               1.0))

(Tenga en cuenta que *y* = (1/2)( *y* + *x* / *y* ) es una transformación simple de la ecuación *y* = *x* / *y* ; para derivarla, agregue *y* a ambos lados de la ecuación y divida por 2.)

Con esta modificación, el procedimiento de la raíz cuadrada funciona. De hecho, si desentrañamos las definiciones, podemos ver que la secuencia de aproximaciones a la raíz cuadrada generada aquí es exactamente la misma que la generada por nuestro procedimiento de raíz cuadrada original de la sección  [1.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.7) . Este enfoque de promediar aproximaciones sucesivas a una solución, una técnica que llamamos*La amortiguación promedio* a menudo ayuda a la convergencia de búsquedas de puntos fijos.

**Ejercicio 1.35.**  Demuestre que la proporción áurea (sección  [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) ) es un punto fijo de la transformación *x* 1 + 1/ *x* , y utilice este hecho para calcular mediante el procedimiento de punto fijo .

**Ejercicio 1.36.**   Modifique fixed-point para que imprima la secuencia de aproximaciones que genera, utilizando las primitivas  de nueva línea y visualización que se muestran en el ejercicio [1.22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.22) . Luego, encuentre una solución para *x x* = 1000 al encontrar un punto fijo de *x* log (1000)/ log ( *x* ). (Use la función de Scheme)procedimiento de logaritmo primitivo , que calcula logaritmos naturales.) Compare la cantidad de pasos que esto requiere con y sin amortiguamiento promedio. (Tenga en cuenta que no puede comenzar en punto fijo con un valor de 1, ya que esto causaría una división por log (1) = 0.)

**Ejercicio 1.37.**  a. Una *fracción continua* infinita es una expresión de la forma

Forma

El contenido generado por IA puede ser incorrecto.

Como ejemplo, se puede demostrar que la expansión de fracción continua infinita con *N i* y *D i* todos iguales a 1 produce 1/ , donde es la proporción áurea (descrita en la sección  [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) ). Una forma de aproximar una fracción continua infinita es truncar la expansión después de un número dado de términos. Tal truncamiento -una denominada *fracción continua finita de k* términos- tiene la forma

![Forma

El contenido generado por IA puede ser incorrecto.]()

Supongamos que n y d son procedimientos de un argumento (el índice del término *i* ) que devuelven *N i* y *D i* de los términos de la fracción continua. Defina un procedimiento cont-frac tal que al evaluar (cont-frac ndk) se calcule el valor de la fracción continua finita *de k* términos. Verifique su procedimiento aproximando 1/ usando

(cont-frac (lambda (i) 1.0)   
           (lambda (i) 1.0)   
           k)

Para valores sucesivos de k . ¿Qué tan grande debe ser el valor de k para obtener una aproximación que sea precisa hasta 4 decimales?

b. Si su procedimiento cont-frac genera un proceso recursivo, escriba uno que genere un proceso iterativo. Si genera un proceso iterativo, escriba uno que genere un proceso recursivo.

**Ejercicio 1.38.**  En 1737, el matemático suizo Leonhard Euler publicó unas memorias, *De Fractionibus Continuis* , que incluían unaExpansión de fracción continua para *e* - 2, donde *e* es la base de los logaritmos naturales. En esta fracción, los *N i* son todos 1, y los *D i* son sucesivamente 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, ... . Escriba un programa que utilice el procedimiento cont-frac del ejercicio  [1.37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.37) para aproximar *e* , basándose en la expansión de Euler.

**Ejercicio 1.39.**  Una representación de fracción continua de la función tangente fue publicada en 1770 por el matemático alemán JH Lambert:

![Forma

El contenido generado por IA puede ser incorrecto.]()

donde *x* está en radianes. Defina un procedimiento (tan-cf xk) que calcule una aproximación a la función tangente según la fórmula de Lambert. K especifica el número de términos a calcular, como en el ejercicio  [1.37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.37) .

**[1.3.4 Procedimientos como valores devueltos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_1.3.4)**

Los ejemplos anteriores demuestran cómo la capacidad de pasar procedimientos como argumentos mejora significativamente el poder expresivo de nuestro lenguaje de programación. Podemos lograr un poder expresivo aún mayor creando procedimientos cuyos valores devueltos sean en sí mismos procedimientos.

Podemos ilustrar esta idea mirando nuevamente el ejemplo de punto fijo descrito al final de la sección  [1.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.3) . Formulamos una nueva versión del procedimiento de raíz cuadrada como una búsqueda de punto fijo, comenzando con la observación de que *x* es un punto fijo de la función *y* *x* / *y* . Luego usamos amortiguamiento promedio para hacer que las aproximaciones converjan. El amortiguamiento promedio es una técnica general útil en sí misma. Es decir, dada una función  *f* , consideramos la función cuyo valor en *x* es igual al promedio de *x* y *f* ( *x* ).

Podemos expresar la idea de amortiguamiento promedio mediante el siguiente procedimiento:

(define (promedio-húmedo f)   
  (lambda (x) (promedio x (f x))))

Average-damp es un procedimiento que toma como argumento un procedimiento f y devuelve como valor un procedimiento (producido por la lambda ) que, cuando se aplica a un número x , produce el promedio de x y (fx) . Por ejemplo, aplicar Average-damp al procedimiento cuadrado produce un procedimiento cuyo valor en algún número *x* es el promedio de *x* y *x* 2. Aplicar este procedimiento resultante a 10 devuelve el promedio de 10 y 100, o 55: [59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_114)

((cuadrado húmedo promedio) 10)   
*55*

Usando promedio-humedad , podemos reformular el procedimiento de raíz cuadrada de la siguiente manera:

(define (sqrt x)   
  (punto fijo (promedio-amortiguado (lambda (y) (/ x y)))   
               1.0))

Observe cómo esta formulación hace explícitas las tres ideas del método: búsqueda de punto fijo, amortiguamiento promedio y la función *y* *x* / *y* . Es instructivo comparar esta formulación del método de la raíz cuadrada con la versión original dada en la sección  [1.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.7) . Tenga en cuenta que estos procedimientos expresan el mismo proceso y observe cuánto más clara se vuelve la idea cuando expresamos el proceso en términos de estas abstracciones. En general, hay muchas formas de formular un proceso como un procedimiento. Los programadores experimentados saben cómo elegir formulaciones procedimentales que son particularmente perspicuas y donde los elementos útiles del proceso se exponen como entidades separadas que pueden reutilizarse en otras aplicaciones. Como un ejemplo simple de reutilización, observe que la raíz cúbica de *x* es un punto fijo de la función *y* *x* / *y* 2 , por lo que podemos generalizar inmediatamente nuestro procedimiento de raíz cuadrada a uno que extraeraíces cúbicas: [60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_115)

(define (raíz cúbica x)   
  (punto fijo (promedio húmedo (lambda (y) (/ x (y cuadrado))))   
               1.0))

**[El método de Newton](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_116)**

Cuando presentamos por primera vez el procedimiento de raíz cuadrada, en la sección  [1.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.7) , mencionamos que este era un caso especial del *método de Newton* . Si *x* *g* ( *x* ) es una función diferenciable, entonces una solución de la ecuación *g* ( *x* ) = 0 es un punto fijo de la función *x* *f* ( *x* ) donde



y *D g* ( *x* ) es la derivada de *g* evaluada en *x* .El método de Newton es el uso del método de punto fijo que vimos anteriormente para aproximar una solución de la ecuación hallando un punto fijo de la función *f* . [61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_117) Para muchas funciones *g* y para conjeturas iniciales suficientemente buenas para *x* , el método de Newton converge muy rápidamente a una solución de *g* ( *x* ) = 0. [62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_118)

Para implementar el método de Newton como procedimiento, primero debemos expresar la idea de derivada. Nótese que la "derivada", al igual que el amortiguamiento promedio, es algo que transforma una función en otra función. Por ejemplo, la derivada de la función *x* *x* 3 es la función *x* 3 *x* 2 . En general, si *g* es una función y *d x* es un número pequeño, entonces la derivada *D g* de *g* es la función cuyo valor en cualquier número *x está dado (en el límite de d x* pequeño ) por



Por lo tanto, podemos expresar la idea de derivada (tomando *d x* como, digamos, 0,00001) como el procedimiento

(define (deriva g)   
  (lambda (x)   
    (/ (- (g (+ x dx)) (g x))   
       dx)))

junto con la definición

(definir dx 0,00001)

Al igual que Average-damp , deriv es un procedimiento que toma un procedimiento como argumento y devuelve un procedimiento como valor. Por ejemplo, para aproximar la derivada de *x* *x* 3 en 5 (cuyo valor exacto es 75) podemos evaluar

(define (cubo x) (\* x x x))   
((deriva cubo) 5)   
*75.00014999664018*

Con la ayuda de deriv , podemos expresar el método de Newton como un proceso de punto fijo:

(define (transformada de newton g)   
  (lambda (x)   
    (- x (/ (g x) ((deriva g) x)))))  
(definir (método de newton g)   
  (punto fijo (transformada de newton g))

El procedimiento de la transformada de Newton expresa la fórmula que se menciona al principio de esta sección, y el método de Newton se define fácilmente en términos de ella. Toma como argumentos un procedimiento que calcula la función para la que queremos encontrar un cero, junto con una estimación inicial. Por ejemplo, para encontrar elraíz cuadrada de *x* , podemos usar el método de Newton para encontrar un cero de la función *y* *y* 2 - *x* comenzando con una estimación inicial de 1. [63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_119) Esto proporciona otra forma del procedimiento de raíz cuadrada:

(define (raíz cuadrada de x)   
  (método de newton (lambda (y) (- (y al cuadrado) x))   
                  1.0))

**[Abstracciones y procedimientos de primera clase](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_120)**

Hemos visto dos formas de expresar el cálculo de la raíz cuadrada como una instancia de un método más general, una como una búsqueda de punto fijo y otra utilizando el método de Newton. Dado que el método de Newton se expresó como un proceso de punto fijo, en realidad vimos dos formas de calcular raíces cuadradas como puntos fijos. Cada método comienza con una función y encuentra unaPunto fijo de alguna transformación de la función. Podemos expresar esta idea general como un procedimiento:

(definir (aproximación de transformación g de punto fijo)   
  (aproximación de transformación g de punto fijo))

Este procedimiento muy general toma como argumentos un procedimiento g que calcula una función, un procedimiento que transforma g y una estimación inicial. El resultado devuelto es un punto fijo de la función transformada.

Usando esta abstracción, podemos reformular el primer cálculo de raíz cuadrada de esta sección (donde buscamos un punto fijo de la versión promedio amortiguada de *y* *x* / *y* ) como una instancia de este método general:

(define (sqrt x)   
  (punto fijo de transformación (lambda (y) (/ x y))   
                            promedio húmedo   
                            1.0))

De manera similar, podemos expresar el segundo cálculo de raíz cuadrada de esta sección (una instancia del método de Newton que encuentra un punto fijo de la transformada de Newton de *y* *y* 2 - *x* ) como

(define (sqrt x)   
  (transformada de punto fijo (lambda (y) (- (cuadrado y) x))   
                            transformada de newton   
                            1.0))

Comenzamos la sección  [1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3) con la observación de que los procedimientos compuestos son un mecanismo de abstracción crucial, porque nos permiten expresar métodos generales de computación como elementos explícitos en nuestro lenguaje de programación. Ahora hemos visto cómo los procedimientos de orden superior nos permiten manipular estos métodos generales para crear más abstracciones.

Como programadores, debemos estar alertas a las oportunidades de identificar las abstracciones subyacentes en nuestros programas y desarrollarlas y generalizarlas para crear abstracciones más poderosas. Esto no quiere decir que uno siempre deba escribir programas de la manera más abstracta posible; los programadores expertos saben cómo elegir el nivel de abstracción apropiado para su tarea. Pero es importante poder pensar en términos de estas abstracciones, de modo que podamos estar listos para aplicarlas en nuevos contextos. La importancia de los procedimientos de orden superior es que nos permiten representar estas abstracciones explícitamente como elementos en nuestro lenguaje de programación, de modo que puedan manejarse como otros elementos computacionales.

En general, los lenguajes de programación imponen restricciones sobre las formas en que se pueden manipular los elementos computacionales. Se dice que los elementos con menos restricciones tienen*condición de primera clase* . Algunos de los «derechos y privilegios» de los elementos de primera clase son: [64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_121)

* Pueden ser nombrados por variables.
* Pueden pasarse como argumentos a los procedimientos.
* Podrán ser devueltos como resultado de procedimientos.
* Pueden incluirse en estructuras de datos. [65](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_122)

Lisp, a diferencia de otros lenguajes de programación comunes, otorga a los procedimientos el estatus de primera clase. Esto plantea desafíos para una implementación eficiente, pero la ganancia resultante en poder expresivo es enorme. [66](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_123)

**Ejercicio 1.40.**   Defina un procedimiento cúbico que pueda utilizarse junto con el procedimiento del método de Newton en expresiones de la forma

(método de newtons (cúbico a b c) 1)

para aproximar ceros del cúbico *x* 3 + *a x* 2 + *b x* + *c* .

**Ejercicio 1.41.**   Defina un procedimiento double que tome un procedimiento de un argumento como argumento y devuelva un procedimiento que aplique el procedimiento original dos veces. Por ejemplo, si inc es un procedimiento que suma 1 a su argumento, entonces (double inc) debería ser un procedimiento que sume 2. ¿Qué valor devuelve

(((doble (doble doble)) inc) 5)

**Ejercicio 1.42.**  Sean *f* y *g* dos funciones de un argumento. La *composición f* después de *g* se define como la función *x* *f* ( *g* ( *x* )). Defina un procedimiento compose que implemente la composición. Por ejemplo, si inc es un procedimiento que suma 1 a su argumento,

((componer cuadrado inc) 6)   
*49*

**Ejercicio 1.43.**  Si *f* es una función numérica y *n* es un entero positivo, entonces podemos formar la *n-* ésima aplicación repetida de *f* , que se define como la función cuyo valor en *x* es *f* ( *f* ( ... ( *f* ( *x* )) ... )). Por ejemplo, si *f* es la función *x* *x* + 1, entonces la *n* -ésima aplicación repetida de *f* es la función *x* *x* + *n* . Si *f* es la operación de elevar al cuadrado un número, entonces la *n* -ésima aplicación repetida de *f* es la función que eleva su argumento a la 2 *-* ésima potencia. Escriba un procedimiento que tome como entradas un procedimiento que calcule *f* y un entero positivo *n* y devuelva el procedimiento que calcula la *n-* ésima aplicación repetida de *f* . Su procedimiento debería poder usarse de la siguiente manera:

((cuadrado repetido 2) 5)   
*625*

Sugerencia: Puede que le resulte conveniente utilizar la función componer del ejercicio  [1.42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.42) .

**Ejercicio 1.44.**  La idea de *suavizar* una función es un concepto importante en el procesamiento de señales. Si *f* es una función y *d x* es un número pequeño, entonces la versión suavizada de *f* es la función cuyo valor en un punto *x* es el promedio de *f* ( *x* - *d x* ), *f* ( *x* ) y *f* ( *x* + *d x* ). Escriba un procedimiento suavizado que tome como entrada un procedimiento que calcule *f* y devuelva un procedimiento que calcule la *f* suavizada . A veces es valioso suavizar repetidamente una función (es decir, suavizar la función suavizada, y así sucesivamente) para obtener la *función suavizada n* veces . Muestre cómo generar la función suavizada *n* veces de cualquier función dada usando suavizar y repetido del ejercicio  [1.43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.43) .

**Ejercicio 1.45.**   Vimos en la sección  [1.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.3) que intentar calcular raíces cuadradas hallando ingenuamente un punto fijo de *y* *x* / *y no converge, y que esto se puede solucionar mediante amortiguamiento promedio. El mismo método funciona para hallar raíces cúbicas como puntos fijos de y* *x* / *y* 2 amortiguados promedio . Desafortunadamente, el proceso no funciona paraCuarta raíz: una sola humedad promedio no es suficiente para hacer que una búsqueda de punto fijo para *y* *x* / *y* 3 converja. Por otro lado, si promediamos la humedad dos veces (es decir, usamos la humedad promedio de la humedad promedio de *y* *x* / *y* 3 ) la búsqueda de punto fijo converge. Realice algunos experimentos para determinar cuántas humedades promedio se requieren para calcular*n* ésimas raíces como una búsqueda de punto fijo basada en la amortiguación promedio repetida de *y* *x* / *y n* -1 . Use esto para implementar un procedimiento simple para calcular *n* ésimas raíces usando punto fijo , amortiguación promedio y el procedimiento repetido del ejercicio  [1.43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.43) . Suponga que todas las operaciones aritméticas que necesita están disponibles como primitivas.

**Ejercicio 1.46.**  Varios de los métodos numéricos descritos en este capítulo son ejemplos de una estrategia computacional extremadamente general conocida como *mejora iterativa* . La mejora iterativa dice que, para calcular algo, comenzamos con una suposición inicial para la respuesta, probamos si la suposición es lo suficientemente buena y, de lo contrario, mejoramos la suposición y continuamos el proceso usando la suposición mejorada como la nueva suposición. Escriba un procedimiento iterative-improve que tome dos procedimientos como argumentos: un método para determinar si una suposición es lo suficientemente buena y un método para mejorar una suposición. Iterative-improve debe devolver como su valor un procedimiento que toma una suposición como argumento y continúa mejorando la suposición hasta que sea lo suficientemente buena. Reescriba el procedimiento sqrt de la sección  [1.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.7) y el procedimiento  de punto fijo de la sección [1.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.3) en términos de iterative-improve .

[49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_90) Esta serie,Generalmente se escribe en la forma equivalente ( /4) = 1 - (1/3) + (1/5) - (1/7) + Â·Â·Â· , se debe a Leibniz. Veremos cómo usar esto como base para algunos trucos numéricos sofisticados en la sección  [3.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.3) .

[50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_91) Observe que hemos utilizado la estructura de bloques (sección [1.1.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.8)) para incorporar las definiciones depi-nextypi-termdentro depi-sum, ya que es poco probable que estos procedimientos sean útiles para cualquier otro propósito. Veremos cómo deshacernos de ellos por completo en la sección [1.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.2).

[51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_95) El propósito de los ejercicios [1.31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.31)a[1.33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.33)es demostrar el poder expresivo que se logra al utilizar una abstracción apropiada para consolidar muchas operaciones aparentemente dispares. Sin embargo, aunque la acumulación y el filtrado son ideas elegantes, estamos un poco atados de manos para utilizarlas en este punto, ya que aún no tenemos estructuras de datos que proporcionen medios adecuados de combinación para estas abstracciones. Volveremos a estas ideas en la sección [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3)cuando mostremos cómo utilizar*secuencias*como interfaces para combinar filtros y acumuladores para construir abstracciones aún más poderosas. Veremos allí cómo estos métodos realmente se convierten en un enfoque poderoso y elegante para diseñar programas.

[52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_96) Esta fórmula fue descubierta por el filósofo del siglo XVII.Matemático inglés John Wallis.

Sería más claro y menos intimidante para las personas que están aprendiendo Lisp si se utilizara un nombre más obvio que [lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_99) ,comomake-procedure. Pero la convención está firmemente arraigada. La notación se adopta de lacálculo, unFormalismo matemático introducido por el lógico matemático Alonzo Church (1941). Church desarrolló el cálculo para proporcionar una base rigurosa para el estudio de las nociones de función y aplicación de funciones. El cálculo se ha convertido en una herramienta básica para las investigaciones matemáticas de la semántica de los lenguajes de programación.

[54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_101) Para comprender las definiciones internas lo suficientemente bien como para estar seguros de que un programa significa lo que pretendemos que signifique, se requiere un modelo más elaborado del proceso de evaluación que el que hemos presentado en este capítulo. Sin embargo, las sutilezas no surgen con las definiciones internas de los procedimientos. Volveremos a esta cuestión en la sección [4.1.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.6), después de que aprendamos más sobre la evaluación.

[55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_104) Hemos utilizado 0,001 como un número "pequeño" representativo para indicar una tolerancia para el error aceptable en un cálculo. La tolerancia adecuada para un cálculo real depende del problema a resolver y de las limitaciones de la computadora y del algoritmo. Esto suele seruna consideración muy sutil, que requiere la ayuda de un analista numérico o algún otro tipo de mago.

[56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_105) Estose puede lograr usando error , que toma como argumentos una serie de elementos que se imprimen como mensajes de error.

[57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_107) Pruebe esto durante una clase aburrida: configure su calculadora enmodo radianes y luego presione repetidamente el botón cos hasta obtener el punto fijo.

[58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_108) (pronunciado "maps to") es la forma que tienen los matemáticos de escribir lambda . *y* *x* / *y* significa (lambda(y) (/ xy)) , es decir, la función cuyo valor en *y* es *x* / *y* .

[59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_114) Obsérvese que esta es una combinación cuyo operador es él mismouna combinación. El ejercicio  [1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.4) ya demostró la capacidad de formar tales combinaciones, pero ese fue solo un ejemplo práctico. Aquí comenzamos a ver la necesidad real de tales combinaciones, cuando se aplica un procedimiento que se obtiene como el valor devuelto por un procedimiento de orden superior.

[60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_115) Véase el ejercicio [1.45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.45)para una generalización adicional.

[61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_117) Los libros de cálculo elemental suelen describir el método de Newton en términos de la secuencia de aproximaciones*x n* +1=*x n* -*g*(*x n* )/*Dg*(*x n* ). Tener un lenguaje para hablar de procesos y utilizar la idea de puntos fijos simplifica la descripción del método.

[62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_118) El método de Newton no siempre converge a una respuesta, pero se puede demostrar que en casos favorables cada iteración duplica la precisión en número de dígitos de la aproximación a la solución. En tales casos,El método de Newton convergerá mucho más rápidamente que el método del medio intervalo.

[63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_119) Para encontrar raíces cuadradas, el método de Newton converge rápidamente a la solución correcta desde cualquier punto de partida.

[64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_121) La noción de estatus de primera clase del lenguaje de programaciónLos elementos se deben al científico informático británico Christopher Strachey (1916-1975).

Veremos ejemplos de esto después de introducir las estructuras de datos en el capítulo 2 [.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_122)

[El principal costo de implementación de los procedimientos de primera clase es que](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_123) permitir que los procedimientos se devuelvan como valores requiere reservar almacenamiento para las variables libres de un procedimiento incluso cuando el procedimiento no se está ejecutando. En la implementación de Scheme que estudiaremos en la sección [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1), estas variables se almacenan en el entorno del procedimiento.

[**Capítulo 2**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_2)

[**Construyendo abstracciones con datos**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_2)

|  |
| --- |
| Llegamos ahora al paso decisivo de la abstracción matemática: nos olvidamos de lo que representan los símbolos. ... [El matemático] no necesita estar ocioso; hay muchas operaciones que puede llevar a cabo con estos símbolos, sin tener que mirar nunca las cosas que representan.  Hermann Weyl, *El modo matemático de pensar* |

En el capítulo 1 nos concentramos en los procesos computacionales y en el papel de los procedimientos en el diseño de programas. Vimos cómo utilizar datos primitivos (números) y operaciones primitivas (operaciones aritméticas), cómo combinar procedimientos para formar procedimientos compuestos mediante la composición, los condicionales y el uso de parámetros, y cómo abstraer procedimientos mediante el uso de define . Vimos que un procedimiento puede considerarse como un patrón para la evolución local de un proceso, y clasificamos, razonamos y realizamos análisis algorítmicos simples de algunos patrones comunes para procesos tal como se incorporan en procedimientos. También vimos que los procedimientos de orden superior mejoran el poder de nuestro lenguaje al permitirnos manipular y, por lo tanto, razonar en términos de métodos generales de computación. Esto es gran parte de la esencia de la programación.

En este capítulo vamos a ver datos más complejos. Todos los procedimientos del capítulo 1 operan sobre datos numéricos simples, y los datos simples no son suficientes para muchos de los problemas que deseamos abordar mediante la computación. Los programas suelen estar diseñados para modelar fenómenos complejos y, en la mayoría de los casos, es necesario construir objetos computacionales que tengan varias partes para modelar fenómenos del mundo real que tengan varios aspectos. Por lo tanto, mientras que en el capítulo 1 nos centramos en la creación de abstracciones mediante la combinación de procedimientos para formar procedimientos compuestos, en este capítulo nos centramos en otro aspecto clave de cualquier lenguaje de programación: los medios que proporciona para crear abstracciones mediante la combinación de objetos de datos para formar *datos compuestos* .

¿Por qué queremos datos compuestos en un lenguaje de programación? Por las mismas razones por las que queremos procedimientos compuestos: para elevar el nivel conceptual en el que podemos diseñar nuestros programas, para aumentar la modularidad de nuestros diseños y para mejorar el poder expresivo de nuestro lenguaje. Así como la capacidad de definir procedimientos nos permite tratar con procesos a un nivel conceptual más alto que el de las operaciones primitivas del lenguaje, la capacidad de construir objetos de datos compuestos nos permite tratar con datos a un nivel conceptual más alto que el de los objetos de datos primitivos del lenguaje.

Consideremos la tarea de diseñar un sistema para realizar operaciones aritméticas con números racionales. Podríamos imaginar una operación add-rat que toma dos números racionales y produce su suma. En términos de datos simples, un número racional puede considerarse como dos enteros: un numerador y un denominador. Por lo tanto, podríamos diseñar un programa en el que cada número racional estaría representado por dos enteros (un numerador y un denominador) y donde add-rat se implementaría mediante dos procedimientos (uno que produce el numerador de la suma y otro que produce el denominador). Pero esto sería complicado, porque entonces tendríamos que llevar un registro explícito de qué numeradores corresponden a qué denominadores. En un sistema destinado a realizar muchas operaciones con muchos números racionales, esos detalles de contabilidad desordenarían sustancialmente los programas, por no hablar de lo que le harían a nuestras mentes. Sería mucho mejor si pudiéramos "pegar" un numerador y un denominador para formar un par -un *objeto de datos compuesto-* que nuestros programas pudieran manipular de una manera que fuera coherente con considerar un número racional como una sola unidad conceptual.

El uso de datos compuestos también nos permite aumentar la modularidad de nuestros programas. Si podemos manipular números racionales directamente como objetos por derecho propio, entonces podemos separar la parte de nuestro programa que trata con números racionales per se de los detalles de cómo los números racionales pueden representarse como pares de números enteros. La técnica general de aislar las partes de un programa que tratan de cómo se representan los objetos de datos de las partes de un programa que tratan de cómo se utilizan los objetos de datos es una poderosa metodología de diseño llamada*Abstracción de datos* . Veremos cómo la abstracción de datos hace que los programas sean mucho más fáciles de diseñar, mantener y modificar.

El uso de datos compuestos conduce a un aumento real en el poder expresivo de nuestro lenguaje de programación. Considere la idea de formar una "combinación lineal" *a x* + *b y* . Podríamos escribir un procedimiento que aceptara *a* , *b* , *x e* y como argumentos y devolviera el valor de *a x* + *b y* . Esto no presenta ninguna dificultad si los argumentos deben ser números, porque podemos definir fácilmente el procedimiento.

(definir (combinación lineal a b x y)    
  (+ (\* a x) (\* b y)))

Pero supongamos que no nos interesan sólo los números. Supongamos que quisiéramos expresar, en términos procedimentales, la idea de que se pueden formar combinaciones lineales siempre que se definan la adición y la multiplicación, para números racionales, números complejos, polinomios o lo que sea. Podríamos expresar esto como un procedimiento de la forma

(definir (combinación lineal a b x y)        
  (añadir (mul a x) (mul b y)))

donde add y mul no son los procedimientos primitivos + y \* sino cosas más complejas que realizarán las operaciones apropiadas para cualquier tipo de datos que pasemos como argumentos a , b , x e y . El punto clave es que lo único que linear-combination debería necesitar saber sobre a , b , x e y es que los procedimientos add y mul realizarán las manipulaciones apropiadas. Desde la perspectiva del procedimiento linear-combination , es irrelevante qué son a , b , x e y y aún más irrelevante cómo podrían representarse en términos de datos más primitivos. Este mismo ejemplo muestra por qué es importante que nuestro lenguaje de programación proporcione la capacidad de manipular objetos compuestos directamente: Sin esto, no hay forma de que un procedimiento como linear-combination pase sus argumentos a add y mul sin tener que conocer su estructura detallada. [1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_131) Comenzamos este capítulo implementando el sistema aritmético de números racionales mencionado anteriormente. Esto formará la base para nuestra discusión de datos compuestos y abstracción de datos. Al igual que con los procedimientos compuestos, la cuestión principal que se abordará es la de la abstracción como técnica para hacer frente a la complejidad, y veremos cómo la abstracción de datos nos permite construir procedimientos adecuados.*Barreras de abstracción* entre diferentes partes de un programa.

Veremos que la clave para formar datos compuestos es que un lenguaje de programación debe proporcionar algún tipo de "pegamento" para que los objetos de datos se puedan combinar para formar objetos de datos más complejos. Hay muchos tipos posibles de pegamento. De hecho, descubriremos cómo formar datos compuestos sin utilizar operaciones especiales de "datos", solo procedimientos. Esto desdibujará aún más la distinción entre "procedimiento" y "datos", que ya se estaba volviendo tenue hacia el final del capítulo 1. También exploraremos algunas técnicas convencionales para representar secuencias y árboles. Una idea clave al tratar con datos compuestos es la noción de*Cierre* : el pegamento que utilizamos para combinar objetos de datos debería permitirnos combinar no solo objetos de datos primitivos, sino también objetos de datos compuestos. Otra idea clave es que los objetos de datos compuestos pueden servir como*Interfaces convencionales* para combinar módulos de programas de forma que se puedan combinar. Ilustramos algunas de estas ideas presentando un lenguaje gráfico simple que aprovecha el cierre.

Luego aumentaremos el poder de representación de nuestro lenguaje introduciendo*Expresiones simbólicas* : datos cuyas partes elementales pueden ser símbolos arbitrarios en lugar de solo números. Exploramos varias alternativas para representar conjuntos de objetos. Descubriremos que, así como una función numérica dada puede calcularse mediante muchos procesos computacionales diferentes, existen muchas maneras en las que una estructura de datos dada puede representarse en términos de objetos más simples, y la elección de la representación puede tener un impacto significativo en los requisitos de tiempo y espacio de los procesos que manipulan los datos. Investigaremos estas ideas en el contexto de la diferenciación simbólica, la representación de conjuntos y la codificación de la información.

A continuación abordaremos el problema de trabajar con datos que pueden representarse de forma diferente en distintas partes de un programa. Esto nos lleva a la necesidad de implementar*Operaciones genéricas* , que deben manejar muchos tipos diferentes de datos. Mantener la modularidad en presencia de operaciones genéricas requiere barreras de abstracción más poderosas que las que se pueden erigir con la simple abstracción de datos por sí sola. En particular, presentamos *la programación dirigida por datos* como una técnica que permite diseñar representaciones de datos individuales de forma aislada y luego combinarlas.*aditivamente* (es decir, sin modificación). Para ilustrar el poder de este enfoque para el diseño de sistemas, cerramos el capítulo aplicando lo que hemos aprendido a la implementación de un paquete para realizar aritmética simbólica con polinomios, en el que los coeficientes de los polinomios pueden ser números enteros, números racionales, números complejos e incluso otros polinomios.

[1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_131) La capacidad de manipular directamente los procedimientos proporciona un aumento análogo en el poder expresivo de un lenguaje de programación. Por ejemplo, en la sección [1.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.1)introdujimos elprocedimientosumatérminocomo argumento y calcula la suma de los valores detérminoen un intervalo especificado. Para definirsuma, es crucial que podamos hablar de un procedimiento comotérminocomo una entidad por derecho propio, sin tener en cuenta cómotérminocon operaciones más primitivas. De hecho, si no tuviéramos la noción de ``un procedimiento'', es dudoso que alguna vez pensáramos en la posibilidad de definir una operación comosuma. Además, en lo que respecta a la realización de la suma, los detalles de cómotérminoa partir de operaciones más primitivas son irrelevantes.

[**2.1 Introducción a la abstracción de datos**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_2.1)

En la sección  [1.1.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.8) , notamos que un procedimiento utilizado como elemento en la creación de un procedimiento más complejo podría considerarse no solo como una colección de operaciones particulares sino también como una abstracción procedimental. Es decir, los detalles de cómo se implementó el procedimiento podrían suprimirse, y el procedimiento particular en sí podría reemplazarse por cualquier otro procedimiento con el mismo comportamiento general. En otras palabras, podríamos hacer una abstracción que separaría la forma en que se usaría el procedimiento de los detalles de cómo se implementaría el procedimiento en términos de procedimientos más primitivos. La noción análoga para datos compuestos se llama*Abstracción de datos* . La abstracción de datos es una metodología que nos permite aislar cómo se utiliza un objeto de datos compuesto de los detalles de cómo se construye a partir de objetos de datos más primitivos.

La idea básica de la abstracción de datos es estructurar los programas que van a utilizar objetos de datos compuestos para que operen en"datos abstractos". Es decir, nuestros programas deben utilizar los datos de tal manera que no hagan suposiciones sobre ellos que no sean estrictamente necesarias para realizar la tarea en cuestión. Al mismo tiempo, unLa representación de datos "concreta" se define independientemente de los programas que utilizan los datos. La interfaz entre estas dos partes de nuestro sistema será un conjunto de procedimientos, denominados*selectores* y*constructores* , que implementan los datos abstractos en términos de la representación concreta. Para ilustrar esta técnica, consideraremos cómo diseñar un conjunto de procedimientos para manipular números racionales.

**[2.1.1 Ejemplo: Operaciones aritméticas para números racionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.1.1)**

Supongamos que queremos hacer operaciones aritméticas con números racionales. Queremos poder sumar, restar, multiplicar y dividir números racionales y comprobar si dos números racionales son iguales.

Comencemos suponiendo que ya tenemos una forma de construir un número racional a partir de un numerador y un denominador. Supongamos también que, dado un número racional, tenemos una forma de extraer (o seleccionar) su numerador y su denominador. Supongamos además que el constructor y los selectores están disponibles como procedimientos:

* (make-rat < *n* > < *d* >) devuelve el número racional cuyo numerador es el entero < *n* > y cuyo denominador es el entero < *d* > .
* (numero < *x* >) devuelve el numerador del número racional < *x* > .
* (denom < *x* >) devuelve el denominador del número racional < *x* > .

Utilizamos aquí una potente estrategia de síntesis:*Es una ilusión* . Aún no hemos dicho cómo se representa un número racional ni cómo se deben implementar los procedimientos numer , denom y make-rat . Aun así, si tuviéramos estos tres procedimientos, podríamos sumar, restar, multiplicar, dividir y comprobar la igualdad utilizando las siguientes relaciones:











Podemos expresar estas reglas como procedimientos:

(definir (add-rat x y)   
  (make-rat (+ (\* (numer x) (denom y))   
               (\* (numer y) (denom x)))   
            (\* (denom x) (denom y))))  
(definir (sub-rat x y)   
  (make-rat (- (\* (numer x) (denom y))   
               (\* (numer y) (denom x)))   
            (\* (denom x) (denom y))))  
(definir (mul-rat x y)   
  (make-rat (\* (número x) (número y))   
            (\* (denom x) (denom y))))  
(definir (div-rat x y)   
  (make-rat (\* (número x) (denom y))   
            (\* (denom x) (número y))))  
(definir (igual-rata? x y)   
  (= (\* (número x) (denominación y))   
     (\* (número y) (denominación x))))

Ahora tenemos las operaciones con números racionales definidas en términos de los procedimientos selectores y constructores numer , denom y make-rat . Pero aún no los hemos definido. Lo que necesitamos es alguna forma de unir un numerador y un denominador para formar un número racional.

**[Pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_132)**

Para permitirnos implementar el nivel concreto de nuestra abstracción de datos, nuestro lenguaje proporciona una estructura compuesta llamada*par* , que se puede construir con el procedimiento primitivocons . Este procedimiento toma dos argumentos y devuelve un objeto de datos compuesto que contiene los dos argumentos como partes. Dado un par, podemos extraer las partes utilizando los procedimientos primitivoscoche ycdr . [2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_133) Por lo tanto, podemos utilizar cons , car y cdr de la siguiente manera:

(define x (cons 1 2))   
  
(auto x)   
*1*  
  
(cdr x)   
*2*

Tenga en cuenta que un par es un objeto de datos al que se le puede asignar un nombre y manipular, al igual que un objeto de datos primitivo. Además, se pueden utilizar cons para formar pares cuyos elementos sean pares, y así sucesivamente:

(define x (cons 1 2))   
  
(define y (cons 3 4))   
  
(define z (cons x y))   
  
(auto (auto z))   
*1*  
  
(auto (cdr z))   
*3*

En la sección  [2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2) veremos cómo esta capacidad de combinar pares significa que los pares pueden usarse como bloques de construcción de propósito general para crear todo tipo de estructuras de datos complejas. El *par* primitivo de datos compuestos único , implementado por los procedimientos cons , car y cdr , es el único pegamento que necesitamos. Los objetos de datos construidos a partir de pares se denominandatos *estructurados en lista .*

**[Representación de números racionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_134)**

Los pares ofrecen una forma natural de completar el sistema de números racionales. Simplemente, represente un número racional como un par de dos enteros: un numerador y un denominador. Luego, make-rat , numer y denom se implementan fácilmente de la siguiente manera: [3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_135)

(definir (hacer-rata n d) (cons n d))  
  
(define (numero x) (coche x))  
  
(definir (denom x) (cdr x))

Además, para mostrar los resultados de nuestros cálculos, podemosImprima números racionales imprimiendo el numerador, una barra y el denominador: [4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_136)

(define (print-rat x)   
  (nueva línea)   
  (muestra (número x))   
  (muestra "/")   
  (muestra (denom x)))

Ahora podemos probar nuestros procedimientos con números racionales:

(definir la mitad (hacer-rata 1 2))   
  
(imprimir-rata la mitad)   
*1/2*  
  
(definir un tercio (hacer-rata 1 3))   
(imprimir-rata (agregar-rata la mitad un tercio))   
*5/6*  
  
(imprimir-rata (mul-rata la mitad un tercio))   
*1/6*  
  
(imprimir-rata (agregar-rata un tercio un tercio))   
*6/9*

Como muestra el ejemplo final, nuestra implementación de números racionales no reduce los números racionales a su mínima expresión. Podemos solucionar esto modificando make-rat . Si tenemos unprocedimiento mcd como el de la sección  [1.2.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.5) que produce el máximo común divisor de dos números enteros, podemos usar mcd para reducir el numerador y el denominador a sus términos más bajos antes de construir el par:

(define (make-rat n d)   
  (sea ((g (mcd n d)))   
    (cons (/ n g) (/ d g))))

Ahora tenemos

(imprimir-rata (añadir-rata un tercio un tercio))   
*2/3*

como se desea. Esta modificación se logró cambiando el constructor make-rat sin cambiar ninguno de los procedimientos (como add-rat y mul-rat ) que implementan las operaciones reales.

**Ejercicio 2.1.**   Defina una versión mejor de make-rat que maneje argumentos tanto positivos como negativos. Make-rat debería normalizar el signo de modo que si el número racional es positivo, tanto el numerador como el denominador sean positivos, y si el número racional es negativo, solo el numerador sea negativo.

**[2.1.2 Barreras de abstracción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.1.2)**

Antes de continuar con más ejemplos de datos compuestos y abstracción de datos, consideremos algunas de las cuestiones planteadas por el ejemplo de los números racionales. Definimos las operaciones de números racionales en términos de un constructor make-rat y selectores numer y denom . En general, la idea subyacente de la abstracción de datos es identificar para cada tipo de objeto de datos un conjunto básico de operaciones en términos de las cuales se expresarán todas las manipulaciones de los objetos de datos de ese tipo y luego usar solo esas operaciones para manipular los datos.

Podemos imaginar la estructura del sistema de números racionales como se muestra en la figura  [2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.1) . Las líneas horizontales representan *barreras de abstracción* que aíslan diferentes "niveles" del sistema. En cada nivel, la barrera separa los programas (arriba) que usan la abstracción de datos de los programas (abajo) que implementan la abstracción de datos. Los programas que usan números racionales los manipulan únicamente en términos de los procedimientos proporcionados "para uso público" por el paquete de números racionales: add-rat , sub-rat , mul-rat , div-rat y equal-rat ?. Estos, a su vez, se implementan únicamente en términos de los procedimientosconstructor y selectores make-rat , numer y denom , que a su vez se implementan en términos de pares. Los detalles de cómo se implementan los pares son irrelevantes para el resto del paquete de números racionales siempre que los pares se puedan manipular mediante el uso de cons , car y cdr . En efecto, los procedimientos en cada nivel son las interfaces que definen las barreras de abstracción y conectan los diferentes niveles.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.1:**   Barreras de abstracción de datos en el paquete de números racionales. |

Esta idea sencilla tiene muchas ventajas. Una de ellas es que hace que los programas sean mucho más fáciles de mantener y modificar. Cualquier estructura de datos compleja se puede representar de diversas maneras con las estructuras de datos primitivas que proporciona un lenguaje de programación. Por supuesto, la elección de la representación influye en los programas que operan sobre ella; por lo tanto, si la representación se cambiara en algún momento posterior, todos esos programas tendrían que modificarse en consecuencia. Esta tarea podría consumir mucho tiempo y resultar costosa en el caso de programas grandes, a menos que la dependencia de la representación se limitara por diseño a unos pocos módulos del programa.

Por ejemplo, una forma alternativa de abordar el problema de reducir números racionales a sus términos más bajos es realizar la reducción cada vez que accedemos a las partes de un número racional, en lugar de cuando lo construimos. Esto conduce a diferentes procedimientos de constructor y selector:

(definir (hacer-rata n d)   
  (cons n d))  
(define (número x)   
  (sea ((g (mcd (automóvil x) (cdr x))))   
    (/ (automóvil x) g)))  
(define (denom x)   
  (sea ((g (mcd (car x) (cdr x))))   
    (/ (cdr x) g)))

La diferencia entre esta implementación y la anterior radica en cuándo calculamos el mcd . Si en nuestro uso típico de los números racionales accedemos a los numeradores y denominadores de los mismos números racionales muchas veces, sería preferible calcular el mcd cuando se construyen los números racionales. Si no, puede que sea mejor esperar hasta el momento del acceso para calcular el mcd . En cualquier caso, cuando cambiamos de una representación a otra, los procedimientos add-rat , sub-rat , etc., no tienen por qué modificarse en absoluto.

Restringir la dependencia de la representación a unos pocos procedimientos de interfaz nos ayuda a diseñar programas y a modificarlos, porque nos permite mantener la flexibilidad para considerar implementaciones alternativas. Para continuar con nuestro ejemplo simple, supongamos que estamos diseñando un paquete de números racionales y no podemos decidir inicialmente si realizar el mcd en el momento de la construcción o en el momento de la selección. La metodología de abstracción de datos nos brinda una manera de posponer esa decisión sin perder la capacidad de avanzar en el resto del sistema.

**Ejercicio 2.2.**   Considere el problema de representarsegmentos de línea en un plano. Cada segmento se representa como un par de puntos: un punto inicial y un punto final. Defina un constructorSegmento de creación y selectoressegmento de inicio ysegmento final que define la representación de segmentos en términos de puntos. Además, un puntose puede representar como un par de números: la coordenada *x y la coordenada y* . En consecuencia, especifique un constructorPunto de creación y selectores punto x y punto y que definen esta representación. Por último, utilizando sus selectores y constructores, defina un procedimientosegmento de punto medio que toma un segmento de línea como argumento y devuelve su punto medio (el punto cuyas coordenadas son el promedio de las coordenadas de los puntos finales). Para probar sus procedimientos, necesitará una forma de imprimir puntos:

(define (punto de impresión p)   
  (nueva línea)   
  (mostrar "(")   
  (mostrar (punto x p))   
  (mostrar ",")   
  (mostrar (punto y p))   
  (mostrar ")"))

**Ejercicio 2.3.**  Implemente una representación para rectángulos en un plano. (Sugerencia: puede que desee utilizar el ejercicio  [2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.2) ). En términos de sus constructores y selectores, cree procedimientos que calculen el perímetro y el área de un rectángulo dado. Ahora implemente una representación diferente para los rectángulos. ¿Puede diseñar su sistema con barreras de abstracción adecuadas, de modo que los mismos procedimientos de perímetro y área funcionen utilizando cualquiera de las representaciones?

**[2.1.3 ¿Qué se entiende por datos?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.1.3)**

Comenzamos la implementación de números racionales en la sección  [2.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.1) implementando las operaciones de números racionales add-rat , sub-rat , etc. en términos de tres procedimientos no especificados: make-rat , numer y denom . En ese punto, podríamos pensar en las operaciones como definidas en términos de objetos de datos (numeradores, denominadores y números racionales) cuyo comportamiento estaba especificado por los últimos tres procedimientos.

Pero, ¿qué se entiende exactamente por *datos* ? No basta con decir "lo que sea implementado por los selectores y constructores dados". Es evidente que no todo conjunto arbitrario de tres procedimientos puede servir como base apropiada para la implementación de números racionales. Necesitamos garantizar que,Si construimos un número racional x a partir de un par de números enteros n y d , entonces extraer el numer y el denom de x y dividirlos debería dar el mismo resultado que dividir n por d . En otras palabras, make-rat , numer y denom deben satisfacer la condición de que, para cualquier número entero n y cualquier número entero distinto de cero d , si x es ( make-rat nd ), entonces



De hecho, esta es la única condición que make-rat , numer y denom deben cumplir para formar una base adecuada para una representación de números racionales. En general, podemos pensar en los datos como definidos por una colección de selectores y constructores, junto con condiciones específicas que estos procedimientos deben cumplir para ser una representación válida. [5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_140)

Este punto de vista puede servir para definir no sólo objetos de datos de "alto nivel", como números racionales, sino también objetos de nivel inferior.Consideremos la noción de par, que utilizamos para definir nuestros números racionales. En realidad, nunca dijimos qué era un par, solo que el lenguaje proporcionaba los procedimientos cons , car y cdr para operar con pares. Pero lo único que necesitamos saber sobre estas tres operaciones es quees que si pegamos dos objetos juntos usando cons podemos recuperar los objetos usando car y cdr . Es decir, las operaciones satisfacen la condición de que, para cualquier objeto x e y , si z es (cons xy) entonces (car z) es x y (cdr z) es y . De hecho, mencionamos que estos tres procedimientos están incluidos como primitivos en nuestro lenguaje. Sin embargo, cualquier triple de procedimientos que satisfaga la condición anterior puede usarse como base para implementar pares. Este punto se ilustra de manera sorprendente por el hecho de que podríamos implementar cons , car y cdr sin usar ninguna estructura de datos en absoluto, sino solo usando procedimientos. Aquí están las definiciones:

(define (cons x y)   
  (define (enviar m)   
    (cond ((= m 0) x)   
          ((= m 1) y)   
          (de lo contrario (error "Argumento no 0 o 1 -- CONS" m))))   
  enviar)  
  
(define (coche z) (z 0))  
  
(define (cdr z) (z 1))

Este uso de procedimientos no se corresponde en nada con nuestra noción intuitiva de lo que deberían ser los datos. Sin embargo, todo lo que necesitamos hacer para demostrar que esta es una forma válida de representar pares es verificar que estos procedimientos satisfacen la condición dada anteriormente.

El punto sutil a notar es que el valor devuelto por (cons xy) es un procedimiento - a saber, el procedimiento definido internamente dispatch , que toma un argumento y devuelve x o y dependiendo de si el argumento es 0 o 1. Correspondientemente, (car z) está definido para aplicar z a 0. Por lo tanto, si z es el procedimiento formado por (cons xy) , entonces z aplicado a 0 producirá x . Por lo tanto, hemos demostrado que (car (cons xy)) produce x , como se desea. De manera similar, (cdr (cons xy)) aplica el procedimiento devuelto por (cons xy) a 1, que devuelve y . Por lo tanto, esta implementación procedimental de pares es una implementación válida, y si accedemos a pares usando solo cons , car y cdr no podemos distinguir esta implementación de una que usa estructuras de datos "reales".

El objetivo de mostrar la representación procedimental de pares no es que nuestro lenguaje funcione de esta manera (Scheme y los sistemas Lisp en general implementan pares directamente, por razones de eficiencia), sino que podría funcionar de esta manera. La representación procedimental, aunque oscura, es una forma perfectamente adecuada de representar pares, ya que cumple las únicas condiciones que los pares deben cumplir. Este ejemplo también demuestra que la capacidad de manipular procedimientos como objetos proporciona automáticamente la capacidad de representar datos compuestos. Esto puede parecer una curiosidad ahora, pero las representaciones procedimentales de datos desempeñarán un papel central en nuestro repertorio de programación. Este estilo de programación a menudo se denomina*paso de mensajes* , y lo utilizaremos como herramienta básica en el capítulo 3 cuando abordemos los problemas de modelado y simulación.

**Ejercicio 2.4.**   A continuación se presenta una representación procedimental alternativa de pares. Para esta representación, verifique que (car (cons xy)) produce x para cualquier objeto x e y .

(define (cons x y)   
  (lambda (m) (m x y)))  
  
(define (coche z)   
  (z (lambda (p q) p)))

¿Cuál es la definición correspondiente de cdr ? (Sugerencia: para verificar que esto funciona, utilice el modelo de sustitución de la sección  [1.1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.5) ).

**Ejercicio 2.5.**   Demuestre que podemos representar pares de números enteros no negativos utilizando únicamente números y operaciones aritméticas si representamos el par *a* y *b* como el entero que es el producto 2 *a* 3 *b* . Dé las definiciones correspondientes de los procedimientos cons , car y cdr .

**Ejercicio 2.6.**   Por si representar pares como procedimientos no fuera lo suficientemente complicado, considere que, en un lenguaje que puede manipular procedimientos, podemos arreglárnoslas sin números (al menos en lo que respecta a los números enteros no negativos) implementando 0 y la operación de sumar 1 como

(define cero (lambda (f) (lambda (x) x)))   
  
(define (add-1 n)   
  (lambda (f) (lambda (x) (f ((n f) x)))))

Esta representación se conoce como*Numerales de la iglesia* , en honor a su inventor,Alonzo Church, el lógico que inventó el cálculo.

Defina uno y dos directamente (no en términos de cero y add-1 ). (Pista: use la sustitución para evaluar (add-1 zero) ). Dé una definición directa del procedimiento de adición + (no en términos de la aplicación repetida de add-1 ).

**[2.1.4 Ejercicio extendido: Aritmética de intervalos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.1.4)**

Alyssa P. Hacker está diseñando un sistema para ayudar a las personas a resolver problemas de ingeniería. Una característica que quiere ofrecer en su sistema es la capacidad de manipular cantidades inexactas (como parámetros medidos de dispositivos físicos) con precisión conocida, de modo que cuando se realicen cálculos con cantidades tan aproximadas, los resultados sean números de precisión conocida.

Los ingenieros eléctricos utilizarán el sistema de Alyssa para calcular magnitudes eléctricas. A veces es necesario que calculen el valor de una resistencia equivalente en paralelo *R p* de dos resistencias *R* 1 y *R* 2 utilizando la fórmula



Los valores de resistencia normalmente se conocen solo hasta cierto punto.Tolerancia garantizada por el fabricante de la resistencia. Por ejemplo, si compra una resistencia etiquetada como ``6,8 ohmios con 10% de tolerancia'', solo puede estar seguro de que la resistencia tiene una resistencia entre 6,8 - 0,68 = 6,12 y 6,8 + 0,68 = 7,48 ohmios. Por lo tanto, si tiene una resistencia de 6,8 ohmios al 10% en paralelo con una resistencia de 4,7 ohmios al 5%, la resistencia de la combinación puede variar de aproximadamente 2,58 ohmios (si las dos resistencias están en los límites inferiores) a aproximadamente 2,97 ohmios (si las dos resistencias están en los límites superiores).

La idea de Alyssa es implementar la "aritmética de intervalos" como un conjunto de operaciones aritméticas para combinar "intervalos" (objetos que representan el rango de valores posibles de una cantidad inexacta). El resultado de sumar, restar, multiplicar o dividir dos intervalos es en sí mismo un intervalo, que representa el rango del resultado.

Alyssa postula la existencia de un objeto abstracto llamado "intervalo" que tiene dos puntos finales: un límite inferior y un límite superior. También supone que, dados los puntos finales de un intervalo, puede construir el intervalo utilizando el constructor de datos.make-interval . Alyssa primero escribe un procedimiento para sumar dos intervalos. Razona que el valor mínimo que podría tener la suma es la suma de los dos límites inferiores y el valor máximo que podría tener es la suma de los dos límites superiores:

(define (suma-intervalo x y)   
  (hace-intervalo (+ (límite inferior x) (límite inferior y))   
                 (+ (límite superior x) (límite superior y))))

Alyssa también calcula el producto de dos intervalos al encontrar el mínimo y el máximo de los productos de los límites y usarlos como límites del intervalo resultante. ( Mín y máximo sonprimitivas que encuentran el mínimo o máximo de cualquier número de argumentos).

(define (multi-intervalo x y)   
  (sea ((p1 (\* (límite inferior x) (límite inferior y)))   
        (p2 (\* (límite inferior x) (límite superior y)))   
        (p3 (\* (límite superior x) (límite inferior y)))   
        (p4 (\* (límite superior x) (límite superior y))))   
    (make-interval (min p1 p2 p3 p4)   
                   (max p1 p2 p3 p4))))

Para dividir dos intervalos, Alyssa multiplica el primero por el recíproco del segundo. Observe que los límites del intervalo recíproco son el recíproco del límite superior y el recíproco del límite inferior, en ese orden.

(define (div-intervalo x y)   
  (mul-intervalo x    
                (make-intervalo (/ 1.0 (límite superior y))   
                               (/ 1.0 (límite inferior y)))))

**Ejercicio 2.7.**   El programa de Alyssa está incompleto porque no ha especificado la implementación de la abstracción de intervalo. A continuación se muestra una definición del constructor de intervalo:

(define (haz-intervalo a b) (cons a b))

Definir selectoreslímite superior ylímite inferior para completar la implementación.

**Ejercicio 2.8.**   Utilizando un razonamiento análogo al de Alyssa, describa cómo se puede calcular la diferencia de dos intervalos. Defina un procedimiento de resta correspondiente, llamadosubintervalo .

**Ejercicio 2.9.**  El *ancho* de un intervalo es la mitad de la diferencia entre sus límites superior e inferior. El ancho es una medida de la incertidumbre del número especificado por el intervalo. Para algunas operaciones aritméticas, el ancho del resultado de combinar dos intervalos es una función únicamente del ancho de los intervalos de los argumentos, mientras que para otras, el ancho de la combinación no es una función del ancho de los intervalos de los argumentos. Demuestre que el ancho de la suma (o diferencia) de dos intervalos es una función únicamente del ancho de los intervalos que se suman (o restan). Dé ejemplos para demostrar que esto no es cierto para la multiplicación o la división.

**Ejercicio 2.10.**  Ben Bitdiddle, un programador de sistemas experto, observa a Alyssa por encima del hombro y comenta que no está claro qué significa dividir por un intervalo que abarca el cero. Modifique el código de Alyssa para comprobar si se produce esta condición y señalar un error si se produce.

**Ejercicio 2.11.**  De paso, Ben también comenta crípticamente: "Al probar los signos de los puntos finales de los intervalos, es posible dividir el intervalo múltiple en nueve casos, de los cuales sólo uno requiere más de dos multiplicaciones". Reescribe este procedimiento usando la sugerencia de Ben.

Después de depurar su programa, Alyssa se lo muestra a un usuario potencial, quien se queja de que su programa resuelve el problema equivocado. Quiere un programa que pueda manejar números representados como un valor central y una tolerancia aditiva; por ejemplo, quiere trabajar con intervalos como 3,5 ± 0,15 en lugar de [3,35, 3,65]. Alyssa regresa a su escritorio y soluciona este problema proporcionando un constructor alternativo y selectores alternativos:

(define (hacer-centro-ancho c w)   
  (hacer-intervalo (- c w) (+ c w)))  
(define (centro i)   
  (/ (+ (límite inferior i) (límite superior i)) 2))  
(define (ancho i)   
  (/ (- (límite superior i) (límite inferior i)) 2))

Lamentablemente, la mayoría de los usuarios de Alyssa son ingenieros. Las situaciones reales de ingeniería suelen implicar mediciones con una pequeña incertidumbre, medida como la relación entre el ancho del intervalo y el punto medio del mismo. Los ingenieros suelen especificar tolerancias porcentuales en los parámetros de los dispositivos, como en las especificaciones de resistencias que se dieron anteriormente.

**Ejercicio 2.12.**   Definir un constructormake-center-percent que toma un centro y una tolerancia porcentual y produce el intervalo deseado. También debe definir un selector de porcentaje que produzca la tolerancia porcentual para un intervalo determinado. El selector de centro es el mismo que el que se muestra arriba.

**Ejercicio 2.13.**   Demuestre que, suponiendo que las tolerancias porcentuales son pequeñas, existe una fórmula sencilla para la tolerancia porcentual aproximada del producto de dos intervalos en términos de las tolerancias de los factores. Puede simplificar el problema suponiendo que todos los números son positivos.

Después de un trabajo considerable, Alyssa P. Hacker entrega su sistema terminado. Varios años después, cuando ya se había olvidado por completo de él, recibe una llamada frenética de un usuario furioso, Lem E. Tweakit. Parece que Lem se dio cuenta de que la fórmula para resistencias en paralelo se puede escribir de dos formas.formas algebraicamente equivalentes:



y



Ha escrito los dos programas siguientes, cada uno de los cuales calcula la fórmula de resistencias paralelas de forma diferente:

(define (par1 r1 r2)   
  (div-intervalo (multi-intervalo r1 r2)   
                (add-intervalo r1 r2)))   
(define (par2 r1 r2)   
  (let ((uno (make-intervalo 1 1)))    
    (div-intervalo uno   
                  (add-intervalo (div-intervalo uno r1)   
                                (div-intervalo uno r2)))))

Lem se queja de que el programa de Alyssa da respuestas diferentes para las dos formas de calcular. Esta es una queja seria.

**Ejercicio 2.14.**   Demuestre que Lem tiene razón. Investigue el comportamiento del sistema en una variedad de expresiones aritméticas. Cree algunos intervalos *A* y *B* y úselos para calcular las expresiones *A* / *A* y *A* / *B.* Obtendrá más información si utiliza intervalos cuyo ancho sea un pequeño porcentaje del valor central. Examine los resultados del cálculo en forma de porcentaje central (consulte el ejercicio  [2.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.12) ).

**Ejercicio 2.15.**   Eva Lu Ator, otra usuaria, también ha notado los diferentes intervalos calculados por expresiones diferentes pero algebraicamente equivalentes. Ella dice que una fórmula para calcular con intervalos utilizando el sistema de Alyssa producirá límites de error más estrictos si se puede escribir de tal forma que no se repita ninguna variable que represente un número incierto. Por lo tanto, dice, par2 es un programa ``mejor'' para resistencias paralelas que par1 . ¿Tiene razón? ¿Por qué?

**Ejercicio 2.16.**   Explique, en general, por qué expresiones algebraicas equivalentes pueden llevar a diferentes respuestas. ¿Puede idear un paquete de aritmética de intervalos que no presente esta deficiencia, o es esta tarea imposible? (Advertencia: este problema es muy difícil).

[2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_133) El nombrecons significa "construir". Los nombrescoche ycdr deriva de la implementación original de Lisp en elIBM 704. Esa máquina tenía un esquema de direccionamiento que permitía hacer referencia a las partes de ``dirección'' y ``decremento'' de una ubicación de memoria. Car significa ``Contenido de la parte de dirección del registro'' y cdr (que se pronuncia ``could-er'') significa ``Contenido de la parte de decremento del registro''.

[3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_135) Otra forma de definir los selectores y el constructor es

(definir make-rat cons)   
(definir num car)   
(definir denom cdr)

La primera definición asocia el nombre make-rat con el valor de la expresión cons , que es el procedimiento primitivo que construye pares. Por lo tanto, make-rat y cons son nombres para el mismo constructor primitivo.

Definir selectores y constructores de esta manera es eficiente: En lugar de que make-rat *llame* a cons , make-rat *es* cons , por lo que solo se llama a un procedimiento, no dos, cuando se llama a make-rat . Por otro lado, hacer esto anula las ayudas de depuración que rastrean las llamadas a procedimientos o colocan puntos de interrupción en las llamadas a procedimientos: Es posible que desee observar cómo se llama a make-rat , pero ciertamente no desea observar cada llamada a cons .

Hemos optado por no utilizar este estilo de definición en este libro.

[4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_136) La visualización es la primitiva de Scheme para imprimir datos. La nueva línea de la primitiva de Scheme inicia una nueva línea para imprimir.Ninguno de estos procedimientos devuelve un valor útil, por lo que en los usos de print-rat a continuación, mostramos solo lo que print-rat imprime, no lo que el intérprete imprime como valor devuelto por print-rat .

[5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_140) Sorprendentemente, esta idea es muy difícil de formular rigurosamente. Hay dos enfoques para dar esa formulación. Uno, iniciado porCAR Hoare (1972), es conocido como el método de*modelos abstractos* . Formaliza la especificación de ``procedimientos más condiciones'' como se describe en el ejemplo de número racional anterior. Nótese que la condición sobre la representación de números racionales se enunció en términos de hechos sobre números enteros (igualdad y división). En general, los modelos abstractos definen nuevos tipos de objetos de datos en términos de tipos de objetos de datos previamente definidos. Por lo tanto, las afirmaciones sobre objetos de datos se pueden verificar reduciéndolas a afirmaciones sobre objetos de datos previamente definidos. Otro enfoque, introducido porZilles en el MIT, porGoguen,Thatcher,Wagner, yWright en IBM (véase Thatcher, Wagner y Wright 1978), y porGuttag en Toronto (ver Guttag 1977), se llama*Especificación algebraica* . Considera los «procedimientos» como elementos de un sistema algebraico abstracto cuyo comportamiento está especificado por axiomas que corresponden a nuestras «condiciones», y utiliza las técnicas del álgebra abstracta para comprobar las afirmaciones sobre los objetos de datos. Ambos métodos se examinan en el artículo deLiskov y Zilles (1975).

[**2.2 Datos jerárquicos y la propiedad de cierre**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_2.2)

Como hemos visto, los pares proporcionan un "pegamento" primitivo que podemos utilizar para construir objetos de datos compuestos. La figura  [2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.2) muestra una forma estándar de visualizar unpar - en este caso, el par formado por (cons 1 2) . En esta representación, que se llama*Notación de caja y puntero* , cada objeto se muestra como un*puntero* a un cuadro. El cuadro de un objeto primitivo contiene una representación del objeto. Por ejemplo, el cuadro de un número contiene un numeral. El cuadro de un par es en realidad un cuadro doble, la parte izquierda contiene (un puntero a) el coche del par y la parte derecha contiene el cdr .

Ya hemos visto que los cons se pueden utilizar para combinar no sólo números sino también pares. (Usted hizo uso de este hecho, o debería haberlo hecho, al realizar los ejercicios  [2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.2) y  [2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.3) ). Como consecuencia, los pares proporcionan un bloque de construcción universal a partir del cual podemos construir todo tipo de estructuras de datos. La Figura  [2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.3) muestra dos formas de utilizar pares para combinar los números 1, 2, 3 y 4.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.2:**   Representación de caja y puntero de (cons 1 2) . |

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.3:**   Dos formas de combinar 1, 2, 3 y 4 usando pares. |

La capacidad de crear pares cuyos elementos son pares es la esencia de la importancia de la estructura de lista como herramienta de representación. Nos referimos a esta capacidad como la*propiedad de cierre* de cons . En general, una operación para combinar objetos de datos satisface la propiedad de cierre si los resultados de combinar cosas con esa operación pueden combinarse mediante la misma operación. [6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_154) El cierre es la clave del poder en cualquier medio de combinación porque nos permite crear*estructuras jerárquicas* : estructuras formadas por partes, que a su vez están formadas por partes, y así sucesivamente.

Desde el comienzo del capítulo 1, hemos hecho un uso esencial del cierre al tratar con procedimientos, porque todos los programas, salvo los más simples, se basan en el hecho de que los elementos de una combinación pueden ser combinaciones en sí mismos. En esta sección, abordamos las consecuencias del cierre para datos compuestos. Describimos algunas técnicas convencionales para usar pares para representar secuencias y árboles, y mostramos un lenguaje gráfico que ilustra el cierre de una manera vívida. [7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_155)

**[2.2.1 Representación de secuencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.2.1)**

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.4:**   La secuencia 1, 2, 3, 4 representada como una cadena de pares. |

Una de las estructuras útiles que podemos construir con pares es una*secuencia* : una colección ordenada de objetos de datos. Por supuesto, existen muchas formas de representar secuencias en términos de pares. Una representación particularmente sencilla se ilustra en la figura  [2.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.4) , donde la secuencia 1, 2, 3, 4 se representa como una cadena de pares. El car de cada par es el elemento correspondiente en la cadena, y el cdr del par es el siguiente par en la cadena. El cdr del par final señala el final de la secuencia al señalar un valor distinguido que no es un par, representado en diagramas de caja y puntero como una línea diagonal.y en programas como el valor de la variablenil . La secuencia completa se construye mediante operaciones cons anidadas :

(contras 1   
      (contras 2   
            (contras 3   
                  (contras 4 nula))))

Una secuencia de pares de este tipo, formada por cons anidados , se denomina*lista* , y Scheme proporciona una primitiva llamadalista para ayudar a construir listas. [8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_156) La secuencia anterior podría producirse mediante (lista 1 2 3 4) . En general,

(lista < *a 1* > < *a 2* >  ... < *a n* >)

es equivalente a

(cons < *a 1* > (cons < *a 2* > (cons  ... (cons < *a n* > nulo)  ... )))

Los sistemas Lisp imprimen convencionalmente listas imprimiendo la secuencia deelementos, encerrados entre paréntesis. Por lo tanto, el objeto de datos en la figura  [2.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.4) se imprime como (1 2 3 4) :

(define del uno al cuatro (enumera 1 2 3 4))   
  
del uno al cuatro   
*(1 2 3 4)*

Tenga cuidado de no confundir la expresión (lista 1 2 3 4) con la lista (1 2 3 4) , que es el resultado obtenido cuando se evalúa la expresión. Intentar evaluar la expresión (1 2 3 4) indicará un error cuando el intérprete intente aplicar el procedimiento 1 a los argumentos 2 , 3 y 4 .

Podemos pensar encoche como seleccionar el primer elemento de la lista, y decdr como selección de la sublista que consta de todos los elementos excepto el primero. Las aplicaciones anidadas de car y cdr se pueden utilizar para extraer el segundo, el tercero y los elementos subsiguientes de la lista. [9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_157) El constructorcons hace una lista como la original, pero con un elemento adicional al principio.

(automóvil uno a cuatro)   
*1*  
  
(cdr uno a cuatro)   
*(2 3 4)*  
(automóvil (cdr uno a cuatro))   
*2*  
  
(cons 10 uno a cuatro)   
*(10 1 2 3 4)*  
  
(cons 5 uno a cuatro)   
*(5 1 2 3 4)*

El valor de nil , utilizado para terminar la cadena de pares, puede considerarse como una secuencia sin elementos, el*lista vacía* . La palabra *nil* es una contracción de la palabra latina *nihil* , que significa "nada". [10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_158)

**[Operaciones de lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_159)**

El uso de pares para representar secuencias de elementos como listas se acompaña de técnicas de programación convencionales para manipular listas sucesivamente.`` cdr ing down '' las listas. Por ejemplo, el procedimientolist-ref toma como argumentos una lista y un número *n* y devuelve el *n-* ésimo elemento de la lista. Es habitual numerar los elementos de la lista comenzando por 0. El método para calcular list-ref es el siguiente:

* Para *n* = 0, list-ref debe devolver el automóvil de la lista.
* De lo contrario, list-ref debe devolver el ( *n* - 1)° elemento del cdr de la lista.

(define (lista-ref elementos n)   
  (si (= n 0)   
      (car elementos)   
      (lista-ref (cdr elementos) (- n 1))))   
(define cuadrados (lista 1 4 9 16 25))   
  
(lista-ref cuadrados 3)   
*16*

A menudo, recorremos toda la lista. Para facilitar esto, Scheme incluye un predicado primitivonull?, que prueba si su argumento es la lista vacía. El procedimientolength , que devuelve el número de elementos de una lista, ilustra este patrón típico de uso:

(define (longitud elementos)   
  (si (¿nulo? elementos)   
      0   
      (+ 1 (longitud (cdr elementos)))))   
(define probabilidades (lista 1 3 5 7))   
  
(longitud probabilidades)   
*4*

El procedimiento de longitud implementa un plan recursivo simple. El paso de reducción es:

* La longitud de cualquier lista es 1 más la longitud del cdr de la lista.

Esto se aplica sucesivamente hasta llegar al caso base:

* La longitud de la lista vacía es 0.

También podríamos calcular la longitud en un estilo iterativo:

(define (longitud elementos)   
  (define (longitud-iter un recuento)   
    (si (nulo? a)   
        recuento   
        (longitud-iter (cdr a) (+ 1 recuento))))   
  (longitud-iter elementos 0))

Otra técnica de programación convencional es`` cons up '' una lista de respuestas mientras cdr ing down una lista, como en el procedimientoappend , que toma dos listas como argumentos y combina sus elementos para formar una nueva lista:

(añadir cuadrados de probabilidades)   
*(1 4 9 16 25 1 3 5 7)*  
  
(añadir cuadrados de probabilidades)   
*(1 3 5 7 1 4 9 16 25)*

La función de anexar también se implementa mediante un plan recursivo. Para anexar las listas list1 y list2 , haga lo siguiente:

* Si list1 es la lista vacía, entonces el resultado es simplemente list2 .
* De lo contrario, agregue el cdr de list1 y list2 , y cons el car de list1 al resultado:

(define (añadir lista1 lista2)   
  (si (nulo? lista1)   
      lista2   
      (cons (auto lista1) (añadir (cdr lista1) lista2))))

**Ejercicio 2.17.**   Definir un procedimientoúltimo par que devuelve la lista que contiene solo el último elemento de una lista dada (no vacía):

(último par (lista 23 72 149 34))   
*(34)*

**Ejercicio 2.18.**   Definir un procedimientoreverse que toma una lista como argumento y devuelve una lista de los mismos elementos en orden inverso:

(revertir (lista 1 4 9 16 25))   
*(25 16 9 4 1)*

**Ejercicio 2.19.**   Consideremos elPrograma de conteo de cambios de la sección  [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) . Sería bueno poder cambiar fácilmente la moneda que utiliza el programa, de modo que pudiéramos calcular la cantidad de formas de cambiar una libra esterlina, por ejemplo. Tal como está escrito el programa, el conocimiento de la moneda se distribuye en parte en el procedimiento first-denomination y en parte en el procedimiento count-change (que sabe que hay cinco tipos de monedas estadounidenses). Sería mejor poder proporcionar una lista de monedas que se utilizarán para hacer el cambio.

Queremos reescribir el procedimiento cc de modo que su segundo argumento sea una lista de los valores de las monedas que se van a utilizar en lugar de un entero que especifique qué monedas se van a utilizar. De este modo, podríamos tener listas que definieran cada tipo de moneda:

(define monedas de ee.uu. (lista 50 25 10 5 1))   
(define monedas del reino unido (lista 100 50 20 10 5 2 1 0,5))

Podríamos entonces llamar a cc de la siguiente manera:

(cc 100 monedas estadounidenses)   
*292*

Para ello, será necesario modificar un poco el programa cc . Seguirá teniendo el mismo formato, pero accederá a su segundo argumento de forma diferente, como se muestra a continuación:

(define (cc cantidad valores-monedas)   
  (cond ((= cantidad 0) 1)   
        ((o (< cantidad 0) (¿no más? valores-monedas)) 0)   
        (sino   
         (+ (cc cantidad   
                (excepto-primera-denominación valores-monedas))   
            (cc (- cantidad   
                   (primera-denominación valores-monedas))   
                valores-monedas)))))

Defina los procedimientos first-denomination , except-first-denomination y no-more? en términos de operaciones primitivas sobre estructuras de lista. ¿El orden de los valores de moneda de la lista afecta la respuesta producida por cc ? ¿Por qué o por qué no?

**Ejercicio 2.20.**  Los procedimientos + , \* y list toman una cantidad arbitraria de argumentos. Una forma de definir dichos procedimientos es usar define con *notación de cola punteada* . En la definición de un procedimiento, una lista de parámetros que tiene un punto antes del último nombre de parámetro indica que, cuando se llama al procedimiento, los parámetros iniciales (si los hay) tendrán como valores los argumentos iniciales, como es habitual, pero el valor del parámetro final será una *lista* de los argumentos restantes. Por ejemplo, dada la definición

(define (f x y . z)  *<cuerpo>* )

El procedimiento f puede ser llamado con dos o más argumentos. Si evaluamos

(1 2 3 4 5 6)

entonces en el cuerpo de f , x será 1, y será 2 y z será la lista (3 4 5 6) . Dada la definición

(define (g . w)  *<cuerpo>* )

El procedimiento g puede ser llamado con cero o más argumentos. Si evaluamos

(g1 2 3 4 5 6)

entonces en el cuerpo de g , w será la lista (1 2 3 4 5 6) . [11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_164)

Utilice esta notación para escribir un procedimiento de paridad igual que tome uno o más números enteros y devuelva una lista de todos los argumentos que tengan la misma paridad par-impar que el primer argumento. Por ejemplo,

(misma paridad 1 2 3 4 5 6 7)   
*(1 3 5 7)*  
  
(misma paridad 2 3 4 5 6 7)   
*(2 4 6)*

**[Mapeo sobre listas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_165)**

Una operación sumamente útil es aplicar alguna transformación a cada elemento de una lista y generar la lista de resultados. Por ejemplo, el siguiente procedimiento escala cada número de una lista según un factor determinado:

(define (factor de elementos de lista de escala)   
  (si (¿nulo? elementos)   
      nil   
      (cons (\* (elementos de automóvil) factor)   
            (factor de lista de escala (elementos de cdr))))   
(lista de escala (lista 1 2 3 4 5) 10)   
*(10 20 30 40 50)*

Podemos abstraer esta idea general y capturarla como un patrón común expresado como un procedimiento de orden superior, tal como en la sección  [1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3) . El procedimiento de orden superior aquí se llama map . Map toma como argumentos un procedimiento de un argumento y una lista, y devuelve una lista de los resultados producidos al aplicar el procedimiento a cada elemento de la lista: [12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_166)

(define (map proc items)   
  (if (null? items)   
      nil   
      (cons (proc (car items))   
            (map proc (cdr items)))))   
(map abs (list -10 2.5 -11.6 17))   
*(10 2.5 11.6 17)*  
(map (lambda (x) (\* x x))   
     (list 1 2 3 4))   
*(1 4 9 16)*

Ahora podemos dar una nueva definición de lista de escala en términos de mapa :

(define (factor de elementos de la lista de escala)   
  (mapa (lambda (x) (\* factor x))   
       elementos))

Map es una construcción importante, no sólo porque captura un patrón común, sino porque establece un nivel más alto de abstracción al tratar con listas. En la definición original de scale-list , la estructura recursiva del programa llama la atención sobre el procesamiento elemento por elemento de la lista. Definir scale-list en términos de map suprime ese nivel de detalle y enfatiza que el escalamiento transforma una lista de elementos en una lista de resultados. La diferencia entre las dos definiciones no es que la computadora esté realizando un proceso diferente (no lo está haciendo), sino que pensamos en el proceso de manera diferente. En efecto, map ayuda a establecer una barrera de abstracción que aísla la implementación de procedimientos que transforman listas de los detalles de cómo se extraen y combinan los elementos de la lista. Al igual que las barreras que se muestran en la figura  [2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.1) , esta abstracción nos da la flexibilidad de cambiar los detalles de bajo nivel de cómo se implementan las secuencias, al mismo tiempo que se preserva el marco conceptual de las operaciones que transforman secuencias en secuencias. La Sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) amplía este uso de secuencias como marco para organizar programas.

**Ejercicio 2.21.**   El procedimiento lista-cuadrados toma una lista de números como argumento y devuelve una lista de los cuadrados de esos números.

(lista cuadrada (lista 1 2 3 4))   
*(1 4 9 16)*

A continuación se muestran dos definiciones diferentes de lista cuadrada . Complete ambas con las expresiones que faltan:

(define (elementos de la lista cuadrada)   
  (si (elementos nulos)   
      nil   
      (cons < *??* > < *??* >)))   
(define (elementos de la lista cuadrada)   
  (mapa < *??* > < *??* >))

**Ejercicio 2.22.**   Louis Reasoner intenta reescribir el primer procedimiento de  lista cuadrada del ejercicio [2.21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.21) de modo que desarrolle un proceso iterativo:

(define (cuadrado-lista elementos)   
  (define (iter cosas respuesta)   
    (si (¿nulo? cosas)   
        respuesta   
        (iter (cdr cosas)    
              (cons (cuadrado (auto cosas))   
                    respuesta))))   
  (iter elementos nulo))

Lamentablemente, definir la lista cuadrada de esta manera produce la lista de respuestas en el orden inverso al deseado. ¿Por qué?

Luego Louis intenta solucionar su error intercambiando los argumentos de cons :

(define (cuadrado-lista elementos)   
  (define (iter cosas respuesta)   
    (si (¿nulo? cosas)   
        respuesta   
        (iter (cdr cosas)   
              (cons respuesta   
                    (cuadrado (auto cosas))))))   
  (iter elementos nulo))

Esto tampoco funciona. Explícamelo.

**Ejercicio 2.23.**   El procedimientofor-each es similar a map . Toma como argumentos un procedimiento y una lista de elementos. Sin embargo, en lugar de formar una lista de resultados, for-each simplemente aplica el procedimiento a cada uno de los elementos por turno, de izquierda a derecha. Los valores devueltos al aplicar el procedimiento a los elementos no se utilizan en absoluto; for-each se utiliza con procedimientos que realizan una acción, como imprimir. Por ejemplo,

(para cada (lambda (x) (nueva línea) (mostrar x))   
          (lista 57 321 88))   
*57*   
*321*   
*88*

El valor devuelto por la llamada a for-each (no ilustrado arriba) puede ser algo arbitrario, como verdadero. Proporcione una implementación de for-each .

**[2.2.2 Estructuras jerárquicas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.2.2)**

La representación de secuencias en términos de listas se generaliza naturalmente para representar secuencias cuyos elementos pueden ser secuencias en sí mismos. Por ejemplo, podemos considerar el objeto ((1 2) 3 4) construido por

(contras (lista 1 2) (lista 3 4))

como una lista de tres elementos, el primero de los cuales es en sí mismo una lista, (1 2) . De hecho, esto lo sugiere la forma en que el intérprete imprime el resultado. La figura  [2.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.5) muestra la representación de esta estructura en términos de pares.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.5:**   Estructura formada por (cons (lista 1 2) (lista 3 4)) . |

Otra forma de pensar en las secuencias cuyos elementos son secuencias es como *árboles* . Los elementos de la secuencia son las ramas del árbol, y los elementos que son secuencias en sí mismos son subárboles. La figura  [2.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.6) muestra la estructura de la figura  [2.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.5) vista como un árbol.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.6:**   La estructura de lista en la figura  [2.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.5) vista como un árbol. |

La recursión es una herramienta natural para tratar con estructuras de árbol, ya que a menudo podemos reducir las operaciones sobre los árboles a operaciones sobre sus ramas, que a su vez se reducen a operaciones sobre las ramas de las ramas, y así sucesivamente, hasta llegar a las hojas del árbol. Como ejemplo, compare el procedimiento  de longitud de la sección [2.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.1) con elprocedimiento count-leaves , que devuelve el número total de hojas de un árbol:

(define x (cons (lista 1 2) (lista 3 4)))   
  
(longitud x)   
*3*  
(contar-hojas x)   
*4*  
  
(lista x x)   
*(((1 2) 3 4) ((1 2) 3 4))*  
  
(longitud (lista x x))   
*2*  
  
(contar-hojas (lista x x))   
*8*

Para implementar count-leaves , recuerde el plan recursivo para calcular length :

* La longitud de una lista x es 1 más la longitud del cdr de x .
* La longitud de la lista vacía es 0.

Count-leaves es similar. El valor de la lista vacía es el mismo:

* El número de hojas de la lista vacía es 0.

Pero en el paso de reducción, donde eliminamos el coche de la lista, debemos tener en cuenta que el coche puede ser en sí mismo un árbol cuyas hojas necesitamos contar. Por lo tanto, el paso de reducción adecuado es

* El conteo de hojas de un árbol x es el conteo de hojas del árbol de x más el conteo de hojas del cdr de x .

Finalmente, al tomar el carro s llegamos a las hojas reales, por lo que necesitamos otro caso base:

* El número de hojas de una hoja es 1.

Para ayudar a escribir procedimientos recursivos en árboles, Scheme proporciona el predicado primitivo¿par?, que prueba si su argumento es un par. Aquí está el procedimiento completo: [13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_170)

(define (count-leaves x)   
  (cond ((null? x) 0)     
        ((not (pair? x)) 1)   
        (else (+ (count-leaves (car x))   
                 (count-leaves (cdr x))))))

**Ejercicio 2.24.**   Supongamos que evaluamos la expresión (lista 1 (lista 2 (lista 3 4))) . Proporcione el resultado impreso por el intérprete, la estructura de caja y puntero correspondiente y la interpretación de este como un árbol (como en la figura  [2.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.6) ).

**Ejercicio 2.25.**   Dar combinaciones de carros y cdr que permitan elegir 7 de cada una de las siguientes listas:

(1 3 (5 7) 9)   
  
((7))   
  
(1 (2 (3 (4 (5 (6 7))))))

**Ejercicio 2.26.**   Supongamos que definimos x e y como dos listas:

(define x (lista 1 2 3))   
(define y (lista 4 5 6))

¿Qué resultado imprime el intérprete en respuesta a evaluar cada una de las siguientes expresiones:

(añadir x y)   
  
(cons x y)   
  
(lista x y)

**Ejercicio 2.27.**   Modifique el procedimiento inverso del ejercicio  [2.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.18) para producir unprocedimiento de inversión profunda que toma una lista como argumento y devuelve como valor la lista con sus elementos invertidos y con todas las sublistas también invertidas. Por ejemplo,

(define x (lista (lista 1 2) (lista 3 4)))   
  
x   
*((1 2) (3 4))*  
  
(x inversa)   
*((3 4) (1 2))*  
  
(x inversa profunda)   
*((4 3) (2 1))*

**Ejercicio 2.28.**   Redactar un procedimientofranja que toma como argumento un árbol (representado como una lista) y devuelve una lista cuyos elementos son todas las hojas del árbol ordenadas de izquierda a derecha. Por ejemplo,

(define x (lista (lista 1 2) (lista 3 4)))   
  
(franja x)   
*(1 2 3 4)*  
  
(franja (lista x x))   
*(1 2 3 4 1 2 3 4)*

**Ejercicio 2.29.**  Un móvil binario consta de dos ramas, una rama izquierda y una rama derecha. Cada rama es una varilla de una longitud determinada, de la que cuelga un peso u otro móvil binario. Podemos representar un móvil binario utilizando datos compuestos construyéndolo a partir de dos ramas (por ejemplo, utilizando list ):

(define (make-mobile izquierda derecha)   
  (lista izquierda derecha))

Una rama se construye a partir de una longitud (que debe ser un número) junto con una estructura , que puede ser un número (que representa un peso simple) u otro móvil:

(define (estructura de longitud de rama de creación)   
  (estructura de longitud de lista))

a. Escriba los selectores correspondientes left-branch y right-branch , que devuelven las ramas de un móvil, y branch-length y branch-structure , que devuelven los componentes de una rama.

b. Utilizando sus selectores, defina un procedimiento total-weight que devuelva el peso total de un móvil.

c. Se dice que un móvil es*equilibrado* si el par aplicado por su rama superior izquierda es igual al aplicado por su rama superior derecha (es decir, si la longitud de la varilla izquierda multiplicada por el peso que cuelga de esa varilla es igual al producto correspondiente para el lado derecho) y si cada uno de los submóviles que cuelgan de sus ramas está equilibrado. Diseñe un predicado que pruebe si un móvil binario está equilibrado.

d. Supongamos que cambiamos la representación de los móviles de modo que los constructores sean

(define (make-mobile izquierda derecha)   
  (cons izquierda derecha))   
(define (make-branch longitud estructura)   
  (cons longitud estructura))

¿Cuánto necesitas cambiar tus programas para convertirlos a la nueva representación?

**[Mapeo sobre árboles](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_177)**

Así como map es una poderosa abstracción para trabajar con secuencias, map junto con recursion es una poderosa abstracción para trabajar con árboles. Por ejemplo, el procedimiento scale-tree , análogo a scale-list de la sección  [2.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.1) , toma como argumentos un factor numérico y un árbol cuyas hojas son números. Devuelve un árbol de la misma forma, donde cada número se multiplica por el factor. El plan recursivo para scale-tree es similar al de count-leaves :

(define (scale-tree tree factor)   
  (cond ((null? tree) nil)   
        ((not (pair? tree)) (\* tree factor))   
        (else (cons (scale-tree (car tree) factor)   
                    (scale-tree (cdr tree) factor)))))   
(scale-tree (lista 1 (lista 2 (lista 3 4) 5) (lista 6 7))   
            10)   
*(10 (20 (30 40) 50) (60 70))*

Otra forma de implementar el árbol de escala es considerar el árbol como una secuencia de subárboles y utilizar map . Mapeamos la secuencia, escalando cada subárbol a su vez, y devolvemos la lista de resultados. En el caso base, donde el árbol es una hoja, simplemente multiplicamos por el factor:

(define (escala-árbol árbol factor)   
  (mapa (lambda (sub-árbol)   
         (si (par? sub-árbol)   
             (escala-árbol sub-árbol factor)   
             (\* sub-árbol factor)))   
       árbol))

Muchas operaciones de árbol se pueden implementar mediante combinaciones similares de operaciones de secuencia y recursión.

**Ejercicio 2.30.**   Defina un procedimiento square-tree análogo al procedimiento square-list del ejercicio  [2.21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.21) . Es decir, square-list debería comportarse de la siguiente manera:

(árbol cuadrado   
 (lista 1   
       (lista 2 (lista 3 4) 5)   
       (lista 6 7)))   
*(1 (4 (9 16) 25) (36 49))*

Defina el árbol cuadrado tanto directamente (es decir, sin utilizar ningún procedimiento de orden superior) como también utilizando mapa y recursión.

**Ejercicio 2.31.**   Resuma su respuesta al ejercicio  [2.30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.30) para producir un procedimientoMapa de árbol con la propiedad de que el árbol cuadrado podría definirse como

(define (árbol cuadrado árbol) (árbol-mapa árbol cuadrado))

**Ejercicio 2.32.**   Podemos representar unaconjunto como una lista de elementos distintos, y podemos representar el conjunto de todos los subconjuntos del conjunto como una lista de listas. Por ejemplo, si el conjunto es (1 2 3) , entonces el conjunto de todos los subconjuntos es (() (3) (2) (2 3) (1) (1 3) (1 2) (1 2 3)) . Complete la siguiente definición de un procedimiento que genera el conjunto de subconjuntos de un conjunto y dé una explicación clara de por qué funciona:

(define (subconjuntos s)   
  (si (nulo? s)   
      (lista nula)   
      (let ((resto (subconjuntos (cdr s))))   
        (añade resto (mapa < *??* > resto)))))

**[2.2.3 Secuencias como interfaces convencionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.2.3)**

Al trabajar con datos compuestos, hemos destacado cómo la abstracción de datos nos permite diseñar programas sin enredarnos en los detalles de las representaciones de datos y cómo la abstracción nos preserva la flexibilidad para experimentar con representaciones alternativas. En esta sección, presentamos otro principio de diseño poderoso para trabajar con estructuras de datos: el uso de *interfaces convencionales* .

En la sección  [1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3) vimos cómo las abstracciones de programas, implementadas como procedimientos de orden superior, pueden capturar patrones comunes en programas que tratan con datos numéricos. Nuestra capacidad para formular operaciones análogas para trabajar con datos compuestos depende fundamentalmente del estilo en el que manipulemos nuestras estructuras de datos. Consideremos, por ejemplo, el siguiente procedimiento, análogo al procedimiento count-leaves de la sección  [2.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.2) , que toma un árbol como argumento y calcula la suma de los cuadrados de las hojas que son impares:

(define (árbol de suma de cuadrados impares)   
  (cond ((¿árbol nulo?) 0)     
        ((no (¿árbol de par?))   
         (si (¿árbol impar?) (árbol de cuadrados) 0))   
        (sino (+ (árbol de suma de cuadrados impares (árbol de coche))   
                 (árbol de suma de cuadrados impares (árbol de cdr))))))

A primera vista, este procedimiento es muy diferente del siguiente, que construye una lista de todos los números pares de Fibonacci *F i b* ( *k* ), donde *k* es menor o igual que un entero dado *n* :

(define (fibras pares n)   
  (define (siguiente k)   
    (si (> k n)   
        nulo   
        (sea ((f (fibra k)))   
          (si (¿par? f)   
              (cons f (siguiente (+ k 1)))   
              (siguiente (+ k 1))))))   
  (siguiente 0))

A pesar de que estos dos procedimientos son estructuralmente muy diferentes, una descripción más abstracta de los dos cálculos revela una gran similitud. El primer programa

* enumera las hojas de un árbol;
* los filtra, seleccionando los impares;
* cuadra cada uno de los seleccionados; y
* acumula los resultados usando + , comenzando con 0.

El segundo programa

* enumera los números enteros de 0 a *n* ;
* calcula el número de Fibonacci para cada entero;
* los filtra, seleccionando los pares; y
* acumula los resultados usando cons , comenzando con la lista vacía.

A un ingeniero de procesamiento de señales le resultaría natural conceptualizar estos procesos en términos de señales que fluyen a través de una cascada de etapas, cada una de las cuales implementa parte del plan del programa, como se muestra en la figura  [2.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.7) . En la suma de cuadrados impares , comenzamos con un*enumerador* , que genera una ``señal'' que consiste en las hojas de un árbol determinado. Esta señal se pasa a través de un*filtro* , que elimina todos los elementos excepto los impares. La señal resultante pasa a su vez a través de un*mapa* , que es un "transductor" que aplica el procedimiento cuadrado a cada elemento. La salida del mapa se envía a un*acumulador* , que combina los elementos usando + , comenzando desde un 0 inicial. El plan para las fibras pares es análogo.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.7:**   Los planes de flujo de señales para los procedimientos suma de cuadrados impares (arriba) y fibras pares (abajo) revelan los puntos en común entre los dos programas. |

Lamentablemente, las dos definiciones de procedimiento anteriores no muestran esta estructura de flujo de señales. Por ejemplo, si examinamos el procedimiento de suma de cuadrados impares , encontramos que la enumeración se implementa en parte mediante las pruebas de nulo y de par y en parte mediante la estructura recursiva de árbol del procedimiento. De manera similar, la acumulación se encuentra en parte en las pruebas y en parte en la adición utilizada en la recursión. En general, no hay partes distintas de ninguno de los procedimientos que correspondan a los elementos en la descripción del flujo de señales. Nuestros dos procedimientos descomponen los cálculos de una manera diferente, extendiendo la enumeración sobre el programa y mezclándola con el mapa, el filtro y la acumulación. Si pudiéramos organizar nuestros programas para hacer que la estructura de flujo de señales se manifieste en los procedimientos que escribimos, esto aumentaría la claridad conceptual del código resultante.

**[Operaciones de secuencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_181)**

La clave para organizar los programas de forma que reflejen más claramente la estructura del flujo de señales es concentrarse en las «señales» que fluyen de una etapa del proceso a la siguiente. Si representamos estas señales como listas, podemos utilizar operaciones de listas para implementar el procesamiento en cada una de las etapas. Por ejemplo, podemos implementar las etapas de mapeo de los diagramas de flujo de señales utilizando el procedimiento  de mapeo de la sección [2.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.1) :

(mapa cuadrado (lista 1 2 3 4 5))   
*(1 4 9 16 25)*

El filtrado de una secuencia para seleccionar solo aquellos elementos que satisfacen un predicado dado se logra mediante

(define (filtrar predicado secuencia)   
  (cond ((null? secuencia) nil)   
        ((predicado (car secuencia))   
         (cons (car secuencia)   
               (filtrar predicado (cdr secuencia))))   
        (else (filtrar predicado (cdr secuencia)))))

Por ejemplo,

(filtrar impar? (lista 1 2 3 4 5))   
*(1 3 5)*

Las acumulaciones se pueden implementar mediante

(define (acumular op secuencia inicial)   
  (si (¿nulo? secuencia)   
      inicial   
      (op (secuencia de carro)   
          (acumular op inicial (secuencia de cdr))))   
(acumular + 0 (lista 1 2 3 4 5))   
*15*  
(acumular \* 1 (lista 1 2 3 4 5))   
*120*  
(acumular cons nil (lista 1 2 3 4 5))   
*(1 2 3 4 5)*

Todo lo que queda para implementar diagramas de flujo de señales es enumerar la secuencia de elementos que se van a procesar. Para los números pares , necesitamos generar la secuencia de números enteros en un rango determinado, lo que podemos hacer de la siguiente manera:

(define (intervalo-de-enumeración bajo alto)   
  (si (> bajo alto)   
      nulo   
      (cons bajo (intervalo-de-enumeración (+ bajo 1) alto))))   
(intervalo-de-enumeración 2 7)   
*(2 3 4 5 6 7)*

Para enumerar las hojas de un árbol, podemos utilizar [14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_182)

(define (enumerate-tree tree)   
  (cond ((null? tree) nil)   
        ((not (pair? tree)) (list tree))   
        (else (append (enumerate-tree (car tree))   
                      (enumerate-tree (cdr tree))))))   
(enumerate-tree (lista 1 (lista 2 (lista 3 4)) 5))   
*(1 2 3 4 5)*

Ahora podemos reformular suma de cuadrados impares y pares-fibras como en los diagramas de flujo de señales. Para suma de cuadrados impares , enumeramos la secuencia de hojas del árbol, la filtramos para conservar solo los números impares en la secuencia, elevamos al cuadrado cada elemento y sumamos los resultados:

(define (suma-impar-cuadrados árbol)   
  (acumular +   
              0   
              (mapa cuadrado   
                   (filtrar impar?   
                           (enumerar-árbol árbol)))))

Para los números pares , enumeramos los números enteros de 0 a *n* , generamos el número de Fibonacci para cada uno de estos números enteros, filtramos la secuencia resultante para mantener solo los elementos pares y acumulamos los resultados en una lista:

(define (fibras-pares n)   
  (acumula cons   
              nil   
              (filtra par?   
                      (mapa fib   
                           (enumerar-intervalo 0 n)))))

El valor de expresar los programas como operaciones de secuencia es que nos ayuda a crear diseños de programas modulares, es decir, diseños que se construyen combinando piezas relativamente independientes. Podemos fomentar el diseño modular proporcionando una biblioteca de componentes estándar junto con una interfaz convencional para conectar los componentes de formas flexibles.

La construcción modular es una estrategia poderosa para controlar la complejidad en el diseño de ingeniería. En aplicaciones de procesamiento de señales reales, por ejemplo, los diseñadores construyen sistemas regularmente conectando en cascada elementos seleccionados de familias estandarizadas de filtros y transductores. De manera similar, las operaciones de secuencia proporcionan una biblioteca de elementos de programa estándar que podemos mezclar y combinar. Por ejemplo, podemos reutilizar partes de los procedimientos de suma de cuadrados impares y pares en un programa que construya una lista de los cuadrados de los primeros *n* + 1 números de Fibonacci:

(define (lista-fib-cuadrados n)   
  (acumula cons   
              nil   
              (mapa cuadrado   
                   (mapa fib   
                        (enumerar-intervalo 0 n)))))   
(lista-fib-cuadrados 10)   
*(0 1 1 4 9 25 64 169 441 1156 3025)*

Podemos reorganizar las piezas y usarlas para calcular el producto de los números enteros impares en una secuencia:

(define (secuencia producto-de-cuadrados-de-elementos-impares)   
  (acumular \*   
              1   
              (mapa cuadrado   
                   (filtrar secuencia impar))))   
(producto-de-cuadrados-de-elementos-impares (lista 1 2 3 4 5))   
*225*

También podemos formular aplicaciones de procesamiento de datos convencionales en términos de operaciones de secuencia. Supongamos que tenemos una secuencia de registros de personal y queremos encontrar el salario del programador mejor pagado. Supongamos que tenemos un selector salary que devuelve el salario de un registro y un predicado programmer? que prueba si un registro es para un programador. Entonces podemos escribir

(define (registros de salario del programador mejor pagado)   
  (acumula un máximo de   
              0   
              (mapa de salario   
                   (filtrar registros de programador))))

Estos ejemplos dan sólo una idea de la amplia gama de operaciones que pueden expresarse como operaciones de secuencia. [15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_183)

Las secuencias, implementadas aquí como listas, sirven como una interfaz convencional que nos permite combinar módulos de procesamiento. Además, cuando representamos uniformemente las estructuras como secuencias, hemos localizado las dependencias de la estructura de datos en nuestros programas a un pequeño número de operaciones de secuencia. Al cambiarlas, podemos experimentar con representaciones alternativas de secuencias, mientras dejamos intacto el diseño general de nuestros programas. Aprovecharemos esta capacidad en la sección  [3.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5) , cuando generalicemos el paradigma de procesamiento de secuencias para admitir secuencias infinitas.

**Ejercicio 2.33.**   Complete las expresiones que faltan para completar las siguientes definiciones de algunas operaciones básicas de manipulación de listas como acumulaciones:

(define (map p secuencia)   
  (acumula (lambda (x y) < *??* >) secuencia nula))  
(define (append seq1 seq2)   
  (accumulate cons < *??* > < *??* >))  
(define (secuencia de longitud)   
  (acumular < *??* > 0 secuencia))

**Ejercicio 2.34.**  La evaluación de un polinomio en *x* para un valor dado de *x* se puede formular como una acumulación. Evaluamos el polinomio



utilizando un algoritmo conocido llamado*La regla de Horner* , que estructura el cálculo como



En otras palabras, empezamos con *a n* , multiplicamos por *x* , sumamos *a n* -1 , multiplicamos por *x* , y así sucesivamente, hasta llegar *a* 0 . [16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_186) Complete la siguiente plantilla para producir un procedimiento que evalúe un polinomio utilizando la regla de Horner. Suponga que los coeficientes del polinomio están dispuestos en una secuencia, desde *a* 0 hasta *a n* .

(define (eval-horner x secuencia-de-coeficientes)   
  (acumula (lambda (este-coeficiente términos-superiores) < *??* >)   
              0   
              secuencia-de-coeficientes))

Por ejemplo, para calcular 1 + 3 *x* + 5 *x* 3 + *x* 5 en *x* = 2, evaluarías

(horner-eval 2 (lista 1 3 0 5 0 1))

**Ejercicio 2.35.**   Redefina count-leaves de la sección  [2.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.2) como una acumulación:

(define (count-leaves t)   
  (accumulate < *??* > < *??* > (map < *??* > < *??* >)))

**Ejercicio 2.36.**   El procedimiento acumular-n es similar a acumular excepto que toma como tercer argumento una secuencia de secuencias, que se supone que tienen todas el mismo número de elementos. Aplica el procedimiento de acumulación designado para combinar todos los primeros elementos de las secuencias, todos los segundos elementos de las secuencias, y así sucesivamente, y devuelve una secuencia de los resultados. Por ejemplo, si s es una secuencia que contiene cuatro secuencias, ((1 2 3) (4 5 6) (7 8 9) (10 11 12)), entonces el valor de (acumula-n + 0 s) debe ser la secuencia (22 26 30) . Complete las expresiones que faltan en la siguiente definición de acumular-n :

(define (acumular-n op init seqs)   
  (si (nulo? (auto seqs))   
      nulo   
      (cons (acumular op init < *??* >)   
            (acumular-n op init < *??* >))))

**Ejercicio 2.37.**  Supongamos que representamos los vectores *v* = ( *v i* ) como secuencias de números y las matrices *m* = ( *m i j* ) como secuencias de vectores (las filas de la matriz). Por ejemplo, la matriz

![Forma

El contenido generado por IA puede ser incorrecto.]()

se representa como la sucesión ((1 2 3 4) (4 5 6 6) (6 7 8 9)) . Con esta representación, podemos utilizar operaciones de sucesión para expresar de forma concisa las operaciones básicas con matrices y vectores. Estas operaciones (que se describen en cualquier libro de álgebra matricial) son las siguientes:

Forma

El contenido generado por IA puede ser incorrecto.

Podemos definir el producto escalar como [17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_190)

(define (producto escalar v w)   
  (acumular + 0 (mapa \* v w)))

Complete las expresiones faltantes en los siguientes procedimientos para calcular las demás operaciones matriciales. (El procedimiento acumular-n se define en el ejercicio  [2.36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.36) ).

(define (matriz-\*-vector m v)   
  (mapa < *??* > m))  
(define (transpone mat)   
  (acumula-n < *??* > < *??* > mat))  
(define (matriz-\*-matriz m n)   
  (sea ((cols (transpone n)))   
    (mapa < *??* > m)))

**Ejercicio 2.38.**  El procedimiento de acumulación también se conoce como fold-right , porque combina el primer elemento de la secuencia con el resultado de combinar todos los elementos de la derecha. También existe un fold-left , que es similar a fold-right , excepto que combina elementos que funcionan en la dirección opuesta:

(define (fold-left op secuencia inicial)   
  (define (iter resultado resto)   
    (si (nulo? resto)   
        resultado   
        (iter (op resultado (car resto))   
              (cdr resto))))   
  (iter secuencia inicial))

¿Cuáles son los valores de?

(plegable a la derecha / 1 (lista 1 2 3))   
(plegable a la izquierda / 1 (lista 1 2 3))   
(plegable a la derecha lista nula (lista 1 2 3))   
(plegable a la izquierda lista nula (lista 1 2 3))

Proporcione una propiedad que op debe satisfacer para garantizar que fold-right y fold-left producirán los mismos valores para cualquier secuencia.

**Ejercicio 2.39.**    Complete las siguientes definiciones de inversa (ejercicio  [2.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.18) ) en términos de pliegue a la derecha y pliegue a la izquierda del ejercicio  [2.38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.38) :

(define (secuencia inversa)   
  (fold-right (lambda (x y) < *??* >) secuencia nula))   
(define (secuencia inversa)   
  (fold-left (lambda (x y) < *??* >) secuencia nula))

**[Mapeos anidados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_193)**

Podemos extender el paradigma de secuencias para incluir muchos cálculos que se expresan comúnmente utilizando bucles anidados. [18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_194) Considere este problema: Dado un entero positivo *n* , encuentre todos los pares ordenados de enteros positivos distintos *i* y *j* , donde 1 < *j* < *i* < *n* , tales que *i* + *j* es primo. Por ejemplo, si *n* es 6, entonces los pares son los siguientes:

Forma

El contenido generado por IA puede ser incorrecto.

Una forma natural de organizar este cálculo es generar la secuencia de todos los pares ordenados de números enteros positivos menores o iguales a *n* , filtrar para seleccionar aquellos pares cuya suma sea prima y luego, para cada par ( *i* , *j* ) que pase por el filtro, producir el triple ( *i* , *j* , *i* + *j* ).

Aquí hay una manera de generar la secuencia de pares: Para cada entero *i* < *n* , enumerar los enteros *j* < *i* , y para cada uno de estos *i* y *j* generar el par ( *i* , *j* ). En términos de operaciones de secuencia, mapeamos a lo largo de la secuencia (enumerar-intervalo 1 n) . Para cada *i* en esta secuencia, mapeamos a lo largo de la secuencia (enumerar-intervalo 1 (- i 1)) . Para cada *j* en esta última secuencia, generamos el par (lista ij) . Esto nos da una secuencia de pares para cada *i* . Combinando todas las secuencias para todos los *i* (acumulando con append ) produce la secuencia requerida de pares: [19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_195)

(acumular añadir   
            nulo   
            (mapa (lambda (i)   
                   (mapa (lambda (j) (lista i j))   
                        (intervalo-enumeración 1 (- i 1))))   
                 (intervalo-enumeración 1 n)))

La combinación de mapeo y acumulación con append es tan común en este tipo de programa que la aislaremos como un procedimiento separado:

(define (map plano proc seq)   
  (acumula anexar nulo (map proc seq)))

Ahora, filtre esta secuencia de pares para encontrar aquellos cuya suma sea prima. El predicado de filtro se llama para cada elemento de la secuencia; su argumento es un par y debe extraer los números enteros del par. Por lo tanto, el predicado que se debe aplicar a cada elemento de la secuencia es

(define (par de suma prima)   
  (par de prima (+ (par de autos) (par de cadr))))

Finalmente, genere la secuencia de resultados mapeando sobre los pares filtrados utilizando el siguiente procedimiento, que construye un triple formado por los dos elementos del par junto con su suma:

(define (hacer-par-suma par)   
  (lista (par de autos) (par de cadr) (+ (par de autos) (par de cadr))))

Combinando todos estos pasos se obtiene el procedimiento completo:

(define (pares de suma prima n)   
  (mapa hace-pares-suma   
       (filtrar suma prima?   
               (mapa-plano   
                (lambda (i)   
                  (mapa (lambda (j) (lista i j))   
                       (intervalo-enumeración 1 (- i 1))))   
                (intervalo-enumeración 1 n)))))

Las asignaciones anidadas también son útiles para secuencias distintas de las que enumeran intervalos. Supongamos que deseamos generar todas laspermutaciones de un conjunto *S* ; es decir, todas las formas de ordenar los elementos del conjunto. Por ejemplo, las permutaciones de {1,2,3} son {1,2,3}, { 1,3,2}, {2,1,3}, { 2,3,1}, { 3,1,2} y { 3,2,1}. He aquí un plan para generar las permutaciones de  *S* : Para cada elemento *x* en *S* , generar recursivamente la secuencia de permutaciones de *S* - *x* , [20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_196) y adjuntar *x* al frente de cada uno. Esto produce, para cada *x* en *S* , la secuencia de permutaciones de *S* que comienzan con  *x* . Combinando estas secuencias para todos *los x* se obtienen todas las permutaciones de  *S* : [21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_197)

(define (permutaciones s)   
  (si (¿nulo? s)                     *; conjunto vacío?*  
      (lista nula)                    *; secuencia que contiene el conjunto vacío*  
      (mapa plano (lambda (x)   
                 (mapa (lambda (p) (cons x p))   
                      (permutaciones (eliminar x s))))   
               s)))

Observe cómo esta estrategia reduce el problema de generar permutaciones de *S* al problema de generar las permutaciones de conjuntos con menos elementos que *S* . En el caso terminal, avanzamos hasta la lista vacía, que representa un conjunto sin elementos. Para esto, generamos (list nil) , que es una secuencia con un elemento, es decir, el conjunto sin elementos. El procedimiento remove utilizado en las permutaciones devuelve todos los elementos de una secuencia dada excepto un elemento dado. Esto se puede expresar como un filtro simple:

(definir (eliminar secuencia de elementos)   
  (filtrar (lambda (x) (no (= x elemento)))   
          secuencia))

**Ejercicio 2.40.**   Definir un procedimientopares únicos que, dado un entero *n* , generan la secuencia de pares ( *i* , *j* ) con 1 < *j* < *i* < *n* . Utilice pares únicos para simplificar la definición de pares de suma prima dada anteriormente.

**Ejercicio 2.41.**   Escriba un procedimiento para encontrar todos los triples ordenados de números enteros positivos distintos *i* , *j* y  *k* menores o iguales a un entero dado *n* que sumen un entero dado *s* .

**Ejercicio 2.42.**  

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.8:**   Una solución al rompecabezas de las ocho reinas. |

ElEl ``rompecabezas de las ocho reinas'' pregunta cómo colocar ocho reinas en un tablero de ajedrez de modo que ninguna reina esté en jaque de ninguna otra (es decir, no hay dos reinas en la misma fila, columna o diagonal). Una posible solución se muestra en la figura  [2.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.8) . Una forma de resolver el rompecabezas es trabajar a lo largo del tablero, colocando una reina en cada columna. Una vez que hemos colocado *k* - 1 reinas, debemos colocar la *k* ésima reina en una posición en la que no jaque a ninguna de las reinas que ya están en el tablero. Podemos formular este enfoque de forma recursiva: supongamos que ya hemos generado la secuencia de todas las formas posibles de colocar *k* - 1 reinas en las primeras *k* - 1 columnas del tablero. Para cada una de estas formas, genere un conjunto extendido de posiciones colocando una reina en cada fila de la *k* ésima columna. Ahora filtre estas, manteniendo solo las posiciones para las que la reina en la *k* ésima columna esté segura con respecto a las otras reinas. Esto produce la secuencia de todas las formas de colocar *k* reinas en las primeras *k* columnas. Si continuamos con este proceso, no solo obtendremos una solución, sino todas las soluciones del rompecabezas.

Implementamos esta solución como un procedimiento queens , que devuelve una secuencia de todas las soluciones al problema de colocar *n* reinas en un tablero de ajedrez *de n* Ã— *n* . Queens tiene un procedimiento interno queen-cols que devuelve la secuencia de todas las formas de colocar reinas en las primeras *k* columnas del tablero.

(define (reinas tamaño-tablero)   
  (define (columnas-reina k)     
    (si (= k 0)   
        (lista tablero-vacío)   
        (filtro   
         (lambda (posiciones) (¿seguro? k posiciones))   
         (mapa-plano   
          (lambda (resto-de-reinas)   
            (mapa (lambda (nueva-fila)   
                   (posición-adyacente nueva-fila k resto-de-reinas))   
                 (intervalo-enumerar 1 tamaño-tablero)))   
          (columnas-reina (- k 1))))))   
  (columnas-reina tamaño-tablero))

En este procedimiento, rest-of-queens es una forma de colocar *k* - 1 reinas en las primeras *k* - 1 columnas, y new-row es una fila propuesta en la que colocar la reina para la *k* ésima columna. Complete el programa implementando la representación para conjuntos de posiciones del tablero, incluido el procedimiento adjoin-position , que une una nueva posición de fila-columna a un conjunto de posiciones, y empty-board , que representa un conjunto vacío de posiciones. También debe escribir el procedimiento safe? , que determina para un conjunto de posiciones, si la reina en la *k* ésima columna está segura con respecto a las demás. (Observe que solo necesitamos verificar si la nueva reina está segura; las otras reinas ya están seguras con respecto a las demás).

**Ejercicio 2.43.**   Louis Reasoner está teniendo un momento terrible al realizar el ejercicio  [2.42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.42) . Su procedimiento de reinas parece funcionar, pero se ejecuta extremadamente lento. (Louis nunca logra esperar lo suficiente para que resuelva incluso el caso 6×6). Cuando Louis le pide ayuda a Eva Lu Ator, ella señala que ha intercambiado el orden de las aplicaciones anidadas en el mapa plano , escribiéndolo como

(mapa-plano   
 (lambda (nueva-fila)   
   (mapa (lambda (resto-de-reinas)   
          (posición-adyacente nueva-fila k resto-de-reinas))   
        (columnas-de-reinas (- k 1))))   
 (intervalo-de-enumeración 1 tamaño-de-tablero))

Explique por qué este intercambio hace que el programa se ejecute lentamente. Calcule cuánto tiempo le tomará al programa de Louis resolver el rompecabezas de las ocho reinas, suponiendo que el programa del ejercicio  2.42 [resuelve](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.42) el rompecabezas en el tiempo *T.*

**[2.2.4 Ejemplo: un lenguaje de imágenes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.2.4)**

Esta sección presenta un lenguaje simple para dibujar imágenes que ilustran el poder de la abstracción y el cierre de datos, y también explota procedimientos de orden superior de una manera esencial. El lenguaje está diseñado para facilitar la experimentación con patrones como los de la figura  [2.9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.9) , que están compuestos de elementos repetidos que se desplazan y escalan. [22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_202) En este lenguaje, los objetos de datos que se combinan se representan como procedimientos en lugar de como una estructura de lista. Al igual que cons , que satisface laLa propiedad de cierre nos permite construir fácilmente estructuras de listas arbitrariamente complicadas; las operaciones en este lenguaje, que también satisfacen la propiedad de cierre, nos permiten construir fácilmente patrones arbitrariamente complicados.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto.           Foto en blanco y negro de un grupo de personas  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.9:**   Diseños generados con el lenguaje de imágenes. |

**[El lenguaje de la imagen](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_203)**

Cuando comenzamos nuestro estudio de programación en la sección  [1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1) , enfatizamos la importancia de describir un lenguaje centrándonos en los primitivos del lenguaje, sus medios de combinación y sus medios de abstracción. Seguiremos ese marco aquí.

Parte de la elegancia de este lenguaje pictórico es que sólo hay un tipo de elemento, llamado*pintor* . Un pintor dibuja una imagen que se desplaza y escala para que encaje en un espacio designado.Marco en forma de paralelogramo. Por ejemplo, hay un pintor primitivo al que llamaremos ola que hace un dibujo lineal rudimentario, como se muestra en la figura  [2.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.10) . La forma real del dibujo depende del marco: las cuatro imágenes de la figura  [2.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.10) son producidas por el mismo pintor de olas , pero con respecto a cuatro marcos diferentes. Los pintores pueden ser más elaborados que esto: el pintor primitivo llamado rogers pinta un cuadro del fundador del MIT, William Barton Rogers, como se muestra en la figura  [2.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.11) . [23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_204) Las cuatro imágenes de la figura  [2.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.11) se dibujan con respecto a los mismos cuatro marcos que las imágenes  de olas de la figura [2.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.10) .

Para combinar imágenes, utilizamos varias operaciones que construyen nuevos pintores a partir de pintores dados. Por ejemplo, laLa operación al lado toma dos pintores y produce un nuevo pintor compuesto que dibuja la imagen del primer pintor en la mitad izquierda del marco y la imagen del segundo pintor en la mitad derecha del marco. De manera similar,A continuación se toman dos pintores y se produce un pintor compuesto que dibuja la imagen del primer pintor debajo de la imagen del segundo pintor. Algunas operaciones transforman un solo pintor para producir un nuevo pintor. Por ejemplo,flip-vert toma un pintor y produce un pintor que dibuja su imagen al revés, yflip-horiz produce un pintor que dibuja la imagen del pintor original de izquierda a derecha invertida.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto.           Forma  El contenido generado por IA puede ser incorrecto.  Forma  El contenido generado por IA puede ser incorrecto.           Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.10:**   Imágenes producidas por el pintor de olas , con respecto a cuatro cuadros diferentes. Los cuadros, que se muestran con líneas de puntos, no forman parte de las imágenes. |

|  |
| --- |
| Imagen en blanco y negro de un hombre con barba y bigote  El contenido generado por IA puede ser incorrecto.           Imagen en blanco y negro  El contenido generado por IA puede ser incorrecto.  Imagen que contiene persona, parado, foto, hombre  El contenido generado por IA puede ser incorrecto.           Imagen en blanco y negro de un hombre con la boca abierta  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.11:**   Imágenes de William Barton Rogers, fundador y primer presidente del MIT, pintadas con respecto a los mismos cuatro marcos que en la figura  [2.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.10) (imagen original reimpresa con el permiso del Museo del MIT). |

La figura  [2.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.12) muestra el dibujo de un pintor llamado wave4 que se construye en dos etapas a partir de wave :

(define wave2 (al lado de wave (onda flip-vert)))   
(define wave4 (debajo de wave2 wave2))

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto.           Forma  El contenido generado por IA puede ser incorrecto.  (define wave2 (define wave4    (al lado de wave (onda flip-vert))) (debajo de wave2 wave2)) |
|  |
| **Figura 2.12:**   Creación de una figura compleja, a partir del pintor de olas de la figura  [2.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.10) . |

Al construir una imagen compleja de esta manera, estamos explotando el hecho de que los pintores están cerrados bajo los medios de combinación del lenguaje. El lado o debajo de dos pintores es en sí mismo un pintor; por lo tanto, podemos usarlo como un elemento para hacer pintores más complejos. Al igual que con la construcción de la estructura de lista utilizando cons , el cierre de nuestros datos bajo los medios de combinación es crucial para la capacidad de crear estructuras complejas mientras se utilizan solo unas pocas operaciones.

Una vez que podamos combinar pintores, nos gustaría poder abstraer patrones típicos de combinación de pintores. Implementaremos las operaciones de pintor como procedimientos de Scheme. Esto significa que no necesitamos un mecanismo de abstracción especial en el lenguaje de imágenes: dado que los medios de combinación son procedimientos de Scheme ordinarios, automáticamente tenemos la capacidad de hacer con las operaciones de pintor todo lo que podemos hacer con los procedimientos. Por ejemplo, podemos abstraer el patrón en wave4 como

(define (pintor de pares invertidos)   
  (let ((pintor2 (al lado de pintor (pintor flip-vert))))   
    (debajo de pintor2 pintor2)))

y definir wave4 como una instancia de este patrón:

(define wave4 (onda de pares invertidos))

También podemos definir operaciones recursivas. Aquí hay una que hace que los pintores se dividan y se ramifiquen hacia la derecha, como se muestra en las figuras  [2.13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.13) y   [2.14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.14) :

(define (pintor dividido a la derecha n)   
  (si (= n 0)   
      pintor   
      (deja ((más pequeño (pintor dividido a la derecha (- n 1))))   
        (al lado del pintor (debajo del más pequeño más pequeño)))))

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto.          Imagen en blanco y negro  El contenido generado por IA puede ser incorrecto.       división a la derecha n  *división*                    en esquina  *n* |
|  |
| **Figura 2.13:**   Planes recursivos para división por la derecha y división por las esquinas . |

Podemos producir patrones equilibrados ramificándonos tanto hacia arriba como hacia la derecha (ver ejercicio  [2.44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.44) y figuras  [2.13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.13) y   [2.14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.14) ):

(define (pintor dividido en esquinas n)   
  (si (= n 0)   
      pintor   
      (let ((arriba (pintor dividido en esquinas (- n 1)))   
            (derecha (pintor dividido en esquinas (- n 1))))   
        (let ((arriba-izquierda (al lado de arriba arriba))   
              (abajo-derecha (abajo derecha derecha))   
              (esquina (pintor dividido en esquinas (- n 1))))   
          (al lado de (abajo pintor arriba-izquierda)   
                  (abajo esquina inferior-derecha))))))

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto.           Imagen que contiene parado, persona, foto, hombre  El contenido generado por IA puede ser incorrecto.       (onda 4 dividida a la derecha) (rogers 4 dividida a la derecha)  Forma  El contenido generado por IA puede ser incorrecto.           Imagen en blanco y negro de un grupo de personas posando para una foto  El contenido generado por IA puede ser incorrecto.      (onda 4 dividida en la esquina) (rogers 4 dividida en la esquina) |
|  |
| **Figura 2.14:**   Las operaciones recursivas división por la derecha y división por las esquinas aplicadas a las figuras de los pintores wave y rogers . La combinación de cuatro figuras de división por las esquinas produce diseños  simétricos de límite cuadrado como se muestra en la figura [2.9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.9) . |

Colocando cuatro copias de una división de esquina de manera apropiada, obtenemos un patrón llamado límite cuadrado , cuya aplicación a wave y rogers se muestra en la figura  [2.9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.9) :

(define (pintor de límite cuadrado n)   
  (sea ((cuarto (pintor de división de esquina n)))   
    (sea ((mitad (al lado de (cuarto flip-horiz) cuarto)))   
      (debajo de (mitad flip-vert) mitad))))

**Ejercicio 2.44.**   Definir el procedimientoup-split usado por corner-split . Es similar a right-split , excepto que intercambia los roles de below y side .

**[Operaciones de orden superior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_206)**

Además de abstraer patrones de combinación de pintores, podemos trabajar a un nivel superior, abstrayendo patrones de combinación de operaciones de pintores. Es decir, podemos ver las operaciones de pintores como elementos para manipular y podemos escribir medios de combinación para estos elementos: procedimientos que toman operaciones de pintores como argumentos y crean nuevas operaciones de pintores.

Por ejemplo, flipped-pairs y square-limit organizan cada uno cuatro copias de la imagen de un pintor en un patrón cuadrado; difieren solo en cómo orientan las copias. Una forma de abstraer este patrón de combinación de pintores es con el siguiente procedimiento, que toma cuatro operaciones de pintor de un argumento y produce una operación de pintor que transforma un pintor dado con esas cuatro operaciones y organiza los resultados en un cuadrado. Tl , tr , bl y br son las transformaciones que se deben aplicar a la copia superior izquierda, la copia superior derecha, la copia inferior izquierda y la copia inferior derecha, respectivamente.

(define (cuadrado de cuatro tl tr bl br)   
  (lambda (pintor)   
    (let ((arriba (al lado de (tl pintor) (tr pintor)))   
          (abajo (al lado de (bl pintor) (br pintor))))   
      (debajo abajo arriba))))

Luego, los pares invertidos se pueden definir en términos del cuadrado de cuatro de la siguiente manera: [24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_207)

(define (pintor de pares invertidos)   
  (let ((combine4 (identidad del cuadrado de cuatro identidad flip-vert   
                                  )))   
    (combine4 pintor)))

y el límite cuadrado se puede expresar como [25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_208)

(define (pintor de límite cuadrado n)   
  (let ((combine4 (identidad de giro horizontal del cuadrado de cuatro rotar   
                                  180 grados girar verticalmente)))   
    (combine4 (pintor de división de esquinas n))))

**Ejercicio 2.45.**   La división por la derecha y la división por arriba se pueden expresar como instancias de una operación de división general. Defina un procedimientodividir con la propiedad que se evalúa

(definir división a la derecha (división al lado de abajo))   
(definir división hacia arriba (división debajo al lado))

produce procedimientos right-split y up-split con los mismos comportamientos que los ya definidos.

**[Marcos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_210)**

Antes de que podamos mostrar cómo implementar pintores y sus medios de combinación, primero debemos considerarMarcos. Un marco se puede describir mediante tres vectores: un vector de origen y dos vectores de borde. El vector de origen especifica el desplazamiento del origen del marco con respecto a un origen absoluto en el plano, y los vectores de borde especifican los desplazamientos de las esquinas del marco con respecto a su origen. Si los bordes son perpendiculares, el marco será rectangular. De lo contrario, el marco será un paralelogramo más general.

La figura  [2.15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.15) muestra un marco y sus vectores asociados. De acuerdo con la abstracción de datos, no necesitamos especificar aún cómo se representan los marcos, salvo decir que hay un constructor.make-frame , que toma tres vectores y produce un marco y tres selectores correspondientesmarco de origen ,borde1-marco , yedge2-frame (ver ejercicio  [2.47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.47) ).

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.15:**   Un marco se describe mediante tres vectores: un origen y dos bordes. |

Usaremos coordenadas en el cuadrado unitario (0 < *x* , *y* < 1) para especificar imágenes. Con cada cuadro, asociamos un*Mapa de coordenadas del marco* , que se utilizará para desplazar y escalar imágenes para que se ajusten al marco. El mapa transforma el cuadrado unitario en el marco al asignar el vector ***v*** = ( *x* , *y* ) a la suma de vectores.



Por ejemplo, (0,0) se asigna al origen del marco, (1,1) al vértice diagonalmente opuesto al origen y (0,5,0,5) al centro del marco. Podemos crear el mapa de coordenadas de un marco con el siguiente procedimiento: [26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_211)

(define (marco-coord-mapa marco)   
  (lambda (v)   
    (add-vect   
     (origen-marco marco)   
     (add-vect (escala-vect (xcor-vect v)   
                           (borde1-marco marco))   
               (escala-vect (ycor-vect v)   
                           (borde2-marco marco))))))

Observe que al aplicar frame-coord-map a un marco se devuelve un procedimiento que, dado un vector, devuelve un vector. Si el vector del argumento está en el cuadrado unitario, el vector del resultado estará en el marco. Por ejemplo,

((mapa de coordenadas del marco a) (make-vect 0 0))

devuelve el mismo vector que

(origen-marco-a)

**Ejercicio 2.46.**  Un vector bidimensional **v** que va desde el origen hasta un punto se puede representar como un par formado por una coordenada *x y una coordenada y* . Implemente una abstracción de datos para vectores proporcionando un constructormake-vect y selectores correspondientesxcor-vect yycor-vect . En términos de sus selectores y constructores, implemente procedimientosañadir-vect ,sub-vecto , yscale-vect que realiza las operaciones de suma de vectores, resta de vectores y multiplicación de un vector por un escalar:

Forma

El contenido generado por IA puede ser incorrecto.

**Ejercicio 2.47.**   A continuación se muestran dos posibles constructores para marcos:

(define (make-frame origen borde1 borde2)   
  (lista origen borde1 borde2))   
  
(define (make-frame origen borde1 borde2)   
  (cons origen (cons borde1 borde2)))

Para cada constructor, proporcione los selectores apropiados para producir una implementación para los marcos.

**[Pintores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_214)**

Un pintor se representa como un procedimiento que, dado un marco como argumento, dibuja una imagen particular desplazada y escalada para ajustarse al marco. Es decir, si p es un pintor y f es un marco, entonces producimos la imagen de p en f llamando a p con f como argumento.

Los detalles de cómo se implementan los pintores primitivos dependen de las características particulares del sistema gráfico y del tipo de imagen a dibujar. Por ejemplo, supongamos que tenemos un procedimientodraw-line que dibuja una línea en la pantalla entre dos puntos especificados. Luego podemos crear pintores para dibujos de líneas, como el pintor de ondas en la figura  [2.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.10) , a partir de listas de segmentos de línea de la siguiente manera: [27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_215)

(define (segmentos->painter lista-segmentos)   
  (lambda (marco)   
    (for-each   
     (lambda (segmento)   
       (draw-line   
        ((marco-coord-mapa marco) (segmento-inicio segmento))   
        ((marco-coord-mapa marco) (segmento-final segmento))))   
     lista-segmentos)))

Los segmentos se dan mediante coordenadas con respecto al cuadrado unitario. Para cada segmento de la lista, el pintor transforma los puntos finales del segmento con el mapa de coordenadas del marco y dibuja una línea entre los puntos transformados.

La representación de los pintores como procedimientos erige una poderosa barrera de abstracción en el lenguaje pictórico. Podemos crear y mezclar todo tipo de pintores primitivos, basándonos en una variedad de capacidades gráficas. Los detalles de su implementación no importan. Cualquier procedimiento puede servir como pintor, siempre que tome un marco como argumento y dibuje algo a escala para que encaje en el marco. [28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_216)

**Ejercicio 2.48.**  Un segmento de línea dirigido en el plano se puede representar como un par de vectores: el vector que va desde el origen hasta el punto inicial del segmento y el vector que va desde el origen hasta el punto final del segmento. Utilice la representación vectorial del ejercicio  [2.46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.46) para definir una representación de segmentos con un constructor.Segmento de creación y selectoressegmento de inicio ysegmento final .

**Ejercicio 2.49.**   Utilice segmentos->pintor para definir los siguientes pintores primitivos:

a. El pintor que dibuja el contorno del marco designado.

b. El pintor que dibuja una «X» uniendo las esquinas opuestas del marco.

c. El pintor que dibuja una forma de diamante conectando los puntos medios de los lados del marco.

d. El pintor de olas .

**[Transformando y combinando pintores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_219)**

Una operación sobre pintores (como flip-vert o side ) funciona creando un pintor que invoca a los pintores originales con respecto a los marcos derivados del argumento marco. Así, por ejemplo, flip-vert no tiene que saber cómo funciona un pintor para darle la vuelta, solo tiene que saber cómo dar vuelta un marco: el pintor volteado simplemente usa el pintor original, pero en el marco invertido.

Las operaciones de pintor se basan en el procedimiento transform-painter , que toma como argumentos un pintor e información sobre cómo transformar un marco y produce un nuevo pintor. El pintor transformado, cuando se llama en un marco, transforma el marco y llama al pintor original en el marco transformado. Los argumentos de transform-painter son puntos (representados como vectores) que especifican las esquinas del nuevo marco: cuando se asignan al marco, el primer punto especifica el origen del nuevo marco y los otros dos especifican los extremos de sus vectores de borde. Por lo tanto, los argumentos dentro del cuadrado unitario especifican un marco contenido dentro del marco original.

(define (transform-painter pintor origen esquina1 esquina2)   
  (lambda (marco)   
    (let ((m (marco-coord-mapa marco)))   
      (let ((nuevo-origen (m origen)))   
        (painter   
         (make-marco nuevo-origen   
                     (sub-vect (m esquina1) nuevo-origen)   
                     (sub-vect (m esquina2) nuevo-origen)))))))

A continuación se explica cómo voltear las imágenes del pintor verticalmente:

(define (flip-vert pintor)   
  (transform-painter pintor   
                     (make-vect 0.0 1.0)    *; nuevo  origen*  
                     (make-vect 1.0 1.0)    *; nuevo final de  edge1*  
                     (make-vect 0.0 0.0)))  *; nuevo final de  edge2*

Con transform-painter podemos definir fácilmente nuevas transformaciones. Por ejemplo, podemos definir un pintor que reduzca su imagen al cuarto superior derecho del cuadro que se le proporciona:

(define (reduce a pintor superior derecho)   
  (transforma-pintor pintor   
                     (make-vect 0.5 0.5)   
                     (make-vect 1.0 0.5)   
                     (make-vect 0.5 1.0)))

Otras transformaciones rotan las imágenes en sentido antihorario 90 grados [29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_220)

(define (rotate90 pintor)   
  (transform-painter pintor   
                     (make-vect 1.0 0.0)   
                     (make-vect 1.0 1.0)   
                     (make-vect 0.0 0.0)))

o aplastar imágenes hacia el centro del marco: [30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_221)

(define (pintor de aplastamiento hacia adentro)   
  (pintor de transformación pintor   
                     (make-vect 0.0 0.0)   
                     (make-vect 0.65 0.35)   
                     (make-vect 0.35 0.65)))

La transformación de cuadros también es la clave para definir los medios de combinar dos o más pintores. El procedimiento side , por ejemplo, toma dos pintores, los transforma para que pinten en las mitades izquierda y derecha de un cuadro de argumento respectivamente, y produce un nuevo pintor compuesto. Cuando se le da un cuadro al pintor compuesto, llama al primer pintor transformado para que pinte en la mitad izquierda del cuadro y llama al segundo pintor transformado para que pinte en la mitad derecha del cuadro:

(define (al lado de pintor1 pintor2)   
  (let ((punto de división (make-vect 0.5 0.0)))   
    (let ((pintar-izquierda   
           (transformar-pintor pintor1   
                              (make-vect 0.0 0.0)   
                              punto de división   
                              (make-vect 0.0 1.0)))   
          (pintar-derecha   
           (transformar-pintor pintor2   
                              punto de división   
                              (make-vect 1.0 0.0)   
                              (make-vect 0.5 1.0))))   
      (lambda (marco)   
        (pintar-izquierda marco)   
        (pintar-derecha marco)))))

Observe cómo la abstracción de datos de los pintores, y en particular la representación de los pintores como procedimientos, hace que sea fácil implementar el procedimiento side . El procedimiento side no necesita saber nada sobre los detalles de los pintores componentes, excepto que cada pintor dibujará algo en su marco designado.

**Ejercicio 2.50.**   Definir la transformaciónflip-horiz , que voltea a los pintores horizontalmente, y transformaciones que rotan a los pintores en sentido antihorario 180 grados y 270 grados.

**Ejercicio 2.51.**   Definir laoperación below para pintores. Below toma dos pintores como argumentos. El pintor resultante, dado un marco, dibuja con el primer pintor en la parte inferior del marco y con el segundo pintor en la parte superior. Defina below de dos maneras diferentes: primero escribiendo un procedimiento que sea análogo al procedimiento side dado anteriormente, y nuevamente en términos de side y operaciones de rotación adecuadas (del ejercicio  [2.50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.50) ).

**[Niveles de lenguaje para un diseño robusto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_224)**

El lenguaje de imágenes ejercita algunas de las ideas críticas que hemos introducido sobre la abstracción con procedimientos y datos. Las abstracciones de datos fundamentales, los pintores, se implementan utilizando representaciones procedimentales, lo que permite que el lenguaje maneje diferentes capacidades básicas de dibujo de una manera uniforme. Los medios de combinación satisfacen la propiedad de clausura, lo que nos permite construir fácilmente diseños complejos. Finalmente, todas las herramientas para abstraer procedimientos están disponibles para abstraer medios de combinación para pintores.

También hemos vislumbrado otra idea crucial sobre los lenguajes y el diseño de programas: el enfoque de*diseño estratificado* , la noción de que un sistema complejo debe estructurarse como una secuencia de niveles que se describen utilizando una secuencia de lenguajes. Cada nivel se construye combinando partes que se consideran primitivas en ese nivel, y las partes construidas en cada nivel se utilizan como primitivas en el nivel siguiente. El lenguaje utilizado en cada nivel de un diseño estratificado tiene primitivas, medios de combinación y medios de abstracción apropiados para ese nivel de detalle.

El diseño estratificado impregna la ingeniería de sistemas complejos. Por ejemplo, en ingeniería informática, se combinan resistencias y transistores (y se describen utilizando un lenguaje de circuitos analógicos) para producir piezas como puertas and y puertas or, que forman los primitivos de un lenguaje para el diseño de circuitos digitales. [31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_225) Estas piezas se combinan para construir procesadores, estructuras de bus y sistemas de memoria, que a su vez se combinan para formar computadoras, utilizando lenguajes apropiados para la arquitectura informática. Las computadoras se combinan para formar sistemas distribuidos, utilizando lenguajes apropiados para describir interconexiones de red, etc.

Como un pequeño ejemplo de estratificación, nuestro lenguaje de imágenes utiliza elementos primitivos (pintores primitivos) que se crean utilizando un lenguaje que especifica puntos y líneas para proporcionar las listas de segmentos de línea para segments->painter , o los detalles de sombreado para un pintor como rogers . La mayor parte de nuestra descripción del lenguaje de imágenes se centró en la combinación de estos primitivos, utilizando combinadores geométricos como side y below . También trabajamos en un nivel superior, considerando side y below como primitivos para ser manipulados en un lenguaje cuyas operaciones, como square-of-four , capturan patrones comunes de combinación de combinadores geométricos.

El diseño estratificado ayuda a que los programas sean *robustos* , es decir, hace que sea probable que pequeños cambios en una especificación requieran cambios correspondientemente pequeños en el programa. Por ejemplo, supongamos que quisiéramos cambiar la imagen basada en la onda que se muestra en la figura  [2.9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.9) . Podríamos trabajar en el nivel más bajo para cambiar la apariencia detallada del elemento de onda ; podríamos trabajar en el nivel medio para cambiar la forma en que corner-split replica la onda ; podríamos trabajar en el nivel más alto para cambiar la forma en que square-limit organiza las cuatro copias de la esquina. En general, cada nivel de un diseño estratificado proporciona un vocabulario diferente para expresar las características del sistema y un tipo diferente de capacidad para cambiarlo.

**Ejercicio 2.52.**   Realice cambios en el límite cuadrado de onda que se muestra en la figura  [2.9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.9) trabajando en cada uno de los niveles descritos anteriormente. En particular:

a. Añade algunos segmentos al pintor de ondas primitivo del ejercicio   [2.49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.49) (para agregar una sonrisa, por ejemplo).

b. Cambie el patrón construido mediante la división de esquinas (por ejemplo, utilizando solo una copia de las imágenes divididas hacia arriba y hacia la derecha en lugar de dos).

c. Modifique la versión de límite de cuadrados que utiliza el cuadrado de cuatro para ensamblar las esquinas en un patrón diferente. (Por ejemplo, puede hacer que el gran Sr. Rogers mire hacia afuera desde cada esquina del cuadrado).

[6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_154) El uso de la palabraEl término "cierre" proviene del álgebra abstracta, donde se dice que un conjunto de elementos está cerrado bajo una operación si al aplicar la operación a los elementos del conjunto se produce un elemento que es a su vez un elemento del conjunto. La comunidad Lisp también utiliza (desafortunadamente) la palabra "cierre" para describir un concepto totalmente ajeno: un cierre es una técnica de implementación para representar procedimientos con variables libres. En este libro no utilizamos la palabra "cierre" en este segundo sentido.

[7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_155) La noción de que un medio deLa idea de que la combinación debe satisfacer el cierre es sencilla. Desafortunadamente, los combinadores de datos que se ofrecen en muchos lenguajes de programación populares no satisfacen el cierre o hacen que sea complicado explotarlo.Fortran oBásica, uno normalmente combina elementos de datos ensamblándolos en matrices, pero no es posible formar matrices cuyos elementos sean en sí mismas matrices.Pascal yC admite estructuras cuyos elementos son estructuras. Sin embargo, esto requiere que el programador manipule punteros explícitamente y se adhiera a la restricción de que cada campo de una estructura puede contener solo elementos de una forma preestablecida. A diferencia de Lisp con sus pares, estos lenguajes no tienen un pegamento de propósito general incorporado que facilite la manipulación de datos compuestos de manera uniforme. Esta limitación se encuentra detrás de AlanEl comentario de Perlis en su prólogo a este libro: ``En Pascal, la plétora de estructuras de datos declarables induce una especialización dentro de las funciones que inhibe y penaliza la cooperación casual. Es mejor tener 100 funciones operando en una estructura de datos que tener 10 funciones operando en 10 estructuras de datos.''

[8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_156) En este libro, utilizamos*el término lista*para referirnos a una cadena de pares que terminan en el marcador de fin de lista. En cambio, el término*La estructura de lista* se refiere a cualquier estructura de datos formada por pares, no solo a listas.

[9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_157) Dado que las aplicaciones anidadas decarycdrson complicadas de escribir, los dialectos Lisp proporcionan abreviaturas para ellas, por ejemplo,



Los nombres de todos estos procedimientos comienzan con c y terminan con r . Cada a entre ellos representa unaoperación de coche y cada d para una operación de cdr , que se aplicarán en el mismo orden en que aparecen en el nombre. Los nombres de coche y cdr persisten porque combinaciones simples como cadr son pronunciables.

[10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_158) Es notable la cantidad de energía que se ha desperdiciado en la estandarización de los dialectos de Lisp en discusiones que literalmente no tienen nada que ver con nada:¿nildebería ser un nombre común? ¿El valor denildebería ser un símbolo? ¿Debería ser una lista? ¿Debería ser un par?En Scheme, nil es un nombre común, que usamos en esta sección como una variable cuyo valor es el marcador de fin de lista (así como true es una variable común que tiene un valor verdadero). Otros dialectos de Lisp, incluido Common Lisp, tratan a nil como un símbolo especial.Los autores de este libro, que han soportado demasiadas peleas por la estandarización del lenguaje, desearían evitar todo el problema. Una vez que hayamos introducido las citas en la sección  [2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3) , denotaremos la lista vacía como '() y prescindiremos por completo de la variable nil .

[11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_164) Para definirfygutilizandolambda escribiríamos

(define f (lambda (x y . z) < *cuerpo* >))   
(define g (lambda w < *cuerpo* >))

[12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_166) El esquema proporciona de manera estándar unaprocedimiento de mapa que es más general que el que se describe aquí. Este mapa más general toma un procedimiento de *n* argumentos, junto con *n* listas, y aplica el procedimiento a todos los primeros elementos de las listas, a todos los segundos elementos de las listas, y así sucesivamente, devolviendo una lista de los resultados. Por ejemplo:

(mapa + (lista 1 2 3) (lista 40 50 60) (lista 700 800 900))   
*(741 852 963)*  
  
(mapa (lambda (x y) (+ x (\* 2 y)))   
     (lista 1 2 3)   
     (lista 4 5 6))   
*(9 12 15)*

El orden de las dos primeras cláusulas de lasegunda [cláusula es](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_170) importante, ya que la lista vacía satisfacenull?y además no es un par.

[14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_182) Éste es, de hecho, precisamente elprocedimiento  de franja del ejercicio [2.28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.28) . Aquí le hemos cambiado el nombre para enfatizar que es parte de una familia de procedimientos generales de manipulación de secuencias.

[15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_183) Richard Waters (1979) desarrolló un programa que analiza automáticamente los datos tradicionales.Programas Fortran, viéndolos en términos de mapas, filtros y acumulaciones. Descubrió que el 90 por ciento del código del Paquete de subrutinas científicas Fortran encaja perfectamente en este paradigma. Una de las razones del éxito de Lisp como lenguaje de programación es que las listas proporcionan un medio estándar para expresar colecciones ordenadas de modo que puedan manipularse mediante operaciones de orden superior. El lenguaje de programaciónAPL debe gran parte de su poder y atractivo a una opción similar. En APL, todos los datos se representan como matrices y existe un conjunto universal y conveniente de operadores genéricos para todo tipo de operaciones con matrices.

[16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_186) SegúnKnuth (1981), esta regla fue formulada porWG Horner a principios del siglo XIX, pero el método fue utilizado por Newton más de cien años antes. La regla de Horner evalúa el polinomio utilizando menos sumas y multiplicaciones que el método sencillo de calcular primero *a n* *x n* , luego sumar *a n* -1 *x n* -1 , y así sucesivamente. De hecho, es posible demostrar que cualquier algoritmo para evaluar polinomios arbitrarios debe utilizar al menos tantas sumas y multiplicaciones como la regla de Horner y, por lo tanto, la regla de Horner es unaAlgoritmo óptimo para la evaluación de polinomios. Esto fue demostrado (para el número de adiciones) porAM Ostrowski en un artículo de 1954 que esencialmente fundó el estudio moderno de algoritmos óptimos. La afirmación análoga para las multiplicaciones fue demostrada porVY Pan en 1966. El libro deBorodin yMunro (1975) ofrece una descripción general de estos y otros resultados sobre algoritmos óptimos.

[17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_190) Esta definición utiliza la versión extendida delmapadescrita en la nota [12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#footnote_Temp_166).

[18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_194) Este enfoque de las asignaciones anidadas nos lo mostróDavid Turner, cuyos idiomasKRC yMiranda ofrece formalismos elegantes para tratar con estos constructos. Los ejemplos de esta sección (ver también el ejercicio  [2.42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.42) ) están adaptados de Turner 1981. En la sección  [3.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.3) , veremos cómo este enfoque se generaliza a secuencias infinitas.

Aquí representamos un par como una lista de dos elementos en lugar de como un par Lisp. Por lo tanto, el «par» ( [i](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_195) *,*j*)*se representa como(lista ij), no como(cons ij).

[20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_196) El conjunto*S*-*x*es el conjunto de todos los elementos de*S*, excluyendo*x*.

[21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_197) Los puntos y comas en el código Scheme se utilizan para introducir *comentarios* . El intérprete ignora todo lo que va desde el punto y coma hasta el final de la línea. En este libro no utilizamos muchos comentarios; intentamos que nuestros programas se autodocumenten mediante el uso de nombres descriptivos.

[22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_202) El lenguaje de la imagen se basa en el lenguajePeter Henderson creó para construir imágenes comoXilografía «Límite cuadrado» de MC Escher (véase Henderson 1982). La xilografía incorpora un patrón repetido a escala, similar a las disposiciones dibujadas con el procedimiento de límite cuadrado en esta sección.

[23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_204) William Barton Rogers (1804-1882) fue el fundador y primer presidente del MIT. Geólogo y profesor de gran talento, impartió clases en el William and Mary College y en la Universidad de Virginia. En 1859 se trasladó a Boston, donde disponía de más tiempo para la investigación, trabajó en un plan para establecer un "instituto politécnico" y fue el primer inspector estatal de medidores de gas de Massachusetts.

Cuando se fundó el MIT en 1861, Rogers fue elegido su primer presidente. Rogers defendía un ideal de «aprendizaje útil» que era diferente de la educación universitaria de la época, con su énfasis excesivo en los clásicos, que, como escribió, «obstaculizaban la instrucción y la disciplina más amplias, superiores y prácticas de las ciencias naturales y sociales». Esta educación también debía ser diferente de la educación limitada de las escuelas profesionales. En palabras de Rogers:

La distinción impuesta por el mundo entre el trabajador práctico y el científico es absolutamente inútil, y toda la experiencia de los tiempos modernos ha demostrado su total inutilidad.

Rogers fue presidente del MIT hasta 1870, cuando renunció debido a problemas de salud. En 1878, el segundo presidente del MIT,John Runkle dimitió bajo la presión de una crisis financiera provocada por el pánico de 1873 y la tensión de luchar contra los intentos de Harvard de apoderarse del MIT. Rogers volvió a ocupar el cargo de presidente hasta 1881.

Rogers se desplomó y murió mientras se dirigía a la clase de graduados del MIT en la ceremonia de graduación de 1882. Runkle citó las últimas palabras de Rogers en un discurso conmemorativo pronunciado ese mismo año:

``Mientras estoy aquí hoy y veo lo que es el Instituto ... recuerdo los comienzos de la ciencia. Recuerdo que hace ciento cincuenta años Stephen Hales publicó un folleto sobre el tema del gas de iluminación, en el que afirmaba que sus investigaciones habían demostrado que 128 granos de carbón bituminoso...''

«Carbón bituminoso», fueron sus últimas palabras en la tierra. Se inclinó hacia delante, como si estuviera consultando algunas notas que había en la mesa que tenía delante, y luego, lentamente, recuperó la posición erguida, levantó las manos y fue trasladado desde el escenario de sus trabajos y triunfos terrenales al «mañana de la muerte», donde se resuelven los misterios de la vida y el espíritu incorpóreo encuentra satisfacción eterna al contemplar los nuevos y aún insondables misterios del futuro infinito.

En palabras de Francis A. Walker(El tercer presidente del MIT):

Durante toda su vida se había comportado con la mayor fidelidad y heroísmo, y murió como seguramente lo hubiera deseado cualquier buen caballero, en el arnés, en su puesto y en el mismo papel y acto del deber público.

[24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_207) De manera equivalente, podríamos escribir

(definir pares invertidos   
  (identidad del cuadrado de cuatro identidad flip-vert))

[25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_208) Rotate180rota un objeto pintor 180 grados (ver ejercicio [2.50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.50)). En lugar derotate180podríamos decir(compose flip-vert flip-horiz), utilizando elprocedimiento compose[1.42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.42).

[Frame](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_211) -coord-maputiliza las operaciones vectoriales descritas en el ejercicio [2.46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.46)a continuación, que suponemos que se han implementado utilizando alguna representación para vectores. Debido a la abstracción de datos, no importa cuál sea esta representación vectorial, siempre que las operaciones vectoriales se comporten correctamente.

[27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_215) Segments->painterutiliza la representación de segmentos de línea descrita en el ejercicio [2.48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.48)a continuación. También utiliza elprocedimiento for-each[2.23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.23).

[28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_216) Por ejemplo, elpintor de Rogers[2.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.11)se construyó a partir de una imagen en niveles de gris. Para cada punto de un marco dado, elde Rogersdetermina el punto de la imagen que se le asigna bajo el mapa de coordenadas del marco y lo sombrea en consecuencia. Al permitir diferentes tipos de pintores, estamos capitalizando la idea de datos abstractos discutida en la sección [2.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.3), donde argumentamos que una representación de número racional podría ser cualquier cosa que satisfaga una condición apropiada. Aquí estamos usando el hecho de que un pintor puede implementarse de cualquier manera, siempre que dibuje algo en el marco designado. La sección [2.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.3)también mostró cómo los pares podrían implementarse como procedimientos. Los pintores son nuestro segundo ejemplo de una representación procedimental para datos.

[29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_220) Rotate90es una rotación pura solo para marcos cuadrados, porque también estira y encoge la imagen para que encaje en el marco rotado.

[30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_221) Las imágenes en forma de diamante de las figuras [2.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.10)y [2.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.11)se crearonsquash-inwardsawaveyrogers.

[31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_225) La sección [3.3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.4)describe uno de esos lenguajes.

[**2.3 Datos simbólicos**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_2.3)

Todos los objetos de datos compuestos que hemos utilizado hasta ahora se construyeron básicamente a partir de números. En esta sección ampliamos la capacidad de representación de nuestro lenguaje introduciendo la posibilidad de trabajar con símbolos arbitrarios como datos.

**[2.3.1 Cotización](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.3.1)**

Si podemos formar datos compuestos usando símbolos, podemos tener listas como

(a b c d)   
(23 45 17)   
((Norah 12) (Molly 9) (Anna 7) (Lauren 6) (Charlotte 4))

Las listas que contienen símbolos pueden parecerse a las expresiones de nuestro idioma:

(\* (+ 23 45) (+ x 9))   
  
(define (hecho n) (si (= n 1) 1 (\* n (hecho (- n 1)))))

Para manipular símbolos necesitamos un nuevo elemento en nuestro lenguaje: la capacidad de *citar* un objeto de datos. Supongamos que queremos construir la lista (ab) . No podemos lograr esto con (list ab) , porque esta expresión construye una lista de los *valores* de a y b en lugar de los símbolos mismos. Este problema es bien conocido en el contexto deLenguajes naturales, donde las palabras y oraciones pueden ser consideradas como entidades semánticas o como cadenas de caracteres (entidades sintácticas). La práctica común en los lenguajes naturales es usar comillas para indicar que una palabra o una oración debe ser tratada literalmente como una cadena de caracteres. Por ejemplo, la primera letra de "Juan" es claramente "J". Si le decimos a alguien "di tu nombre en voz alta", esperamos escuchar el nombre de esa persona. Sin embargo, si le decimos a alguien "di 'tu nombre' en voz alta", esperamos escuchar las palabras "tu nombre". Nótese que estamos obligados a anidar comillas para describir lo que otra persona podría decir. [32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_227)

Podemos seguir esta misma práctica para identificar listas y símbolos que deben tratarse como objetos de datos en lugar de expresiones que deben evaluarse. Sin embargo, nuestro formato para citar difiere del de los lenguajes naturales en que colocamos una comilla (tradicionalmente, la únicaEl símbolo de comillas simples  ' ) solo se utiliza al principio del objeto que se va a citar. Esto se puede hacer sin problemas en la sintaxis de Scheme porque dependemos de espacios en blanco y paréntesis para delimitar los objetos. Por lo tanto, el significado del carácter de comillas simples es citar el siguiente objeto. [33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_228)

Ahora podemos distinguir entre símbolos y sus valores:

(define a 1)   
  
(define b 2)   
  
(enumera a b)   
*(1 2)*  
  
(enumera 'a 'b)   
*(a b)*  
  
(enumera 'a b)   
*(a 2)*

Las citas también nos permiten escribir objetos compuestos, utilizando la representación impresa convencional para listas: [34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_229)

(coche '(a b c))   
*a*  
  
(cdr '(a b c))   
*(b c)*

De acuerdo con esto, podemos obtener la lista vacía evaluando '() , y así prescindir de la variable nil .

Un primitivo adicional utilizado para manipular símbolos eseq? , que toma dos símbolos como argumentos y prueba si son iguales. [35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_230) Usando eq? , podemos implementar un procedimiento útil llamado memq . Este toma dos argumentos, un símbolo y una lista. Si el símbolo no está contenido en la lista (es decir, no es eq? para ningún elemento de la lista), entonces memq devuelve falso. De lo contrario, devuelve la sublista de la lista comenzando con la primera ocurrencia del símbolo:

(define (memq elemento x)   
  (cond ((null? x) falso)   
        ((eq? elemento (car x)) x)   
        (else (memq elemento (cdr x)))))

Por ejemplo, el valor de

(memq 'manzana' (pera, plátano, ciruela))

es falso, mientras que el valor de

(memq 'manzana' (x (puré de manzana) y manzana pera))

es (manzana pera) .

**Ejercicio 2.53.**   ¿Qué imprimiría el intérprete en respuesta a evaluar cada una de las siguientes expresiones?

(lista 'a' b' c)   
  
(lista (lista 'george))   
(cdr '((x1 x2) (y1 y2)))   
  
(cadr '((x1 x2) (y1 y2)))   
(par? (auto '(una lista corta)))   
(memq 'rojo '((zapatos rojos) (calcetines azules)))   
  
(memq 'rojo '(zapatos rojos calcetines azules))

**Ejercicio 2.54.**   Se dice que dos listas son¿Iguales? Si contienen elementos iguales dispuestos en el mismo orden. Por ejemplo,

(¿igual? '(esta es una lista) '(esta es una lista))

es cierto, pero

(¿igual? '(esta es una lista) '(esta (es una) lista))

es falso. Para ser más precisos, podemos definir equal? ​​recursivamente en términos de la igualdad básica eq? de símbolos diciendo que a y b son equal? ​​si ambos son símbolos y los símbolos son eq?, o si ambos son listas tales que (car a) es igual? a (car b) y (cdr a) es igual? a (cdr b) . Usando esta idea, implemente equal? ​​como un procedimiento. [36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_233)

**Ejercicio 2.55.**   Eva Lu Ator le escribe al intérprete la expresión

(coche ''abracadabra)

Para su sorpresa, el intérprete imprime la cita de nuevo . Explíquese.

**[2.3.2 Ejemplo: Diferenciación simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.3.2)**

Como ilustración de la manipulación de símbolos y una ilustración adicional de la abstracción de datos, considere el diseño de un procedimiento que realiza la diferenciación simbólica de expresiones algebraicas. Nos gustaría que el procedimiento tomara como argumentos una expresión algebraica y una variable y devolviera la derivada de la expresión con respecto a la variable. Por ejemplo, si los argumentos del procedimiento son *a x* 2 + *b x* + *c* y *x* , el procedimiento debería devolver 2 *a x* + *b* . La diferenciación simbólica tiene una importancia histórica especial en Lisp. Fue uno de los ejemplos motivadores detrás del desarrollo de un lenguaje de computadora para la manipulación de símbolos. Además, marcó el comienzo de la línea de investigación que condujo al desarrollo de poderosos sistemas para el trabajo matemático simbólico, que actualmente están siendo utilizados por un número creciente de matemáticos y físicos aplicados.

Al desarrollar el programa de diferenciación simbólica, seguiremos la misma estrategia de abstracción de datos que seguimos al desarrollar el sistema de números racionales de la sección  [2.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.1) . Es decir, primero definiremos un algoritmo de diferenciación que opere sobre objetos abstractos como “sumas”, “productos” y “variables” sin preocuparnos por cómo se deben representar. Solo después abordaremos el problema de representación.

**[El programa de diferenciación con datos abstractos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_235)**

Para simplificar las cosas, consideraremos un programa de diferenciación simbólica muy simple que maneja expresiones que se construyen utilizando únicamente las operaciones de suma y multiplicación con dos argumentos. La diferenciación de cualquier expresión de este tipo se puede realizar aplicando las siguientes reglas de reducción:









Observe que las dos últimas reglas son de naturaleza recursiva. Es decir, para obtener la derivada de una suma primero encontramos las derivadas de los términos y las sumamos. Cada uno de los términos puede ser a su vez una expresión que necesita ser descompuesta. Al descomponer en partes cada vez más pequeñas, eventualmente se obtendrán partes que son constantes o variables, cuyas derivadas serán 0 o 1.

Para plasmar estas reglas en un procedimiento nos permitimos un poco...Pensamiento ilusorio, como hicimos al diseñar la implementación de números racionales. Si tuviéramos un medio para representar expresiones algebraicas, deberíamos poder decir si una expresión es una suma, un producto, una constante o una variable. Deberíamos poder extraer las partes de una expresión. Para una suma, por ejemplo, queremos poder extraer el sumando (primer término) y el sugando (segundo término). También deberíamos poder construir expresiones a partir de partes. Supongamos que ya tenemos procedimientos para implementar los siguientes selectores, constructores y predicados:

|  |  |
| --- | --- |
| (variable? e) | ¿Es e una variable? |
| (misma variable? v1 v2) | ¿ V1 y v2 son la misma variable? |
| (suma? e) | ¿Es e una suma? |
| (añadir e) | Sumando de la suma e . |
| (aumenta e) | Aumente de la suma e . |
| (hacer suma a1 a2) | Construye la suma de a1 y a2 . |
| (producto?e) | ¿ Es un producto? |
| (multiplicador e) | Multiplicador del producto e . |
| (multiplicando e) | Multiplicando del producto e . |
| (fabricar-producto m1 m2) | Construya el producto de m1 y m2 . |

Usando estos, y el predicado primitivo número ?,que identifica números, podemos expresar las reglas de diferenciación como el siguiente procedimiento:

(define (deriva exp var)   
  (cond ((número? exp) 0)   
        ((variable? exp)   
         (if (misma-variable? exp var) 1 0))   
        ((suma? exp)   
         (hacer-suma (deriva (addend exp) var)   
                   (deriva (augend exp) var)))   
        ((producto? exp)   
         (hacer-suma   
           (hacer-producto (multiplicador exp)   
                         (deriva (multiplicando exp) var))   
           (hacer-producto (deriva (multiplicador exp) var)   
                         (multiplicando exp))))   
        (else   
         (error "tipo de expresión desconocido -- DERIV" exp))))

Este procedimiento de derivación incorpora el algoritmo de diferenciación completo. Dado que se expresa en términos de datos abstractos, funcionará sin importar cómo elijamos representar las expresiones algebraicas, siempre que diseñemos un conjunto adecuado de selectores y constructores. Esta es la cuestión que debemos abordar a continuación.

**[Representación de expresiones algebraicas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_236)**

Podemos imaginar muchas formas de utilizar la estructura de listas para representar expresiones algebraicas. Por ejemplo, podríamos utilizar listas de símbolos que reflejen la notación algebraica habitual, representando *a x* + *b* como la lista (a \* x + b) . Sin embargo, una opción especialmente sencilla es utilizar la misma notación de prefijo entre paréntesis que utiliza Lisp para las combinaciones; es decir, representar *a x* + *b* como (+ (\* ax) b) . Entonces nuestra representación de datos para el problema de diferenciación es la siguiente:

* Las variables son símbolos. Se identifican mediante el predicado primitivo.¿símbolo? :

(definir (variable? x) (símbolo? x))

* Dos variables son iguales si los símbolos que las representan son eq? :

(definir (misma variable? v1 v2)   
  (y (variable? v1) (variable? v2) (eq? v1 v2)))

* Las sumas y los productos se construyen como listas:

(define (hace-suma a1 a2) (lista '+ a1 a2))  
  
(define (make-product m1 m2) (lista '\* m1 m2))

* Una suma es una lista cuyo primer elemento es el símbolo + :

(definir (suma? x)   
  (y (par? x) (ecuación? (auto x) '+)))

* El sumando es el segundo elemento de la lista de suma:

(define (añade s) (cadr s))

* La sugendo es el tercer elemento de la lista suma:

(definir (aumentar s) (caddr s))

* Un producto es una lista cuyo primer elemento es el símbolo \* :

(define (producto? x)   
  (y (par? x) (ecuación? (auto x) '\*)))

* El multiplicador es el segundo elemento de la lista de productos:

(definir (multiplicador p) (cadr p))

* El multiplicando es el tercer elemento de la lista de productos:

(define (multiplicando p) (caddr p))

Por lo tanto, sólo necesitamos combinarlos con el algoritmo que incorpora deriv para tener un programa de diferenciación simbólica que funcione. Veamos algunos ejemplos de su comportamiento:

(deriva '(+ x 3) 'x)   
*(+ 1 0)*  
(deriva '(\* x y) 'x)   
*(+ (\* x 0) (\* 1 y))*  
(deriva '(\* (\* x y) (+ x 3)) 'x)   
*(+ (\* (\* x y) (+ 1 0))   
   (\* (+ (\* x 0) (\* 1 y))   
      (+ x 3)))*

El programa produce respuestas que son correctas, pero no están simplificadas. Es cierto que



pero nos gustaría que el programa supiera que *x* · 0 = 0, 1 · *y* = *y* , y 0 + *y* = *y* . La respuesta para el segundo ejemplo debería haber sido simplemente  y . Como muestra el tercer ejemplo, esto se convierte en un problema serio cuando las expresiones son complejas.

Nuestra dificultad es muy parecida a la que encontramos con la implementación de números racionales: no hemos reducido las respuestas a la forma más simple. Para lograr la reducción de números racionales, necesitábamos cambiar solo los constructores y los selectores de la implementación. Podemos adoptar una estrategia similar aquí. No cambiaremos deriv en absoluto. En cambio, cambiaremos make-sum de modo que si ambos sumandos son números, make-sum los sumará y devolverá su suma. Además, si uno de los sumandos es 0, entonces make-sum devolverá el otro sumando.

(define (make-sum a1 a2)   
  (cond ((=número? a1 0) a2)   
        ((=número? a2 0) a1)   
        ((y (número? a1) (número? a2)) (+ a1 a2))   
        (sino (lista '+ a1 a2))))

Esto utiliza el procedimiento =número?, que verifica si una expresión es igual a un número dado:

(define (=número? exp num)   
  (y (número? exp) (= exp num)))

De manera similar, cambiaremos make-product para incorporar las reglas de que 0 veces cualquier cosa es 0 y 1 veces cualquier cosa es la cosa misma:

(define (make-product m1 m2)   
  (cond ((o (=número? m1 0) (=número? m2 0)) 0)   
        ((=número? m1 1) m2)   
        ((=número? m2 1) m1)   
        ((y (número? m1) (número? m2)) (\* m1 m2))   
        (sino (lista '\* m1 m2))))

Así es como funciona esta versión en nuestros tres ejemplos:

(deriva '(+ x 3) 'x)   
*1*  
(deriva '(\* x y) 'x)   
*y*  
(deriva '(\* (\* x y) (+ x 3)) 'x)   
*(+ (\* x y) (\* y (+ x 3)))*

Aunque esto es una mejora considerable, el tercer ejemplo muestra que todavía hay un largo camino por recorrer antes de que obtengamos un programa que ponga expresiones en una forma que podamos considerar "más simple". El problema de la simplificación algebraica es complejo porque, entre otras razones, una forma que puede ser más simple para un propósito puede no serlo para otro.

**Ejercicio 2.56.**  Muestra cómo ampliar el diferenciador básico para manejar más tipos de expresiones. Por ejemplo, implementa la regla de diferenciación



agregando una nueva cláusula al programa deriv y definiendo los procedimientos apropiados exponentiation?, base , exponent y make -exponentiation . (Puede usar el símbolo \*\* para denotar exponenciación). Incorpore las reglas de que cualquier cosa elevada a la potencia 0 es 1 y cualquier cosa elevada a la potencia 1 es la cosa en sí.

**Ejercicio 2.57.**   Amplíe el programa de diferenciación para que pueda manejar sumas y productos de cantidades arbitrarias de (dos o más) términos. Entonces, el último ejemplo anterior podría expresarse como

(deriva '(\* x y (+ x 3)) 'x)

Intente hacer esto modificando únicamente la representación de las sumas y los productos, sin cambiar en absoluto el procedimiento de derivación . Por ejemplo, el sumando de una suma sería el primer término y el sugando sería la suma del resto de los términos.

**Ejercicio 2.58.**  Supongamos que queremos modificar el programa de diferenciación para que funcione con notación matemática ordinaria, en la que + y \* son operadores infijos en lugar de prefijos. Dado que el programa de diferenciación se define en términos de datos abstractos, podemos modificarlo para que funcione con diferentes representaciones de expresiones simplemente cambiando los predicados, selectores y constructores que definen la representación de las expresiones algebraicas sobre las que debe operar el diferenciador.

a. Muestre cómo hacer esto para diferenciar expresiones algebraicas presentadas en forma infija, como (x + (3 \* (x + (y + 2)))) . Para simplificar la tarea, suponga que + y \* siempre toman dos argumentos y que las expresiones están completamente entre paréntesis.

b. El problema se vuelve sustancialmente más difícil si permitimos la notación algebraica estándar, como (x + 3 \* (x + y + 2)) , que elimina los paréntesis innecesarios y supone que la multiplicación se realiza antes de la suma. ¿Puede diseñar predicados, selectores y constructores adecuados para esta notación de modo que nuestro programa derivado siga funcionando?

**[2.3.3 Ejemplo: Representación de conjuntos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.3.3)**

En los ejemplos anteriores construimos representaciones para dos tipos de objetos de datos compuestos: números racionales y expresiones algebraicas. En uno de estos ejemplos teníamos la opción de simplificar (reducir) las expresiones en el momento de la construcción o en el momento de la selección, pero, aparte de eso, la elección de una representación para estas estructuras en términos de listas fue sencilla. Cuando pasamos a la representación de conjuntos, la elección de una representación no es tan obvia. De hecho, hay varias representaciones posibles y difieren significativamente entre sí en varios aspectos.

De manera informal, un conjunto es simplemente una colección de objetos distintos. Para dar una definición más precisa, podemos emplear el método de abstracción de datos. Es decir, definimos "conjunto" especificando las operaciones que se van a utilizar en los conjuntos. Estas son union-set , junction-set , element-of-set? y adjoin-set .¿Elemento de un conjunto? es un predicado que determina si un elemento dado es miembro de un conjunto.Adjoin-set toma un objeto y un conjunto como argumentos y devuelve un conjunto que contiene los elementos del conjunto original y también el elemento adjunto.Union-set calcula la unión de dos conjuntos, que es el conjunto que contiene cada elemento que aparece en cualquiera de los argumentos.El conjunto de intersección calcula la intersección de dos conjuntos, que es el conjunto que contiene únicamente los elementos que aparecen en ambos argumentos. Desde el punto de vista de la abstracción de datos, tenemos la libertad de diseñar cualquier representación que implemente estas operaciones de una manera coherente con las interpretaciones dadas anteriormente. [37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_240)

**[Conjuntos como listas desordenadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_241)**

Una forma de representar un conjunto es como una lista de sus elementos en la que ningún elemento aparece más de una vez. El conjunto vacío se representa mediante la lista vacía. En esta representación, element-of-set? es similar al procedimiento memq de la sección  [2.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.1) . Utiliza equal? ​​en lugar de eq? de modo que los elementos del conjunto no necesitan ser símbolos:

(define (elemento-del-conjunto? x conjunto)   
  (cond ((nulo? conjunto) falso)   
        ((igual? x (conjunto coche)) verdadero)   
        (de lo contrario (elemento-del-conjunto? x (conjunto cdr)))))

Con esto, podemos escribir adjoin-set . Si el objeto que se va a adjuntar ya está en el conjunto, simplemente devolvemos el conjunto. De lo contrario, usamos cons para agregar el objeto a la lista que representa el conjunto:

(define (conjunto adjunto x conjunto)   
  (si (elemento del conjunto? x conjunto)   
      conjunto   
      (cons x conjunto)))

Para el conjunto de intersección podemos utilizar una estrategia recursiva. Si sabemos cómo formar la intersección del conjunto2 y el cdr del conjunto1 , solo necesitamos decidir si incluimos el coche del conjunto1 en esto. Pero esto depende de si (coche del conjunto1) también está en el conjunto2 . Aquí está el procedimiento resultante:

(define (conjunto-de-intersección conjunto1 conjunto2)   
  (cond ((o (¿nulo? conjunto1) (¿nulo? conjunto2)) '())   
        ((elemento-del-conjunto? (automóvil conjunto1) conjunto2)           
         (cons (automóvil conjunto1)   
               (conjunto-de-intersección (cdr conjunto1) conjunto2)))   
        (sino (conjunto-de-intersección (cdr conjunto1) conjunto2))))

Al diseñar una representación, una de las cuestiones que nos debe preocupar es la eficiencia. Consideremos el número de pasos requeridos por nuestras operaciones de conjunto. Dado que todas utilizan element-of-set?, la velocidad de esta operación tiene un impacto importante en la eficiencia de la implementación del conjunto en su conjunto. Ahora, para verificar si un objeto es miembro de un conjunto, element-of-set? puede tener que escanear todo el conjunto. (En el peor de los casos, resulta que el objeto no está en el conjunto). Por lo tanto, si el conjunto tiene *n* elementos, element-of-set? puede tomar hasta *n* pasos. Por lo tanto, el número de pasos requeridos crece como ( *n* ). El número de pasos requeridos por adjoin-set , que utiliza esta operación, también crece como ( *n* ). Para junction-set , que realiza una verificación element-of-set? para cada elemento del conjunto1 , el número de pasos requeridos crece como el producto de los tamaños de los conjuntos involucrados, o ( *n* 2 ) para dos conjuntos de tamaño *n* . Lo mismo ocurrirá con el conjunto unión .

**Ejercicio 2.59.**   Implementar laoperación de unión de conjuntos para la representación de conjuntos en forma de lista desordenada.

**Ejercicio 2.60.**   Hemos especificado que un conjunto se representaría como una lista sin duplicados. Ahora supongamos que permitimos duplicados. Por ejemplo, el conjunto {1,2,3} podría representarse como la lista (2 3 2 1 3 2 2) . Diseñe los procedimientos elemento-del-conjunto? , conjunto-adjunto , conjunto-unión y conjunto-intersección que operan sobre esta representación. ¿Cómo se compara la eficiencia de cada uno con el procedimiento correspondiente para la representación sin duplicados? ¿Existen aplicaciones para las que utilizaría esta representación en lugar de la sin duplicados?

**[Conjuntos como listas ordenadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_244)**

Una forma de acelerar nuestras operaciones de conjunto es cambiar la representación de modo que los elementos del conjunto se enumeren en orden creciente. Para ello, necesitamos alguna forma de comparar dos objetos para poder decir cuál es más grande. Por ejemplo, podríamos comparar símbolos lexicográficamente, o podríamos acordar algún método para asignar un número único a un objeto y luego comparar los elementos comparando los números correspondientes. Para mantener nuestra discusión simple, consideraremos solo el caso en el que los elementos del conjunto son números, de modo que podamos comparar elementos usando > y < . Representaremos un conjunto de números enumerando sus elementos en orden creciente. Mientras que nuestra primera representación anterior nos permitía representar el conjunto {1,3,6,10} enumerando los elementos en cualquier orden, nuestra nueva representación solo permite la lista (1 3 6 10) .

Una ventaja de ordenar se muestra en el elemento del conjunto : al verificar la presencia de un elemento, ya no tenemos que escanear todo el conjunto. Si llegamos a un elemento del conjunto que es más grande que el elemento que estamos buscando, entonces sabemos que el elemento no está en el conjunto:

(define (elemento-del-conjunto? x conjunto)   
  (cond ((nulo? conjunto) falso)   
        ((= x (conjunto de carro)) verdadero)   
        ((< x (conjunto de carro)) falso)   
        (else (elemento-del-conjunto? x (conjunto de cdr)))))

¿Cuántos pasos se ahorran con esto? En el peor de los casos, el elemento que buscamos puede ser el más grande del conjunto, por lo que el número de pasos es el mismo que para la representación desordenada. Por otro lado, si buscamos elementos de muchos tamaños diferentes, podemos esperar que a veces podamos detener la búsqueda en un punto cerca del comienzo de la lista y que otras veces aún necesitemos examinar la mayor parte de la lista. En promedio, deberíamos esperar tener que examinar aproximadamente la mitad de los elementos del conjunto. Por lo tanto, el número promedio de pasos necesarios será de aproximadamente *n* /2. Esto sigue siendo un crecimiento de ( *n* ), pero nos ahorra, en promedio, un factor de 2 en el número de pasos con respecto a la implementación anterior.

Obtenemos una aceleración más impresionante con el conjunto de intersección . En la representación desordenada, esta operación requirió ( *n* 2 ) pasos, porque realizamos un escaneo completo del conjunto2 para cada elemento del conjunto1 . Pero con la representación ordenada, podemos usar un método más inteligente. Comience por comparar los elementos iniciales, x1 y x2 , de los dos conjuntos. Si x1 es igual a x2 , entonces eso da un elemento de la intersección, y el resto de la intersección es la intersección de las cdr de los dos conjuntos. Supongamos, sin embargo, que x1 es menor que x2 . Como x2 es el elemento más pequeño del conjunto2 , podemos concluir inmediatamente que x1 no puede aparecer en ningún lugar del conjunto2 y, por lo tanto, no está en la intersección. Por lo tanto, la intersección es igual a la intersección del conjunto2 con la cdr del conjunto1 . De manera similar, si x2 es menor que x1 , entonces la intersección está dada por la intersección del conjunto1 con la cdr del conjunto2 . Aquí está el procedimiento:

(define (conjunto-intersección conjunto1 conjunto2)   
  (si (o (¿nulo? conjunto1) (¿nulo? conjunto2))   
      '()       
      (let ((x1 (coche conjunto1)) (x2 (coche conjunto2)))   
        (cond ((= x1 x2)   
               (cons x1   
                     (conjunto-intersección (cdr conjunto1)   
                                       (cdr conjunto2))))   
              ((< x1 x2)   
               (conjunto-intersección (cdr conjunto1) conjunto2))   
              ((< x2 x1)   
               (conjunto-intersección conjunto1 (cdr conjunto2)))))))

Para estimar el número de pasos requeridos por este proceso, observe que en cada paso reducimos el problema de intersección al cálculo de intersecciones de conjuntos más pequeños, eliminando el primer elemento del conjunto 1 o del conjunto 2 o de ambos. Por lo tanto, el número de pasos requeridos es como máximo la suma de los tamaños del conjunto 1 y del conjunto 2 , en lugar del producto de los tamaños como con la representación no ordenada. Esto es un crecimiento de ( *n* ) en lugar de ( *n* 2 ), una aceleración considerable, incluso para conjuntos de tamaño moderado.

**Ejercicio 2.61.**   Dar una implementación deconjunto adjunto utilizando la representación ordenada. Por analogía con el elemento del conjunto, muestre cómo aprovechar el orden para producir un procedimiento que requiere en promedio aproximadamente la mitad de pasos que con la representación desordenada.

**Ejercicio 2.62.**   Dar una ( *n* ) implementación deconjunto de unión para conjuntos representados como listas ordenadas.

**[Conjuntos como árboles binarios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_247)**

Podemos hacerlo mejor que la representación de lista ordenada organizando los elementos del conjunto en forma de árbol. Cada nodo del árbol contiene un elemento del conjunto, llamado la "entrada" en ese nodo, y un enlace a cada uno de los otros dos nodos (posiblemente vacíos). El enlace "izquierdo" apunta a elementos más pequeños que el del nodo, y el enlace "derecho" a elementos más grandes que el del nodo. La Figura  [2.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.16) muestra algunos árboles que representan el conjunto {1,3,5,7,9,11}. El mismo conjunto puede representarse mediante un árbol de varias maneras diferentes. Lo único que requerimos para una representación válida es que todos los elementos del subárbol izquierdo sean más pequeños que la entrada del nodo y que todos los elementos del subárbol derecho sean más grandes.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.16:**   Varios árboles binarios que representan el conjunto { 1,3,5,7,9,11 }. |

La ventaja de la representación en árbol es la siguiente: supongamos que queremos comprobar si un número *x* está contenido en un conjunto. Empezamos comparando *x* con la entrada en el nodo superior. Si *x* es menor que esto, sabemos que sólo necesitamos buscar en el subárbol izquierdo; si *x* es mayor, sólo necesitamos buscar en el subárbol derecho. Ahora bien, si el árbol está "equilibrado", cada uno de estos subárboles tendrá aproximadamente la mitad del tamaño del original. Por tanto, en un paso hemos reducido el problema de buscar en un árbol de tamaño *n* a buscar en un árbol de tamaño *n* /2. Puesto que el tamaño del árbol se reduce a la mitad en cada paso, deberíamos esperar que el número de pasos necesarios para buscar en un árbol de tamaño *n* crezca como ( log *n* ). [38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_248) Para conjuntos grandes, esto supondrá una aceleración significativa con respecto a las representaciones anteriores.

Podemos representar árboles mediante listas. Cada nodo será una lista de tres elementos: la entrada en el nodo, el subárbol izquierdo y el subárbol derecho. Un subárbol izquierdo o derecho de la lista vacía indicará que no hay ningún subárbol conectado allí. Podemos describir esta representación mediante los siguientes procedimientos: [39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_249)

(definir (árbol de entrada) (árbol de coche))  
(definir (árbol de rama izquierda) (árbol cadr))  
(definir (árbol de rama derecha) (árbol caddr))  
(define (make-tree entrada izquierda derecha)   
  (lista entrada izquierda derecha))

Ahora podemos escribir el procedimiento ¿elemento del conjunto? utilizando la estrategia descrita anteriormente:

(define (elemento-del-conjunto? x conjunto)   
  (cond ((nulo? conjunto) falso)   
        ((= x (conjunto de entrada)) verdadero)   
        ((< x (conjunto de entrada))   
         (elemento-del-conjunto? x (conjunto de la rama izquierda)))   
        ((> x (conjunto de entrada))   
         (elemento-del-conjunto? x (conjunto de la rama derecha)))))

La anexión de un elemento a un conjunto se implementa de manera similar y también requiere ( log *n* ) pasos. Para anexar un elemento x , comparamos x con la entrada del nodo para determinar si x debe agregarse a la rama derecha o izquierda y, una vez anexado x a la rama apropiada, unimos esta rama recién construida con la entrada original y la otra rama. Si x es igual a la entrada, simplemente devolvemos el nodo. Si se nos pide que anexemos x a un árbol vacío, generamos un árbol que tiene x como entrada y ramas derecha e izquierda vacías. Este es el procedimiento:

(define (conjunto-adjunto x conjunto)   
  (cond ((conjunto? nulo) (árbol-hacer x '() '()))   
        ((= x (conjunto de entrada)) conjunto)   
        ((< x (conjunto de entrada))   
         (árbol-hacer (conjunto de entrada)    
                    (conjunto-adjunto x (conjunto de rama izquierda))   
                    (conjunto de rama derecha)))   
        ((> x (conjunto de entrada))   
         (árbol-hacer (conjunto de entrada)   
                    (conjunto de rama izquierda)   
                    (conjunto-adjunto x (conjunto de rama derecha))))))

La afirmación anterior de que la búsqueda en el árbol se puede realizar en un número logarítmico de pasos se basa en el supuesto de que el árbol es"equilibrado", es decir, que el subárbol izquierdo y el derecho de cada árbol tienen aproximadamente el mismo número de elementos, de modo que cada subárbol contiene aproximadamente la mitad de los elementos de su padre. Pero, ¿cómo podemos estar seguros de que los árboles que construimos estarán equilibrados? Incluso si comenzamos con un árbol equilibrado, agregar elementos con un conjunto adjunto puede producir un resultado desequilibrado. Dado que la posición de un elemento recién adjunto depende de cómo se compara el elemento con los elementos que ya están en el conjunto, podemos esperar que si agregamos elementos "al azar" el árbol tenderá a estar equilibrado en promedio. Pero esto no es una garantía. Por ejemplo, si comenzamos con un conjunto vacío y unimos los números del 1 al 7 en secuencia, terminamos con el árbol altamente desequilibrado que se muestra en la figura  [2.17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.17) . En este árbol, todos los subárboles izquierdos están vacíos, por lo que no tiene ninguna ventaja sobre una lista ordenada simple. Una forma de resolver este problema es definir una operación que transforme un árbol arbitrario en un árbol equilibrado con los mismos elementos. Luego podemos realizar esta transformación después de cada pocas operaciones de conjunto adjunto para mantener nuestro conjunto en equilibrio. También existen otras formas de resolver este problema, la mayoría de las cuales implican el diseño de nuevas estructuras de datos para las que la búsqueda y la inserción se pueden realizar en pasos ( log *n ).* [40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_250)

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.17:**   Árbol desequilibrado producido al unir los números 1 a 7 en secuencia. |

**Ejercicio 2.63.**   Cada uno de los dos procedimientos siguientes convierte unárbol binario a una lista.

(define (árbol->lista-1 árbol)   
  (if (null? árbol)   
      '()   
      (append (árbol->lista-1 (árbol de rama izquierda))   
              (cons (árbol de entrada)   
                    (árbol->lista-1 (árbol de rama derecha))))))   
(define (árbol->lista-2 árbol)   
  (define (copiar-a-lista árbol lista-resultados)   
    (if (null? árbol)   
        lista-resultados   
        (copiar-a-lista (árbol de rama izquierda)   
                      (cons (árbol de entrada)   
                            (copiar-a-lista (árbol de rama derecha)   
                                          lista-resultados)))))   
  (copiar-a-lista árbol '()))

a. ¿Ambos procedimientos producen el mismo resultado para cada árbol? Si no es así, ¿en qué difieren los resultados? ¿Qué listas producen los dos procedimientos para los árboles de la figura  [2.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.16) ?

b. ¿Los dos procedimientos tienen el mismo orden de crecimiento en el número de pasos necesarios para convertir un árbol balanceado con *n* elementos en una lista? Si no es así, ¿cuál crece más lentamente?

**Ejercicio 2.64.**  El siguiente procedimiento list->tree convierte una lista ordenada en un árbol binario balanceado. El procedimiento auxiliar partial-tree toma como argumentos un entero *n* y una lista de al menos *n* elementos y construye un árbol balanceado que contiene los primeros *n* elementos de la lista. El resultado devuelto por partial-tree es un par (formado con cons ) cuyo car es el árbol construido y cuyo cdr es la lista de elementos no incluidos en el árbol.

(define (lista->árbol elementos)   
  (carro (árbol parcial elementos (longitud elementos))))   
  
(define (árbol parcial elts n)   
  (si (= n 0)   
      (cons '() elts)   
      (let ((tamaño-izquierdo (cociente (- n 1) 2)))   
        (let ((resultado-izquierdo (árbol parcial elts tamaño-izquierdo)))   
          (let ((árbol-izquierdo (carro resultado-izquierdo))   
                (no-izquierdo-elts (cdr resultado-izquierdo))   
                (tamaño-derecho (- n (+ tamaño-izquierdo 1))))   
            (let ((esta-entrada (carro no-izquierdo-elts))   
                  (resultado-derecho (árbol-parcial (cdr no-izquierdo-elts)   
                                              tamaño-derecho)))   
              (let ((árbol-derecho (carro resultado-derecho))   
                    (resultado-restante (cdr resultado-derecho)))   
                (cons (hacer-árbol esta-entrada árbol-izquierdo árbol-derecho)   
                      restantes-elts)))))))))

a. Escribe un párrafo breve que explique de la forma más clara posible cómo funciona el árbol parcial . Dibuja el árbol producido por list->tree para la lista (1 3 5 7 9 11) .

b. ¿Cuál es el orden de crecimiento en el número de pasos requeridos por list->tree para convertir una lista de *n* elementos?

**Ejercicio 2.65.**  Utilice los resultados de los ejercicios  [2.63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.63) y   [2.64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.64) para dar ( *n* ) implementaciones de conjuntos de unión y conjuntos de intersección para conjuntos implementados como árboles binarios (balanceados). [41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_254)

**[Conjuntos y recuperación de información](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_255)**

Hemos examinado las opciones para utilizar listas para representar conjuntos y hemos visto cómo la elección de la representación de un objeto de datos puede tener un gran impacto en el rendimiento de los programas que utilizan los datos. Otra razón para concentrarnos en los conjuntos es que las técnicas que se analizan aquí aparecen una y otra vez en aplicaciones que implican la recuperación de información.

Consideremos una base de datos que contiene una gran cantidad de registros individuales,como los archivos de personal de una empresa o las transacciones en un sistema de contabilidad. Un sistema de gestión de datos típico dedica una gran cantidad de tiempo a acceder o modificar los datos de los registros y, por lo tanto, requiere un método eficiente para acceder a los registros. Esto se hace identificando una parte de cada registro que sirva como un identificador*clave* . Una clave puede ser cualquier cosa que identifique de forma única el registro. En el caso de un archivo de personal, puede ser el número de identificación de un empleado. En el caso de un sistema de contabilidad, puede ser un número de transacción. Cualquiera que sea la clave, cuando definimos el registro como una estructura de datos, debemos incluir unaprocedimiento selector de claves que recupera la clave asociada a un registro determinado.

Ahora representamos la base de datos como un conjunto de registros. Para localizar el registro con una clave dada utilizamos un procedimiento lookup , que toma como argumentos una clave y una base de datos y que devuelve el registro que tiene esa clave, o false si no existe dicho registro. Lookup se implementa casi de la misma manera que element-of-set? . Por ejemplo, si el conjunto de registros se implementa como una lista desordenada, podríamos utilizar

(define (lookup clave-dada conjunto-de-registros)   
  (cond ((null? conjunto-de-registros) false)   
        ((equal? ​​clave-dada (key (car conjunto-de-registros)))   
         (car conjunto-de-registros))   
        (else (lookup clave-dada (cdr conjunto-de-registros)))))

Por supuesto, hay mejores formas de representar conjuntos grandes que como listas desordenadas. Los sistemas de recuperación de información en los que se debe acceder a los registros de forma "aleatoria" se implementan normalmente mediante un método basado en árboles, como la representación en árbol binario que se analizó anteriormente. Al diseñar un sistema de este tipo, la metodología de abstracción de datos puede ser de gran ayuda. El diseñador puede crear una implementación inicial utilizando una representación sencilla y directa, como listas desordenadas. Esto no será adecuado para el sistema final, pero puede ser útil para proporcionar una base de datos "rápida y sencilla" con la que probar el resto del sistema. Más adelante, la representación de los datos se puede modificar para que sea más sofisticada. Si se accede a la base de datos en términos de selectores y constructores abstractos, este cambio en la representación no requerirá ningún cambio en el resto del sistema.

**Ejercicio 2.66.**   Implementar el procedimiento de búsqueda para el caso donde el conjunto de registros está estructurado como un árbol binario, ordenado por los valores numéricos de las claves.

**[2.3.4 Ejemplo: árboles de codificación de Huffman](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.3.4)**

Esta sección proporciona práctica en el uso de la estructura de listas y la abstracción de datos para manipular conjuntos y árboles. La aplicación es a métodos para representar datos como secuencias de unos y ceros (bits). Por ejemplo,El código estándar ASCII utilizado para representar texto en computadoras codifica cadacarácter como una secuencia de siete bits. El uso de siete bits nos permite distinguir 2 7 , o 128, posibles caracteres diferentes. En general, si queremos distinguir *n* símbolos diferentes, necesitaremos utilizar log 2 *n* bits por símbolo. Si todos nuestros mensajes están compuestos por los ocho símbolos A, B, C, D, E, F, G y H, podemos elegir un código con tres bits por carácter, por ejemplo

|  |  |  |  |
| --- | --- | --- | --- |
| Un 000 | C 010 | E100 | G110 |
| B 001 | D 011 | F 101 | H111 |

Con este código, el mensaje

BACADAEAFABBAAAGAH

se codifica como una cadena de 54 bits

001000010000011000100000101000001001000000000110000111

Los códigos como ASCII y el código A a H anteriores se conocen como*códigos de longitud fija* , porque representan cada símbolo del mensaje con el mismo número de bits. A veces es ventajoso utilizar*códigos de longitud variable* , en los que distintos símbolos pueden representarse mediante distintos números de bits. Por ejemplo,El código Morse no utiliza el mismo número de puntos y rayas para cada letra del alfabeto. En particular, la letra E, la más frecuente, se representa con un solo punto. En general, si nuestros mensajes son tales que algunos símbolos aparecen con mucha frecuencia y otros muy raramente, podemos codificar los datos de manera más eficiente (es decir, utilizando menos bits por mensaje) si asignamos códigos más cortos a los símbolos frecuentes. Considere el siguiente código alternativo para las letras A a H:

|  |  |  |  |
| --- | --- | --- | --- |
| Un 0 | C 1010 | E1100 | G 1110 |
| B100 | D 1011 | F 1101 | H1111 |

Con este código, el mismo mensaje que el anterior se codifica como la cadena

100010100101101100011010100100000111001111

Esta cadena contiene 42 bits, por lo que ahorra más del 20% en espacio en comparación con el código de longitud fija que se muestra arriba.

Una de las dificultades de utilizar un código de longitud variable es saber cuándo se ha llegado al final de un símbolo al leer una secuencia de ceros y unos. El código Morse resuelve este problema utilizando un código especial*código separador* (en este caso, una pausa) después de la secuencia de puntos y rayas para cada letra. Otra solución es diseñar el código de tal manera que ningún código completo para ningún símbolo sea el comienzo (o *prefijo* ) del código para otro símbolo. Este tipo de código se denomina*Código de prefijo* . En el ejemplo anterior, A está codificado con 0 y B con 100, por lo que ningún otro símbolo puede tener un código que comience con 0 o con 100.

En general, podemos lograr ahorros significativos si utilizamos códigos de prefijo de longitud variable que aprovechen las frecuencias relativas de los símbolos en los mensajes que se van a codificar. Un esquema particular para lograr esto se denomina método de codificación de Huffman, en honor a su descubridor,David Huffman. Un código de Huffman se puede representar como unárbol binario cuyas hojas son los símbolos que se codifican. En cada nodo no hoja del árbol hay un conjunto que contiene todos los símbolos de las hojas que se encuentran debajo del nodo. Además, a cada símbolo de una hoja se le asigna un peso (que es su frecuencia relativa), y cada nodo no hoja contiene un peso que es la suma de todos los pesos de las hojas que se encuentran debajo de él. Los pesos no se utilizan en el proceso de codificación o decodificación. Veremos a continuación cómo se utilizan para ayudar a construir el árbol.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.18:**   Un árbol de codificación de Huffman. |

La figura  [2.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.18) muestra el árbol de Huffman para el código de la A a la H indicado anteriormente. Los pesos en las hojas indican que el árbol fue diseñado para mensajes en los que A aparece con una frecuencia relativa de 8, B con una frecuencia relativa de 3 y las demás letras con una frecuencia relativa de 1.

Dado un árbol de Huffman, podemos encontrar la codificación de cualquier símbolo comenzando en la raíz y bajando hasta llegar a la hoja que contiene el símbolo. Cada vez que bajamos por una rama izquierda, añadimos un 0 al código, y cada vez que bajamos por una rama derecha, añadimos un 1. (Decidimos qué rama seguir probando para ver qué rama es el nodo hoja del símbolo o contiene el símbolo en su conjunto). Por ejemplo, comenzando desde la raíz del árbol de la figura  [2.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.18) , llegamos a la hoja de D siguiendo una rama derecha, luego una rama izquierda, luego una rama derecha, luego una rama derecha; por tanto, el código para D es 1011.

Para decodificar una secuencia de bits mediante un árbol de Huffman, comenzamos en la raíz y usamos los ceros y unos sucesivos de la secuencia de bits para determinar si nos movemos hacia abajo por la rama izquierda o derecha. Cada vez que llegamos a una hoja, hemos generado un nuevo símbolo en el mensaje, momento en el cual comenzamos de nuevo desde la raíz del árbol para encontrar el siguiente símbolo. Por ejemplo, supongamos que nos dan el árbol anterior y la secuencia 10001010. Comenzando en la raíz, nos movemos hacia abajo por la rama derecha (ya que el primer bit de la cadena es 1), luego hacia abajo por la rama izquierda (ya que el segundo bit es 0), luego hacia abajo por la rama izquierda (ya que el tercer bit también es 0). Esto nos lleva a la hoja de B, por lo que el primer símbolo del mensaje decodificado es B. Ahora comenzamos nuevamente en la raíz y hacemos un movimiento hacia la izquierda porque el siguiente bit en la cadena es 0. Esto nos lleva a la hoja de A. Luego comenzamos nuevamente en la raíz con el resto de la cadena 1010, por lo que nos movemos derecha, izquierda, derecha, izquierda y llegamos a C. Por lo tanto, todo el mensaje es BAC.

**[Generando árboles de Huffman](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_257)**

Dado un «alfabeto» de símbolos y sus frecuencias relativas, ¿cómo construimos el «mejor» código? (En otras palabras, ¿qué árbol codificará los mensajes con la menor cantidad de bits?) Huffman presentó un algoritmo para hacer esto y demostró que el código resultante es de hecho el mejor código de longitud variable para mensajes donde la frecuencia relativa de los símbolos coincide con las frecuencias con las que se construyó el código.No probaremos aquí esta optimalidad de los códigos de Huffman, pero mostraremos cómo se construyen los árboles de Huffman. [42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_258)

El algoritmo para generar un árbol de Huffman es muy simple. La idea es organizar el árbol de manera que los símbolos con la frecuencia más baja aparezcan más alejados de la raíz. Comience con el conjunto de nodos de hoja, que contiene los símbolos y sus frecuencias, según lo determinado por los datos iniciales a partir de los cuales se construirá el código. Ahora encuentre dos hojas con los pesos más bajos y únalas para producir un nodo que tenga estos dos nodos como sus ramas izquierda y derecha. El peso del nuevo nodo es la suma de los dos pesos. Retire las dos hojas del conjunto original y reemplácelas por este nuevo nodo. Ahora continúe con este proceso. En cada paso, combine dos nodos con los pesos más bajos, eliminándolos del conjunto y reemplazándolos con un nodo que tenga estos dos como sus ramas izquierda y derecha. El proceso se detiene cuando solo queda un nodo, que es la raíz de todo el árbol. Así es como  se generó el árbol de Huffman de la figura [2.18 :](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.18)

|  |  |
| --- | --- |
| Hojas iniciales | {(A 8) (B 3) (C 1) (D 1) (E 1) (F 1) (G 1) (H 1)} |
| Unir | {(A 8) (B 3) ({CD} 2) (E 1) (F 1) (G 1) (H 1)} |
| Unir | {(A 8) (B 3) ({CD} 2) ({EF} 2) (G 1) (H 1)} |
| Unir | {(A 8) (B 3) ({CD} 2) ({EF} 2) ({GH} 2)} |
| Unir | {(A 8) (B 3) ({CD} 2) ({EFGH} 4)} |
| Unir | {(A 8) ({BCD} 5) ({EFGH} 4)} |
| Unir | {(A 8) ({BCDEFGH} 9)} |
| Fusión final | {({ABCDEFGH} 17)} |
|  |  |

El algoritmo no siempre especifica un árbol único, ya que puede que no haya nodos únicos con el peso más pequeño en cada paso. Además, la elección del orden en el que se fusionan los dos nodos (es decir, cuál será la rama derecha y cuál será la rama izquierda) es arbitraria.

**[Representando árboles de Huffman](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_259)**

En los ejercicios que se presentan a continuación, trabajaremos con un sistema que utiliza árboles de Huffman para codificar y decodificar mensajes y genera árboles de Huffman según el algoritmo descrito anteriormente. Comenzaremos analizando cómo se representan los árboles.

Las hojas del árbol se representan mediante una lista que consta del símbolo hoja , el símbolo en la hoja y el peso:

(define (make-leaf symbol weight)   
  (lista 'leaf symbol weight))  
(define (objeto hoja?)   
  (objeto eq? (coche) 'hoja))  
(define (símbolo-hoja x) (cadr x))  
(define (peso-hoja x) (caddr x))

Un árbol general será una lista de una rama izquierda, una rama derecha, un conjunto de símbolos y un peso. El conjunto de símbolos será simplemente una lista de los símbolos, en lugar de una representación de conjunto más sofisticada. Cuando hacemos un árbol fusionando dos nodos, obtenemos el peso del árbol como la suma de los pesos de los nodos y el conjunto de símbolos como la unión de los conjuntos de símbolos para los nodos. Dado que nuestros conjuntos de símbolos se representan como listas, podemos formar la unión utilizando el procedimiento append que definimos en la sección  [2.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.1) :

(define (make-code-tree left right)   
  (list left   
        right   
        (append (símbolos left) (símbolos right))   
        (+ (peso left) (peso right))))

Si hacemos un árbol de esta manera, tenemos los siguientes selectores:

(definir (árbol de rama izquierda) (árbol de automóviles))  
  
(definir (árbol de rama derecha) (árbol cadr))  
(define (árbol de símbolos)   
  (si (¿hoja? árbol)   
      (lista (árbol de símbolos-hojas))   
      (árbol caddr)))  
(define (peso árbol)   
  (si (hoja? árbol)   
      (peso-hoja árbol)   
      (cadddr árbol)))

Los símbolos y pesos de los procedimientos deben hacer algo ligeramente diferente según se los llame con una hoja o con un árbol general. Estos son ejemplos simples de*procedimientos genéricos* (procedimientos que pueden manejar más de un tipo de datos), sobre los cuales hablaremos mucho más en las secciones  [2.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4) y  [2.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.5) .

**[El procedimiento de decodificación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_260)**

El siguiente procedimiento implementa el algoritmo de decodificación. Toma como argumentos una lista de ceros y unos, junto con un árbol de Huffman.

(define (decodificar bits árbol)   
  (define (decodificar-1 bits rama-actual)   
    (if (null? bits)   
        '()   
        (let ((siguiente-rama   
               (elegir-rama (car bits) rama-actual)))   
          (if (hoja? siguiente-rama)   
              (cons (símbolo-hoja siguiente-rama)   
                    (decodificar-1 (cdr bits) árbol))   
              (decodificar-1 (cdr bits) siguiente-rama)))))   
  (decodificar-1 bits árbol))   
(define (elegir-rama bit rama)   
  (cond ((= bit 0) (rama-izquierda))   
        ((= bit 1) (rama-derecha))   
        (else (error "bit incorrecto -- ELEGIR-RAMA" bit))))

El procedimiento decode-1 toma dos argumentos: la lista de bits restantes y la posición actual en el árbol. Sigue moviéndose ``hacia abajo'' en el árbol, eligiendo una rama izquierda o derecha según si el siguiente bit en la lista es un cero o un uno. (Esto se hace con el procedimiento choose-branch ). Cuando llega a una hoja, devuelve el símbolo en esa hoja como el siguiente símbolo en el mensaje al aplicarle una cons al resultado de decodificar el resto del mensaje, comenzando en la raíz del árbol. Observe la comprobación de errores en la cláusula final de choose-branch , que se queja si el procedimiento encuentra algo distinto de un cero o un uno en los datos de entrada.

**[Conjuntos de elementos ponderados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_261)**

En nuestra representación de árboles, cada nodo que no es una hoja contiene un conjunto de símbolos, que hemos representado como una lista simple. Sin embargo, el algoritmo de generación de árboles que analizamos anteriormente requiere que también trabajemos con conjuntos de hojas y árboles, fusionando sucesivamente los dos elementos más pequeños. Dado que tendremos que encontrar repetidamente el elemento más pequeño de un conjunto, es conveniente utilizar una representación ordenada para este tipo de conjunto.

Representaremos un conjunto de hojas y árboles como una lista de elementos, ordenados en orden creciente de peso. El siguiente procedimiento de conjunto adjunto para construir conjuntos es similar al descrito en el ejercicio  [2.61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.61) ; sin embargo, los elementos se comparan por sus pesos y el elemento que se agrega al conjunto nunca está ya en él.

(define (conjunto contiguo x conjunto)   
  (cond ((conjunto nulo?) (lista x))   
        ((< (peso x) (peso (conjunto carro))) (conjunto cons x))   
        (de lo contrario (conjunto cons (conjunto carro)   
                    (conjunto contiguo x (conjunto cdr))))))

El siguiente procedimiento toma una lista de pares símbolo-frecuencia como ((A 4) (B 2) (C 1) (D 1)) y construye un conjunto inicial ordenado de hojas, listo para ser fusionado de acuerdo con el algoritmo de Huffman:

(define (make-leaf-set pairs)   
  (if (null? pairs)   
      '()   
      (let ((pair (car pairs)))   
        (adjoin-set (make-leaf (car pair)     *; symbol*  
                               (cadr pair))   *; frequency*  
                    (make-leaf-set (cdr pairs)))))

**Ejercicio 2.67.**   Defina un árbol de codificación y un mensaje de ejemplo:

(define árbol-de-muestra   
  (árbol-de-código-make (hoja-make 'A 4)   
                  (árbol-de-código-make   
                   (hoja-make 'B 2)   
                   (árbol-de-código-make (hoja-make 'D 1)   
                                   (hoja-make 'C 1)))))   
  
(define mensaje-de-muestra '(0 1 1 0 0 1 0 1 0 1 1 1 0))

Utilice el procedimiento de decodificación para decodificar el mensaje y proporcionar el resultado.

**Ejercicio 2.68.**   El procedimiento encode toma como argumentos un mensaje y un árbol y produce la lista de bits que da el mensaje codificado.

(define (codificar mensaje árbol)   
  (si (nulo? mensaje)   
      '()   
      (agrega (codificar-símbolo (car mensaje) árbol)   
              (codificar (cdr mensaje) árbol))))

Encode-symbol es un procedimiento que debes escribir y que devuelve la lista de bits que codifica un símbolo determinado según un árbol determinado. Debes diseñar encode-symbol de forma que indique un error si el símbolo no está en el árbol. Prueba el procedimiento codificando el resultado que obtuviste en el ejercicio  [2.67](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.67) con el árbol de muestra y observando si es el mismo que el mensaje de muestra original.

**Ejercicio 2.69.**   El siguiente procedimiento toma como argumento una lista de pares símbolo-frecuencia (donde ningún símbolo aparece en más de un par) y genera un árbol de codificación de Huffman según el algoritmo de Huffman.

(definir (generar-pares-de-árbol-huffman)   
  (fusionar-sucesivamente (crear-pares-de-conjuntos-de-hojas)))

Make-leaf-set es el procedimiento indicado anteriormente que transforma la lista de pares en un conjunto ordenado de hojas. Successive-merge es el procedimiento que debes escribir, utilizando make-code-tree para fusionar sucesivamente los elementos de menor peso del conjunto hasta que solo quede un elemento, que es el árbol de Huffman deseado. (Este procedimiento es un poco complicado, pero no realmente complicado. Si te encuentras diseñando un procedimiento complejo, entonces es casi seguro que estás haciendo algo mal. Puedes aprovechar significativamente el hecho de que estamos utilizando una representación de conjunto ordenado).

**Ejercicio 2.70.**  El siguiente alfabeto de ocho símbolos con frecuencias relativas asociadas fue diseñado para codificar de manera eficiente las letras de las canciones de rock de la década de 1950. (Tenga en cuenta que los "símbolos" de un "alfabeto" no necesitan ser letras individuales).

|  |  |  |  |
| --- | --- | --- | --- |
| A | 2 | N / A | 16 |
| AUGE | 1 | SAH | 3 |
| CONSEGUIR | 2 | ¡Sí! | 9 |
| TRABAJO | 2 | Vaya. | 1 |

Utilice generate-huffman-tree (ejercicio  [2.69](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.69) ) para generar un árbol Huffman correspondiente y utilice encode (ejercicio  [2.68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.68) ) para codificar el siguiente mensaje:

Hacer trabajo

Sha na na na na na na na na

Hacer trabajo

Sha na na na na na na na na

Wah yip yip yip yip yip yip yip yip yip

El auge del sha

¿Cuántos bits se necesitan para la codificación? ¿Cuál es la cantidad mínima de bits que se necesitaría para codificar esta canción si utilizáramos un código de longitud fija para el alfabeto de ocho símbolos?

**Ejercicio 2.71.**   Supongamos que tenemos un árbol de Huffman para un alfabeto de *n* símbolos, y que las frecuencias relativas de los símbolos son 1, 2, 4, ... , 2 *n* -1 . Dibuje el árbol para *n* = 5; para *n* = 10. En un árbol de este tipo (para *n* general ), ¿cuántos bits se requieren para codificar el símbolo más frecuente? ¿y el símbolo menos frecuente?

**Ejercicio 2.72.**  Considere el procedimiento de codificación que diseñó en el ejercicio  [2.68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.68) . ¿Cuál es el orden de crecimiento en el número de pasos necesarios para codificar un símbolo? Asegúrese de incluir el número de pasos necesarios para buscar en la lista de símbolos en cada nodo encontrado. Responder a esta pregunta en general es difícil. Considere el caso especial donde las frecuencias relativas de los *n* símbolos son como se describe en el ejercicio  [2.71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.71) , y proporcione el orden de crecimiento (como una función de *n* ) del número de pasos necesarios para codificar los símbolos más frecuentes y menos frecuentes en el alfabeto.

[32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_227) Permitir la cita en un idioma causa estragos en la capacidad de razonar sobre el idioma en términos simples, porque destruye la noción de que los iguales pueden sustituirse por iguales. Por ejemplo, tres es uno más dos, pero la palabra "tres" no es la frase "uno más dos". La cita es poderosa porque nos da una manera de construir expresiones que manipulan otras expresiones (como veremos cuando escribamos un intérprete en el capítulo 4). Pero permitir enunciados en un idioma que hablen de otros enunciados en ese idioma hace que sea muy difícil mantener cualquier principio coherente de lo que debería significar "igual puede sustituirse por igual". Por ejemplo, si sabemos quela estrella de la tarde es la estrella de la mañana, entonces de la afirmación ``la estrella de la tarde es Venus'' podemos deducir ``la estrella de la mañana es Venus''. Sin embargo, dado que ``Juan sabe que la estrella de la tarde es Venus'' no podemos inferir que ``Juan sabe que la estrella de la mañana es Venus''.

[33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_228) La comilla simple es diferenteDe las comillas dobles que hemos estado usando para encerrar cadenas de caracteres que se van a imprimir. Mientras que las comillas simples se pueden usar para indicar listas o símbolos, las comillas dobles se usan solo con cadenas de caracteres. En este libro, el único uso de las cadenas de caracteres es como elementos que se van a imprimir.

[34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_229) Estrictamente, nuestro uso de las comillas viola la regla general de que todas las expresiones compuestas en nuestro idioma deben estar delimitadas por paréntesis y parecer listas.Podemos recuperar esta consistencia introduciendo una forma especial quote , que cumple la misma función que las comillas. Por lo tanto, escribiríamos (quote a) en lugar de 'a , y escribiríamos (quote (abc)) en lugar de '(abc) . Así es precisamente como funciona el intérprete. Las comillas son solo una abreviatura de un solo carácter para envolver la siguiente expresión completa con comillas para formar (quote < *expresión* >) . Esto es importante porque mantiene el principio de que cualquier expresión vista por el intérprete puede manipularse como un objeto de datos. Por ejemplo, podríamos construir la expresión (car '(a b c)) , que es la misma que (car (quote (abc))) , evaluando (list 'car (list 'quote '(a b c))) .

[Podemos considerar que](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_230) dos símbolos son «iguales» si están compuestos por los mismos caracteres en el mismo orden. Esta definición evita una cuestión profunda que todavía no estamos preparados para abordar: el significado de «igualdad» en un lenguaje de programación. Volveremos a este tema en el capítulo 3 (sección [3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.3)).

[36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_233) En la práctica, los programadores usanequal?​​para comparar listas que contienen números y símbolos. Los números no se consideran símbolos. La preguntaLa cuestión de si dos números numéricamente iguales (como se prueba con = ) también son iguales depende en gran medida de la implementación. Una mejor definición de igual? (como la que viene como primitiva en Scheme) también estipularía que si a y b son ambos números, entonces a y b son iguales? si son numéricamente iguales.

[37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_240) Si queremos ser más formales, podemos especificar «consistente con las interpretaciones dadas anteriormente» para significar que las operaciones satisfacen una colección de reglas como estas:

* Para cualquier conjunto S y cualquier objeto x , (elemento-del-conjunto? x (conjunto-contiguo x S)) es verdadero (informalmente: ``Unir un objeto a un conjunto produce un conjunto que contiene el objeto'').
* Para cualesquiera conjuntos S y T y cualquier objeto x , (elemento-del-conjunto? x (conjunto-unión ST)) es igual a (o (elemento-del-conjunto? x S) (elemento-del-conjunto? x T)) (informalmente: ``Los elementos de (conjunto-unión ST) son los elementos que están en S o en T '').
* Para cualquier objeto x , (elemento-del-conjunto? x '()) es falso (informalmente: "Ningún objeto es un elemento del conjunto vacío").

[Reducir a](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_248) la mitad el tamaño del problema en cada paso es la característica distintiva decrecimiento logarítmico, como vimos con el algoritmo de exponenciación rápida de la sección  [1.2.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.4) y el método de búsqueda de medio intervalo de la sección  [1.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.3) .

[39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_249) Representamos conjuntos en términos de árboles y árboles en términos de listas: en efecto, una abstracción de datos construida sobre una abstracción de datos. Podemos considerar los procedimientosentry,left-branch,right-branchymake-treecomo una forma de aislar la abstracción de un ``árbol binario'' de la forma particular en que podríamos desear representar dicho árbol en términos de estructura de lista.

[40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_250) Ejemplos de tales estructuras incluyen*Árboles B* y *árboles rojo-negros* . Existe una gran cantidad de literatura sobre estructuras de datos dedicada a este problema. Véase Cormen,Leiserson y Rivest 1990.

[41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_254) Ejercicios [2.63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.63)-[2.65](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.65) Se deben a Paul Hilfinger.

[42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_258) Véase Hamming 1980para una discusión de las propiedades matemáticas de los códigos de Huffman.

[**2.4 Representaciones múltiples para datos abstractos**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_2.4)

Hemos introducido la abstracción de datos, una metodología para estructurar sistemas de tal manera que gran parte de un programa se pueda especificar independientemente de las opciones involucradas en la implementación de los objetos de datos que el programa manipula. Por ejemplo, vimos en la sección  [2.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.1) cómo separar la tarea de diseñar un programa que use números racionales de la tarea de implementar números racionales en términos de los mecanismos primitivos del lenguaje de computadora para construir datos compuestos. La idea clave era erigir una barrera de abstracción - en este caso, los selectores y constructores para números racionales ( make-rat , numer , denom ) - que aísla la forma en que se usan los números racionales de su representación subyacente en términos de estructura de lista. Una barrera de abstracción similar aísla los detalles de los procedimientos que realizan aritmética racional ( add-rat , sub-rat , mul-rat y div-rat ) de los procedimientos de "nivel superior" que usan números racionales. El programa resultante tiene la estructura que se muestra en la figura  [2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.1) .

Estas barreras de abstracción de datos son herramientas poderosas para controlar la complejidad. Al aislar las representaciones subyacentes de los objetos de datos, podemos dividir la tarea de diseñar un programa grande en tareas más pequeñas que se pueden realizar por separado. Pero este tipo de abstracción de datos aún no es lo suficientemente poderosa, porque puede que no siempre tenga sentido hablar de "la representación subyacente" de un objeto de datos.

Por un lado, puede haber más de una representación útil para un objeto de datos, y tal vez queramos diseñar sistemas que puedan manejar múltiples representaciones. Para tomar un ejemplo simple, los números complejos pueden representarse de dos maneras casi equivalentes: en forma rectangular (partes reales e imaginarias) y en forma polar (magnitud y ángulo). A veces la forma rectangular es más apropiada y a veces la forma polar es más apropiada. De hecho, es perfectamente plausible imaginar un sistema en el que los números complejos se representen de ambas maneras, y en el que los procedimientos para manipular números complejos funcionen con cualquiera de las dos representaciones.

Más importante aún, los sistemas de programación suelen ser diseñados por muchas personas que trabajan durante largos períodos de tiempo, sujetos a requisitos que cambian con el tiempo. En un entorno así, simplemente no es posible que todos se pongan de acuerdo de antemano sobre las opciones de representación de los datos. Por lo tanto, además de las barreras de abstracción de datos que aíslan la representación del uso, necesitamos barreras de abstracción que aíslen las diferentes opciones de diseño entre sí y permitan que coexistan diferentes opciones en un solo programa. Además, dado que los programas grandes a menudo se crean combinando módulos preexistentes que se diseñaron de forma aislada, necesitamos convenciones que permitan a los programadores incorporar módulos en sistemas más grandes.*aditivamente* , es decir, sin tener que rediseñar o reimplementar estos módulos.

En esta sección, aprenderemos a manejar datos que pueden representarse de distintas maneras en distintas partes de un programa. Esto requiere construir*procedimientos genéricos* : procedimientos que pueden operar sobre datos que pueden representarse de más de una manera. Nuestra técnica principal para crear procedimientos genéricos será trabajar en términos de objetos de datos que tengan*etiquetas de tipo* , es decir, objetos de datos que incluyen información explícita sobre cómo se deben procesar. También analizaremos*Programación dirigida por datos* , una estrategia de implementación poderosa y conveniente para ensamblar de forma aditiva sistemas con operaciones genéricas.

Comenzaremos con el ejemplo simple de los números complejos. Veremos cómo las etiquetas de tipo y el estilo orientado a los datos nos permiten diseñar representaciones rectangulares y polares independientes para números complejos, manteniendo al mismo tiempo la noción de un objeto de datos de "número complejo" abstracto.Lograremos esto definiendo procedimientos aritméticos para números complejos ( add-complex , sub-complex , mul-complex y div-complex ) en términos de selectores genéricos que acceden a partes de un número complejo independientemente de cómo se represente el número. El sistema de números complejos resultante, como se muestra en la figura  [2.19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.19) , contiene dos tipos diferentes deBarreras de abstracción. Las barreras de abstracción «horizontales» desempeñan el mismo papel que las de la figura  [2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.1) . Aíslan las operaciones de «nivel superior» de las representaciones de «nivel inferior». Además, existe una barrera «vertical» que nos da la capacidad de diseñar e instalar por separado representaciones alternativas.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.19:**   Barreras de abstracción de datos en el sistema de números complejos. |

En la sección  [2.5,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.5) mostraremos cómo utilizar etiquetas de tipo y un estilo dirigido a datos para desarrollar un paquete aritmético genérico. Esto proporciona procedimientos ( add , mul , etc.) que se pueden utilizar para manipular todo tipo de «números» y se pueden ampliar fácilmente cuando se necesita un nuevo tipo de número. En la sección  [2.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.5.3) , mostraremos cómo utilizar la aritmética genérica en un sistema que realiza álgebra simbólica.

**[2.4.1 Representaciones para números complejos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.4.1)**

Desarrollaremos un sistema que realice operaciones aritméticas con números complejos como un ejemplo simple pero poco realista de un programa que utiliza operaciones genéricas. Comenzaremos analizando dos representaciones plausibles para números complejos como pares ordenados: forma rectangular (parte real y parte imaginaria) y forma polar (magnitud y ángulo). [43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_268) La sección  [2.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.2) mostrará cómo se puede hacer que ambas representaciones coexistan en un solo sistema mediante el uso de etiquetas de tipo y operaciones genéricas.

Al igual que los números racionales, los números complejos se representan naturalmente como pares ordenados. El conjunto de números complejos puede considerarse como un espacio bidimensional con dos ejes ortogonales, el eje «real» y el eje «imaginario». (Véase la figura  [2.20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.20) .) Desde este punto de vista, el número complejo *z* = *x* + *i y* (donde *i* 2 = - 1) puede considerarse como el punto en el plano cuya coordenada real es *x* y cuya coordenada imaginaria es *y* . La suma de números complejos se reduce en esta representación a la suma de coordenadas:





Al multiplicar números complejos, es más natural pensar en términos de representar un número complejo en forma polar, como una magnitud y un ángulo ( *r* y *A* en la figura  [2.20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.20) ). El producto de dos números complejos es el vector que se obtiene al estirar un número complejo por la longitud del otro y luego rotarlo a través del ángulo del otro:





Por lo tanto, existen dos representaciones diferentes para los números complejos, que son apropiadas para diferentes operaciones. Sin embargo, desde el punto de vista de alguien que escribe un programa que utiliza números complejos, el principio de abstracción de datos sugiere que todas las operaciones para manipular números complejos deberían estar disponibles independientemente de la representación que utilice el ordenador. Por ejemplo, a menudo resulta útil poder encontrar la magnitud de un número complejo que se especifica mediante coordenadas rectangulares. De manera similar, suele resultar útil poder determinar la parte real de un número complejo que se especifica mediante coordenadas polares.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.20:**   Números complejos como puntos en el plano. |

Para diseñar un sistema de este tipo, podemos seguir el mismo procedimiento:Estrategia de abstracción de datos que seguimos al diseñar el paquete de números racionales en la sección  [2.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.1) . Suponga que las operaciones con números complejos se implementan en términos de cuatro selectores: parte real , parte imag , magnitud y ángulo . Suponga también que tenemos dos procedimientos para construir números complejos: make-from-real-imag devuelve un número complejo con partes reales e imaginarias especificadas, y make-from-mag-ang devuelve un número complejo con magnitud y ángulo especificados. Estos procedimientos tienen la propiedad de que, para cualquier número complejo z , tanto

(hacer-a-partir-de-imag-real (parte-real-z) (parte-imag-z))

y

(hacer desde-mag-ang (magnitud z) (ángulo z))

producir números complejos que sean iguales a z .

Usando estos constructores y selectores, podemos implementar aritmética en números complejos usando los "datos abstractos" especificados por los constructores y selectores, tal como lo hicimos para los números racionales en la sección  [2.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.1) . Como se muestra en las fórmulas anteriores, podemos sumar y restar números complejos en términos de partes reales e imaginarias mientras multiplicamos y dividimos números complejos en términos de magnitudes y ángulos:

(define (add-complex z1 z2)   
  (make-from-real-imag (+ (real-parte z1) (real-parte z2))   
                       (+ (imag-parte z1) (imag-parte z2))))  
(define (subcomplejo z1 z2)   
  (make-from-real-imag (- (parte-real z1) (parte-real z2))   
                       (- (parte-imag z1) (parte-imag z2))))  
(define (multi-complejo z1 z2)   
  (make-from-mag-ang (\* (magnitud z1) (magnitud z2))   
                     (+ (ángulo z1) (ángulo z2))))  
(define (div-complejo z1 z2)   
  (make-from-mag-ang (/ (magnitud z1) (magnitud z2))   
                     (- (ángulo z1) (ángulo z2))))

Para completar el paquete de números complejos, debemos elegir una representación y debemos implementar los constructores y selectores en términos de números primitivos y estructura de lista primitiva. Hay dos formas obvias de hacer esto: podemos representar un número complejo en "forma rectangular" como un par (parte real, parte imaginaria) o en "forma polar" como un par (magnitud, ángulo). ¿Cuál elegiremos?

Para concretar las diferentes opciones, imaginemos que hay dos programadores, Ben Bitdiddle y Alyssa P. Hacker, que diseñan independientemente representaciones para el sistema de números complejos.Ben elige representar los números complejos en forma rectangular. Con esta opción, seleccionar las partes reales e imaginarias de un número complejo es sencillo, al igual que construir un número complejo con partes reales e imaginarias dadas. Para encontrar la magnitud y el ángulo, o para construir un número complejo con una magnitud y un ángulo dados, utiliza las relaciones trigonométricas.





que relacionan las partes reales e imaginarias ( *x* , *y* ) con la magnitud y el ángulo ( *r* , *A* ). [44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_269) Por lo tanto, la representación de Ben viene dada por los siguientes selectores y constructores:

(define (parte real z) (automóvil z))  
(define (imag-parte z) (cdr z))  
(define (magnitud z)   
  (sqrt (+ (cuadrado (parte real z)) (cuadrado (parte imag z)))))  
(definir (ángulo z)   
  (atan (imag-part z) (real-part z)))  
(definir (hacer-a-partir-de-imagen-real x y) (cons x y))  
(define (hace-a-partir-de-mag-ang r a)    
  (cons (\* r (cos a)) (\* r (sin a))))

Alyssa, por el contrario, elige representar números complejos en forma polar. Para ella, seleccionar la magnitud y el ángulo es sencillo, pero tiene que usar laRelaciones trigonométricas para obtener las partes reales e imaginarias. La representación de Alyssa es:

(define (parte real z)   
  (\* (magnitud z) (cos (ángulo z))))  
(define (imag-parte z)   
  (\* (magnitud z) (sin (ángulo z))))  
(definir (magnitud z) (automóvil z))  
(define (ángulo z) (cdr z))  
(define (make-from-real-imag x y)    
  (cons (sqrt (+ (cuadrado x) (cuadrado y)))   
        (atan y x)))  
(definir (hacer-a-partir-de-mag-ang r a) (cons r a))

La disciplina de abstracción de datos garantiza que la misma implementación de add-complex , sub-complex , mul-complex y div-complex funcionará con la representación de Ben o con la representación de Alyssa.

**[2.4.2 Datos etiquetados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.4.2)**

Una forma de ver la abstracción de datos es como una aplicación de la``principio de mínimo compromiso''. Al implementar el sistema de números complejos de la sección  [2.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.1) , podemos utilizar la representación rectangular de Ben o la representación polar de Alyssa. La barrera de abstracción formada por los selectores y constructores nos permite aplazar hasta el último momento posible la elección de una representación concreta para nuestros objetos de datos y, de esta manera, conservar la máxima flexibilidad en el diseño de nuestro sistema.

El principio del mínimo compromiso puede llevarse a extremos aún más extremos. Si lo deseamos, podemos mantener la ambigüedad de la representación incluso *después de* haber diseñado los selectores y constructores, y elegir utilizar tanto la representación de Ben *como* la de Alyssa. Sin embargo, si ambas representaciones están incluidas en un único sistema, necesitaremos alguna forma de distinguir los datos en forma polar de los datos en forma rectangular. De lo contrario, si se nos pidiera, por ejemplo, encontrar la magnitud del par (3,4), no sabríamos si responder 5 (interpretando el número en forma rectangular) o 3 (interpretando el número en forma polar). Una forma sencilla de lograr esta distinción es incluir una*Etiqueta de tipo* (el símbolo rectangular o polar ) como parte de cada número complejo. Luego, cuando necesitamos manipular un número complejo, podemos usar la etiqueta para decidir qué selector aplicar.

Para manipular datos etiquetados, supondremos que tenemos los procedimientos type-tag y contents que extraen de un objeto de datos la etiqueta y el contenido real (las coordenadas polares o rectangulares, en el caso de un número complejo). También postularemos un procedimiento join-tag que toma una etiqueta y el contenido y produce un objeto de datos etiquetado. Una forma sencilla de implementar esto es usar la estructura de lista común:

(define (adjuntar etiqueta tipo contenido)   
  (cons etiqueta tipo contenido))  
(define (dato de etiqueta de tipo)   
  (si (dato de par?)   
      (dato de automóvil)   
      (error "Dato de etiqueta incorrecta -- dato de ETIQUETA DE TIPO")))  
(define (dato de contenido)   
  (si (dato de par?)   
      (dato de cdr)   
      (error "Dato mal etiquetado -- dato de CONTENIDO")))

Utilizando estos procedimientos, podemos definir predicados rectangulares? y polares?, que reconocen números polares y rectangulares, respectivamente:

(define (rectangular? z)   
  (eq? (tipo-etiqueta z) 'rectangular))  
(define (polar? z)   
  (eq? (tipo-etiqueta z) 'polar))

Con las etiquetas de tipo, Ben y Alyssa ahora pueden modificar su código para que sus dos representaciones diferentes puedan coexistir en el mismo sistema. Siempre que Ben construye un número complejo, lo etiqueta como rectangular. Siempre que Alyssa construye un número complejo, lo etiqueta como polar. Además, Ben y Alyssa deben asegurarse de que los nombres de sus procedimientos no entren en conflicto. Una forma de hacer esto es que Ben agregue el sufijo rectangular al nombre de cada uno de sus procedimientos de representación y que Alyssa agregue polar a los nombres de los suyos. Aquí está la representación rectangular revisada de Ben de la sección  [2.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.1) :

(define (parte-real-rectangular z) (automóvil z))  
(define (imag-parte-rectangular z) (cdr z))  
(define (magnitud-rectangular z)   
  (sqrt (+ (cuadrado (parte-real-rectangular z))   
           (cuadrado (parte-imag-rectangular z)))))  
(define (ángulo-rectangular z)   
  (atan (imag-parte-rectangular z)   
        (real-parte-rectangular z)))  
(define (make-from-real-imag-rectangular x y)   
  (adjunta-etiqueta 'rectangular (cons x y)))  
(define (make-from-mag-ang-rectangular r a)    
  (attach-tag 'rectangular   
              (cons (\* r (cos a)) (\* r (sin a)))))

Y aquí está la representación polar revisada de Alyssa:

(define (z polar en la parte real)   
  (\* (z polar en la magnitud) (cos (z polar en el ángulo))))  
(define (imag-parte-polar z)   
  (\* (magnitud-polar z) (sin (ángulo-polar z))))  
(definir (magnitud-polar z) (automóvil z))  
(define (z angular-polar) (cdr z))  
(define (make-from-real-imag-polar x y)    
  (attach-tag 'polar   
               (cons (sqrt (+ (cuadrado x) (cuadrado y)))   
                     (atan y x))))  
(define (make-from-mag-ang-polar r a)   
  (adjunta-etiqueta 'polar (cons r a)))

Cada selector genérico se implementa como un procedimiento que verifica la etiqueta de su argumento y llama al procedimiento adecuado para manejar datos de ese tipo. Por ejemplo, para obtener la parte real de un número complejo, real-part examina la etiqueta para determinar si se debe utilizar real-part-rectangular de Ben o real-part-polar de Alyssa . En cualquier caso, utilizamos contents para extraer el dato desnudo, sin etiquetar, y lo enviamos al procedimiento rectangular o polar según sea necesario:

(define (parte-real z)   
  (cond ((rectangular? z)    
         (parte-real-rectangular (contenido z)))   
        ((polar? z)   
         (parte-real-polar (contenido z)))   
        (else (error "Tipo desconocido -- PARTE-REAL" z))))  
(define (imag-part z)   
  (cond ((rectangular? z)   
         (imag-part-rectangular (contenido z)))   
        ((polar? z)   
         (imag-part-polar (contenido z)))   
        (else (error "Tipo desconocido -- IMAG-PART" z))))  
(define (magnitud z)   
  (cond ((rectangular? z)   
         (magnitud-rectangular (contenido z)))   
        ((polar? z)   
         (magnitud-polar (contenido z)))   
        (else (error "Tipo desconocido -- MAGNITUD" z))))  
(define (ángulo z)   
  (cond ((rectangular? z)   
         (ángulo-rectangular (contenido z)))   
        ((polar? z)   
         (ángulo-polar (contenido z)))   
        (else (error "Tipo desconocido -- ÁNGULO" z))))

Para implementar las operaciones aritméticas de números complejos, podemos utilizar los mismos procedimientos add-complex , sub-complex , mul-complex y div-complex de la sección  [2.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.1) , porque los selectores que llaman son genéricos y, por lo tanto, funcionarán con cualquiera de las representaciones. Por ejemplo, el procedimiento add-complex sigue siendo

(define (add-complex z1 z2)   
  (make-from-real-imag (+ (real-parte z1) (real-parte z2))   
                       (+ (imag-parte z1) (imag-parte z2))))

Por último, debemos elegir si construimos números complejos utilizando la representación de Ben o la representación de Alyssa. Una opción razonable es construir números rectangulares siempre que tengamos partes reales e imaginarias y construir números polares siempre que tengamos magnitudes y ángulos:

(define (hacer-a-partir-de-imag-real x y)   
  (hacer-a-partir-de-imag-real-rectangular x y))  
(definir (hacer-desde-mag-ang r a)   
  (hacer-desde-mag-ang-polar r a))

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.21:**   Estructura del sistema aritmético complejo genérico. |

El sistema de números complejos resultante tiene la estructura que se muestra en la figura  [2.21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.21) . El sistema se ha descompuesto en tres partes relativamente independientes: las operaciones aritméticas de números complejos, la implementación polar de Alyssa y la implementación rectangular de Ben. Las implementaciones polar y rectangular podrían haber sido escritas por Ben y Alyssa trabajando por separado, y ambas pueden ser utilizadas como representaciones subyacentes por un tercer programador que implemente los procedimientos aritméticos complejos en términos de la interfaz abstracta constructor/selector.

Dado que cada objeto de datos está etiquetado con su tipo, los selectores operan sobre los datos de una manera genérica. Es decir, cada selector está definido para tener un comportamiento que depende del tipo particular de datos al que se aplica. Observe el mecanismo general para interconectar las representaciones separadas: dentro de una implementación de representación dada (por ejemplo, el paquete polar de Alyssa), un número complejo es un par sin tipo (magnitud, ángulo). Cuando un selector genérico opera sobre un número de tipo polar , elimina la etiqueta y pasa el contenido al código de Alyssa. Por el contrario, cuando Alyssa construye un número para uso general, lo etiqueta con un tipo para que pueda ser reconocido apropiadamente por los procedimientos de nivel superior. Esta disciplina de eliminar y agregar etiquetas a medida que los objetos de datos se pasan de un nivel a otro puede ser una estrategia organizacional importante, como veremos en la sección  [2.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.5) .

**[2.4.3 Programación dirigida por datos y aditividad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.4.3)**

La estrategia general de comprobar el tipo de un dato y llamar a un procedimiento apropiado se denomina*despacho en tipo* . Esta es una estrategia poderosa para obtener modularidad en el diseño de sistemas. Por otro lado, implementar el despacho como en la sección  [2.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.2) tiene dos debilidades significativas. Una debilidad es que los procedimientos de interfaz genéricos ( parte-real , parte-imag , magnitud y ángulo ) deben conocer todas las diferentes representaciones. Por ejemplo, supongamos que quisiéramos incorporar una nueva representación para números complejos en nuestro sistema de números complejos. Tendríamos que identificar esta nueva representación con un tipo y luego agregar una cláusula a cada uno de los procedimientos de interfaz genéricos para verificar el nuevo tipo y aplicar el selector apropiado para esa representación.

Otra debilidad de la técnica es que, aunque las representaciones individuales se pueden diseñar por separado, debemos garantizar que no haya dos procedimientos en todo el sistema que tengan el mismo nombre. Por eso, Ben y Alyssa tuvieron que cambiar los nombres de sus procedimientos originales de la sección  [2.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.1) .

El problema subyacente a ambas debilidades es que la técnica para implementar interfaces genéricas no es *aditiva* . La persona que implementa los procedimientos de selección genéricos debe modificar esos procedimientos cada vez que se instala una nueva representación, y las personas que interactúan con las representaciones individuales deben modificar su código para evitar conflictos de nombres. En cada uno de estos casos, los cambios que se deben realizar en el código son sencillos, pero deben realizarse de todos modos, y esto es una fuente de inconvenientes y errores. Esto no es un gran problema para el sistema de números complejos tal como está, pero supongamos que no hubiera dos sino cientos de representaciones diferentes para números complejos. Y supongamos que hubiera muchos selectores genéricos que mantener en la interfaz de datos abstractos. Supongamos, de hecho, que ningún programador conociera todos los procedimientos de interfaz o todas las representaciones. El problema es real y debe abordarse en programas como los sistemas de gestión de bases de datos a gran escala.

Lo que necesitamos es un medio para modularizar aún más el diseño del sistema. Esto lo proporciona la técnica de programación conocida como *programación dirigida por datos* . Para entender cómo funciona la programación dirigida por datos, comience con la observación de que siempre que tratamos con un conjunto de operaciones genéricas que son comunes a un conjunto de tipos diferentes, estamos, en efecto, tratando con una tabla bidimensional que contiene las posibles operaciones en un eje y los posibles tipos en el otro eje. Las entradas de la tabla son los procedimientos que implementan cada operación para cada tipo de argumento presentado. En el sistema de números complejos desarrollado en la sección anterior, la correspondencia entre el nombre de la operación, el tipo de datos y el procedimiento real se distribuía entre las diversas cláusulas condicionales en los procedimientos de interfaz genéricos. Pero la misma información podría haberse organizado en una tabla, como se muestra en la figura  [2.22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.22) .

La programación dirigida por datos es la técnica de diseñar programas para trabajar directamente con una tabla de este tipo. Anteriormente, implementamos el mecanismo que interconecta el código aritmético complejo con los dos paquetes de representación como un conjunto de procedimientos que realizan cada uno un envío explícito sobre el tipo. Aquí implementaremos la interfaz como un único procedimiento que busca la combinación del nombre de la operación y el tipo de argumento en la tabla para encontrar el procedimiento correcto a aplicar y luego lo aplica al contenido del argumento. Si hacemos esto, entonces para agregar un nuevo paquete de representación al sistema no necesitamos cambiar ningún procedimiento existente; solo necesitamos agregar nuevas entradas a la tabla.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.22:**   Tabla de operaciones para el sistema de números complejos. |

Para implementar este plan, supongamos que tenemos dos procedimientos, put y get , para manipular la tabla de operaciones y tipos:

* (put < *op* > < *type* > < *item* >)  
  instala el < *item* > en la tabla, indexada por el < *op* > y el < *type* > .
* (get < *op* > < *type* >)  
  busca la entrada < *op* > , < *type* > en la tabla y devuelve el elemento que se encuentra allí. Si no se encuentra ningún elemento, get devuelve falso.

Por ahora, podemos suponer que put y get están incluidas en nuestro lenguaje. En el capítulo 3 (sección  [3.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.3) , ejercicio  [3.24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.24) ) veremos cómo implementar estas y otras operaciones para manipular tablas.

A continuación se muestra cómo se puede utilizar la programación dirigida por datos en el sistema de números complejos. Ben, que desarrolló la representación rectangular, implementa su código tal como lo hizo originalmente. Define una colección de procedimientos, o un*paquete* y los conecta al resto del sistema agregando entradas a la tabla que le indican al sistema cómo operar con números rectangulares. Esto se logra llamando al siguiente procedimiento:

(define (install-rectangular-package)   
  *;; procedimientos internos*  
  (define (real-part z) (car z))   
  (define (imag-part z) (cdr z))   
  (define (make-from-real-imag x y) (cons x y))   
  (define (magnitude z)   
    (sqrt (+ (square (real-part z))   
             (square (imag-part z)))))   
  (define (angle z)   
    (atan (imag-part z) (real-part z)))   
  (define (make-from-mag-ang r a)    
    (cons (\* r (cos a)) (\* r (sin a))))   
  *;; interfaz con el resto del sistema*  
  (define (tag x) (attach-tag 'rectangular x))   
  (put 'real-part '(rectangular) real-part)   
  (put 'imag-part '(rectangular) imag-part)   
  (put 'magnitude 'magnitud (rectangular))   
  (poner 'ángulo '(rectangular) ángulo)   
  (poner 'hacer-desde-imagen-real 'rectangular    
       (lambda (x y) (etiqueta (hacer-desde-imagen-real x y))))   
  (poner 'hacer-desde-mag-ang 'rectangular    
       (lambda (r a) (etiqueta (hacer-desde-mag-ang r a))))   
  'hecho)

Observe que los procedimientos internos aquí son los mismos procedimientos de la sección  [2.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.1) que Ben escribió cuando trabajaba de forma aislada. No se necesitan cambios para interconectarlos con el resto del sistema. Además, dado que estas definiciones de procedimientos son internas al procedimiento de instalación, Ben no necesita preocuparse por conflictos de nombres con otros procedimientos fuera del paquete rectangular. Para interconectarlos con el resto del sistema, Ben instala su procedimiento de parte real bajo el nombre de operación parte-real y el tipo (rectangular) , y de manera similar para los otros selectores. [45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_270) La interfaz también define los constructores que utilizará el sistema externo. [46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_271) Estos son idénticos a los constructores definidos internamente por Ben, excepto que adjuntan la etiqueta.

El paquete polar de Alyssa es análogo:

(define (install-polar-package)   
  *;; procedimientos internos*  
  (define (magnitud z) (car z))   
  (define (ángulo z) (cdr z))   
  (define (make-from-mag-ang r a) (cons r a))   
  (define (parte-real z)   
    (\* (magnitud z) (cos (ángulo z))))   
  (define (parte-imag z)   
    (\* (magnitud z) (sin (ángulo z))))   
  (define (make-from-real-imag x y)    
    (cons (sqrt (+ (cuadrado x) (cuadrado y)))   
          (atan y x)))   
  *;; interfaz con el resto del sistema*  
  (define (etiqueta x) (etiqueta-adjuntar 'polar x))   
  (put 'parte-real '(polar) parte-real)   
  (put 'parte-imag '(polar) parte-imag)   
  (put 'magnitud '(polar) magnitud)   
  (put 'ángulo '(polar) ángulo)   
  (put 'make-from-real-imag 'polar   
       (lambda (x y) (etiqueta (make-from-real-imag x y))))   
  (poner 'make-from-mag-ang 'polar    
       (lambda (r a) (etiqueta (make-from-mag-ang r a))))   
  'hecho)

Aunque Ben y Alyssa todavía usan sus procedimientos originales definidos con los mismos nombres que el otro (por ejemplo, real-part ), estas definiciones ahora son internas a diferentes procedimientos (ver sección  [1.1.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.8) ), por lo que no hay ningún conflicto de nombres.

Los selectores aritméticos complejos acceden a la tabla mediante un procedimiento de ``operación'' general llamado apply-generic , que aplica una operación genérica a algunos argumentos. Apply-generic busca en la tabla bajo el nombre de la operación y los tipos de los argumentos y aplica el procedimiento resultante si hay alguno presente: [47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_272)

(define (apply-generic op . args)   
  (let ((type-tags (map type-tag args)))   
    (let ((proc (get op type-tags)))   
      (if proc   
          (apply proc (map contents args))   
          (error   
            "No hay método para estos tipos -- APPLY-GENERIC"   
            (list op type-tags)))))

Usando apply-generic , podemos definir nuestros selectores genéricos de la siguiente manera:

(define (parte-real z) (aplica-genérico 'parte-real z))  
(define (imag-parte z) (apply-generic 'imag-parte z))  
(define (magnitud z) (apply-generic 'magnitud z))  
(define (ángulo z) (apply-generic 'ángulo z))

Observe que estos no cambian en absoluto si se agrega una nueva representación al sistema.

También podemos extraer de la tabla los constructores que utilizarán los programas externos a los paquetes para construir números complejos a partir de partes reales e imaginarias y de magnitudes y ángulos. Como en la sección  [2.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.2) , construimos números rectangulares siempre que tengamos partes reales e imaginarias, y números polares siempre que tengamos magnitudes y ángulos:

(define (make-from-real-imag x y)   
  ((obtiene 'make-from-real-imag 'rectangular) x y))  
(definir (hacer-desde-mag-ang r a)   
  ((obtener 'hacer-desde-mag-ang 'polar) r a))

**Ejercicio 2.73.**   En la sección  [2.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.2) se describe un programa que realiza una diferenciación simbólica:

(define (deriva exp var)   
  (cond ((número? exp) 0)   
        ((variable? exp) (if (misma-variable? exp var) 1 0))   
        ((suma? exp)   
         (make-sum (deriva (addend exp) var)   
                   (deriva (augend exp) var)))   
        ((producto? exp)   
         (make-sum   
           (make-product (multiplicador exp)   
                         (deriva (multiplicando exp) var))   
           (make-product (deriva (multiplicador exp) var)   
                         (multiplicando exp))))   
        < *se pueden agregar más reglas aquí* >   
        (else (error "tipo de expresión desconocido -- DERIV" exp))))

Podemos considerar que este programa realiza un envío sobre el tipo de expresión que se va a diferenciar. En esta situación, la ``etiqueta de tipo'' del dato es el símbolo del operador algebraico (como + ) y la operación que se está realizando es deriv . Podemos transformar este programa en un estilo dirigido por datos reescribiendo el procedimiento de derivada básico como

(define (deriva exp var)   
   (cond ((número? exp) 0)   
         ((variable? exp) (si (misma-variable? exp var) 1 0))   
         (de lo contrario ((obtiene 'deriva (operador exp)) (operandos exp)   
                                            var))))   
(define (operador exp) (auto exp))   
(define (operandos exp) (cdr exp))

a. Explique lo que se hizo anteriormente. ¿Por qué no podemos asimilar los predicados number? y same-variable? en el envío dirigido por datos?

b. Escriba los procedimientos para derivadas de sumas y productos, y el código auxiliar necesario para instalarlos en la tabla utilizada por el programa anterior.

c. Elija cualquier regla de diferenciación adicional que le guste, como la de exponentes (ejercicio  [2.56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.56) ), e instálela en este sistema dirigido por datos.

d. En este manipulador algebraico simple, el tipo de una expresión es el operador algebraico que la une. Supongamos, sin embargo, que indexamos los procedimientos de la manera opuesta, de modo que la línea de envío en deriv se viera así

((obtener (operador exp) 'derivar) (operandos exp) var)

¿Qué cambios correspondientes se requieren en el sistema derivado?

**Ejercicio 2.74.**  Insatiable Enterprises, Inc., es un conglomerado empresarial altamente descentralizado que consta de un gran número de divisiones independientes ubicadas en todo el mundo. Las instalaciones informáticas de la empresa acaban de ser interconectadas mediante un ingenioso esquema de interconexión de redes que hace que toda la red parezca a cualquier usuario como una sola computadora. La presidenta de Insatiable, en su primer intento de explotar la capacidad de la red para extraer información administrativa de los archivos de las divisiones, se queda consternada al descubrir que, aunque todos los archivos de las divisiones se han implementado como estructuras de datos en Scheme, la estructura de datos particular utilizada varía de una división a otra. Se convoca apresuradamente una reunión de directores de división para buscar una estrategia para integrar los archivos que satisfaga las necesidades de la sede central y, al mismo tiempo, preserve la autonomía existente de las divisiones.

Demuestre cómo se puede implementar una estrategia de este tipo con programación dirigida por datos. Como ejemplo, supongamos que los registros de personal de cada división consisten en un solo archivo, que contiene un conjunto de registros codificados según los nombres de los empleados. La estructura del conjunto varía de una división a otra. Además, el registro de cada empleado es en sí mismo un conjunto (estructurado de manera diferente de una división a otra) que contiene información codificada según identificadores como la dirección y el salario . En particular:

a. Implemente para la sede central un procedimiento de obtención de registros que recupere el registro de un empleado específico de un expediente de personal específico. El procedimiento debe ser aplicable a cualquier expediente de división. Explique cómo deben estructurarse los expedientes de cada división. En particular, ¿qué tipo de información debe proporcionarse?

b. Implementar para la sede central un procedimiento de obtención de salarios que devuelva la información salarial del registro de un empleado determinado del archivo de personal de cualquier división. ¿Cómo debe estructurarse el registro para que esta operación funcione?

c. Implemente un procedimiento de búsqueda de registros de empleados para la sede central . Este procedimiento debería buscar en todos los archivos de las divisiones el registro de un empleado determinado y devolver el registro. Suponga que este procedimiento toma como argumentos el nombre de un empleado y una lista de todos los archivos de las divisiones.

d. Cuando Insatiable se hace cargo de una nueva empresa, ¿qué cambios se deben realizar para incorporar la nueva información del personal al sistema central?

**[Paso de mensajes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_275)**

La idea clave de la programación dirigida por datos es manejar operaciones genéricas en programas al tratar explícitamente con tablas de operaciones y tipos, como la tabla de la figura  [2.22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.22) . El estilo de programación que usamos en la sección  [2.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.2) organizó el despacho requerido por tipo haciendo que cada operación se encargara de su propio despacho. En efecto, esto descompone la tabla de operaciones y tipos en filas, donde cada procedimiento de operación genérica representa una fila de la tabla.

Una estrategia de implementación alternativa es descomponer la tabla en columnas y, en lugar de utilizar «operaciones inteligentes» que se despachan sobre tipos de datos, trabajar con «objetos de datos inteligentes» que se despachan sobre nombres de operaciones. Podemos hacer esto organizando las cosas de modo que un objeto de datos, como un número rectangular, se represente como un procedimiento que toma como entrada el nombre de la operación requerida y realiza la operación indicada. En una disciplina de este tipo, make-from-real-imag podría escribirse como

(define (make-from-real-imag x y)   
  (define (dispatch op)   
    (cond ((eq? op 'parte-real) x)   
          ((eq? op 'parte-imag) y)   
          ((eq? op 'magnitud)   
           (sqrt (+ (cuadrado x) (cuadrado y))))   
          ((eq? op 'ángulo) (atan y x))   
          (else   
           (error "Op desconocido -- MAKE-FROM-REAL-IMAG" op))))   
  dispatch)

El procedimiento apply-generic correspondiente , que aplica una operación genérica a un argumento, ahora simplemente introduce el nombre de la operación al objeto de datos y deja que el objeto haga el trabajo: [48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_276)

(define (aplicar operación genérica arg) (opción arg))

Tenga en cuenta que el valor devuelto por make-from-real-imag es un procedimiento (el procedimiento de envío interno ). Este es el procedimiento que se invoca cuando apply-generic solicita que se realice una operación.

Este estilo de programación se denomina *paso de mensajes* . El nombre proviene de la imagen de que un objeto de datos es una entidad que recibe el nombre de la operación solicitada como un "mensaje". Ya hemos visto un ejemplo de paso de mensajes en la sección  [2.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.3) , donde vimos cómo cons , car y cdr podrían definirse sin objetos de datos sino solo procedimientos. Aquí vemos que el paso de mensajes no es un truco matemático sino una técnica útil para organizar sistemas con operaciones genéricas. En el resto de este capítulo continuaremos utilizando la programación dirigida por datos, en lugar del paso de mensajes, para analizar operaciones aritméticas genéricas. En el capítulo 3 volveremos al paso de mensajes y veremos que puede ser una herramienta poderosa para estructurar programas de simulación.

**Ejercicio 2.75.**  Implemente el constructor make-from-mag-ang en el estilo de paso de mensajes. Este procedimiento debería ser análogo al procedimiento make-from-real-imag indicado anteriormente.

**Ejercicio 2.76.**  A medida que evoluciona un sistema grande con operaciones genéricas, pueden necesitarse nuevos tipos de objetos de datos o nuevas operaciones. Para cada una de las tres estrategias (operaciones genéricas con envío explícito, estilo dirigido por datos y estilo de paso de mensajes), describa los cambios que se deben realizar en un sistema para agregar nuevos tipos o nuevas operaciones. ¿Qué organización sería la más apropiada para un sistema en el que a menudo se deben agregar nuevos tipos? ¿Cuál sería la más apropiada para un sistema en el que a menudo se deben agregar nuevas operaciones?

[43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_268) En los sistemas computacionales reales, la forma rectangular es preferible a la forma polar la mayor parte del tiempo debido aErrores de redondeo en la conversión entre la forma rectangular y la polar. Por eso el ejemplo de los números complejos no es realista. No obstante, proporciona una ilustración clara del diseño de un sistema que utiliza operaciones genéricas y una buena introducción a los sistemas más importantes que se desarrollarán más adelante en este capítulo.

[44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_269) La función arcotangente a la que se hace referenciaAquí, calculado mediante el procedimiento atan de Scheme , se define de modo que tome dos argumentos *y*  y *x* y devuelva el ángulo cuya tangente es *y* / *x* . Los signos de los argumentos determinan el cuadrante del ángulo.

[Utilizamos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_270) la lista(rectangular)en lugar del símbolorectangularpara permitir la posibilidad de operaciones con múltiples argumentos, no todos del mismo tipo.

[El tipo bajo el](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_271) cual se instalan los constructores no necesita ser una lista porque un constructor siempre se usa para crear un objeto de un tipo particular.

[47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_272) Aplicar-genéricoutiliza elNotación de cola punteada descrita en el ejercicio  [2.20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.20) , porque diferentes operaciones genéricas pueden tomar diferentes cantidades de argumentos. En apply-generic , op tiene como valor el primer argumento de apply-generic y args tiene como valor una lista de los argumentos restantes.

Apply-generic también utiliza el procedimiento primitivoapply , que toma dos argumentos, un procedimiento y una lista. Apply aplica el procedimiento, utilizando los elementos de la lista como argumentos. Por ejemplo,

(aplicar + (lista 1 2 3 4))

devuelve 10.

[48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_276) Una limitación de esta organización es que sólo permite procedimientos genéricos de un argumento.

[**2.5 Sistemas con operaciones genéricas**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_2.5)

En la sección anterior, vimos cómo diseñar sistemas en los que los objetos de datos se pueden representar de más de una manera. La idea clave es vincular el código que especifica las operaciones de datos a las diversas representaciones por medio de procedimientos de interfaz genéricos. Ahora veremos cómo usar esta misma idea no sólo para definir operaciones que sean genéricas sobre diferentes representaciones sino también para definir operaciones que sean genéricas sobre diferentes tipos de argumentos. Ya hemos visto varios paquetes diferentes de operaciones aritméticas: la aritmética primitiva ( + , - , \* , / ) incorporada en nuestro lenguaje, la aritmética de números racionales ( add-rat , sub-rat , mul-rat , div-rat ) de la sección  [2.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.1) , y la aritmética de números complejos que implementamos en la sección  [2.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.3) . Ahora usaremos técnicas dirigidas por datos para construir un paquete de operaciones aritméticas que incorpore todos los paquetes aritméticos que ya hemos construido.

La figura  [2.23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.23) muestra la estructura del sistema que vamos a construir. Observe labarreras de abstracción. Desde la perspectiva de alguien que usa "números", hay un único procedimiento add que opera sobre cualquier número que se le proporcione. Add es parte de una interfaz genérica que permite que los paquetes separados de aritmética ordinaria, aritmética racional y aritmética compleja sean accedidos de manera uniforme por programas que usan números. Cualquier paquete aritmético individual (como el paquete complejo) puede ser accedido a través de procedimientos genéricos (como add-complex ) que combinan paquetes diseñados para diferentes representaciones (como rectangular y polar). Además, la estructura del sistema es aditiva, de modo que uno puede diseñar los paquetes aritméticos individuales por separado y combinarlos para producir un sistema aritmético genérico.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.23:**   Sistema aritmético genérico. |

**[2.5.1 Operaciones aritméticas genéricas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.5.1)**

La tarea de diseñar operaciones aritméticas genéricas es análoga a la de diseñar operaciones genéricas con números complejos. Nos gustaría, por ejemplo, tener un procedimiento de adición genérico add que actúe como la adición primitiva ordinaria + en números ordinarios, como add-rat en números racionales y como add-complex en números complejos. Podemos implementar add y las otras operaciones aritméticas genéricas siguiendo la misma estrategia que usamos en la sección  [2.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.3) para implementar los selectores genéricos para números complejos. Adjuntaremos una etiqueta de tipo a cada tipo de número y haremos que el procedimiento genérico se envíe a un paquete apropiado según el tipo de datos de sus argumentos.

Los procedimientos aritméticos genéricos se definen de la siguiente manera:

(define (agrega x y) (aplica-genérico 'agrega x y))  
(define (sub x y) (aplica-genérico 'sub x y))  
(define (mul x y) (aplica-genérico 'mul x y))  
(define (div x y) (aplica-genérico 'div x y))

Comenzamos instalando un paquete para manejar números *ordinarios* , es decir, los números primitivos de nuestro lenguaje. Los etiquetaremos con el símbolo scheme-number . Las operaciones aritméticas en este paquete son los procedimientos aritméticos primitivos (por lo que no hay necesidad de definir procedimientos adicionales para manejar los números no etiquetados). Como estas operaciones toman dos argumentos cada una, se instalan en la tabla codificada por la lista (scheme-number scheme-number) :

(define (install-scheme-number-package)   
  (define (tag x)   
    (attach-tag 'scheme-number x))       
  (put 'add '(scheme-number scheme-number)   
       (lambda (x y) (tag (+ x y))))   
  (put 'sub '(scheme-number scheme-number)   
       (lambda (x y) (tag (- x y))))   
  (put 'mul '(scheme-number scheme-number)   
       (lambda (x y) (tag (\* x y))))   
  (put 'div '(scheme-number scheme-number)   
       (lambda (x y) (tag (/ x y))))   
  (put 'make 'scheme-number   
       (lambda (x) (tag x)))   
  'done)

Los usuarios del paquete Scheme-number crearán (etiquetarán) números ordinarios mediante el procedimiento:

(define (make-scheme-number n)   
  ((obtiene 'make 'scheme-number) n))

Ahora que el marco del sistema aritmético genérico está en su lugar, podemos incluir fácilmente nuevos tipos de números. Aquí hay un paquete que realiza aritmética racional. Observe que, como beneficio de la aditividad, podemos usar sin modificación el código de números racionales de la sección  [2.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.1) como los procedimientos internos del paquete:

(define (install-rational-package)   
  *;; procedimientos internos*  
  (define (numer x) (car x))   
  (define (denom x) (cdr x))   
  (define (make-rat n d)   
    (let ((g (mcd n d)))   
      (cons (/ n g) (/ d g))))   
  (define (add-rat x y)   
    (make-rat (+ (\* (numer x) (denom y))   
                 (\* (numer y) (denom x)))   
              (\* (denom x) (denom y))))   
  (define (sub-rat x y)   
    (make-rat (- (\* (numer x) (denom y))   
                 (\* (numer y) (denom x)))   
              (\* (denom x) (denom y))))   
  (define (mul-rat x y)   
    (make-rat (\* (numer x) (numer y))   
              (\* (denom x) (denom y))))   
  (define (div-rat x y)   
    (make-rat (\* (numer x) (denom y))   
              (\* (denom x) (numer y))))   
  *;; interfaz con el resto del sistema*  
  (define (tag x) (attach-tag 'racional x))   
  (put 'add '(racional racional)   
       (lambda (x y) (tag (add-rat x y))))   
  (put 'sub '(racional racional)   
       (lambda (x y) (tag (sub-rat x y))))   
  (put 'mul '(racional racional)   
       (lambda (x y) (tag (mul-rat x y))))   
  (put 'div '(racional racional)   
       (lambda (x y) (tag (div-rat x y))))   
  
  (put 'make 'racional   
       (lambda (n d) (tag (make-rat n d))))   
  'hecho)  
(define (make-rational n d)   
  ((obtiene 'make 'rational) n d))

Podemos instalar un paquete similar para manejar números complejos, utilizando la etiqueta complex . Al crear el paquete, extraemos de la tabla las operaciones make-from-real-imag y make-from-mag-ang que fueron definidas por los paquetes rectangular y polar.La aditividad nos permite utilizar, como operaciones internas, los mismos procedimientos  add-complex , sub-complex , mul-complex y div-complex de la sección [2.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.1) .

(define (install-complex-package)   
  *;; procedimientos importados de paquetes rectangulares y polares*  
  (define (make-from-real-imag x y)   
    ((get 'make-from-real-imag 'rectangular) x y))   
  (define (make-from-mag-ang r a)   
    ((get 'make-from-mag-ang 'polar) r a))   
  *;; procedimientos internos*  
  (define (add-complex z1 z2)   
    (make-from-real-imag (+ (real-part z1) (real-part z2))   
                         (+ (imag-part z1) (imag-part z2))))   
  (define (sub-complex z1 z2)   
    (make-from-real-imag (- (real-part z1) (real-part z2))   
                         (- (imag-part z1) (imag-part z2))))   
  (define (mul-complex z1 z2)   
    (make-from-mag-ang (\* (magnitude z1) (magnitud z2))   
                       (+ (ángulo z1) (ángulo z2))))   
  (define (div-complex z1 z2)   
    (make-from-mag-ang (/ (magnitud z1) (magnitud z2))   
                       (- (ángulo z1) (ángulo z2))))   
  *;; interfaz con el resto del sistema*  
  (define (tag z) (attach-tag 'complex z))   
  (put 'add '(complex complex)   
       (lambda (z1 z2) (tag (add-complex z1 z2))))   
  (put 'sub '(complex complex)   
       (lambda (z1 z2) (tag (sub-complex z1 z2))))   
  (put 'mul '(complex complex)   
       (lambda (z1 z2) (tag (mul-complex z1 z2))))   
  (put 'div '(complex complex)   
       (lambda (z1 z2) (tag (div-complex z1 z2))))   
  (poner 'hacer-desde-imagen-real' complejo   
       (lambda (x y) (etiqueta (hacer-desde-imagen-real) x y))))   
  (poner 'hacer-desde-mag-ang' complejo   
       (lambda (r a) (etiqueta (hacer-desde-mag-ang r a))))   
  'hecho)

Los programas que no forman parte del paquete de números complejos pueden construir números complejos a partir de partes reales e imaginarias o de magnitudes y ángulos. Observe cómo los procedimientos subyacentes, definidos originalmente en los paquetes rectangular y polar, se exportan al paquete complejo y, desde allí, al mundo exterior.

(define (make-complex-from-real-imag x y)   
  ((obtiene 'make-from-real-imag 'complex) x y))  
(define (make-complex-from-mag-ang r a)   
  ((obtiene 'make-from-mag-ang 'complex) r a))

Lo que tenemos aquí es un sistema de etiquetas de dos niveles. Un número complejo típico, como 3 + 4 *i* en forma rectangular, se representaría como se muestra en la figura  [2.24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.24) . La etiqueta externa ( complex ) se utiliza para dirigir el número al paquete complejo. Una vez dentro del paquete complejo, la siguiente etiqueta ( rectangular ) se utiliza para dirigir el número al paquete rectangular. En un sistema grande y complicado puede haber muchos niveles, cada uno interconectado con el siguiente por medio de operaciones genéricas. A medida que un objeto de datos se pasa "hacia abajo", la etiqueta externa que se utiliza para dirigirlo al paquete apropiado se elimina (aplicando contents ) y el siguiente nivel de etiqueta (si lo hay) se vuelve visible para ser utilizado para un envío posterior.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.24:**   Representación de 3 + 4 *i* en forma rectangular. |

En los paquetes anteriores, utilizamos add-rat , add-complex y los demás procedimientos aritméticos exactamente como se escribieron originalmente. Sin embargo, una vez que estas definiciones sean internas a diferentes procedimientos de instalación, ya no necesitarán nombres que sean distintos entre sí: simplemente podríamos nombrarlas add , sub , mul y div en ambos paquetes.

**Ejercicio 2.77.**   Louis Reasoner intenta evaluar la expresión (magnitud z) donde z es el objeto que se muestra en la figura  [2.24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.24) . Para su sorpresa, en lugar de la respuesta 5, recibe un mensaje de error de apply-generic , que dice que no hay ningún método para la operación magnitud en los tipos (complex) . Le muestra esta interacción a Alyssa P. Hacker, quien dice: "El problema es que los selectores de números complejos nunca se definieron para números complejos , solo para números polares y rectangulares . Todo lo que tienes que hacer para que esto funcione es agregar lo siguiente al paquete complex :"

(poner 'parte-real '(complejo) parte-real)   
(poner 'parte-imag '(complejo) parte-imag)   
(poner 'magnitud '(complejo) magnitud)   
(poner 'ángulo '(complejo) ángulo)

Describa en detalle por qué funciona esto. Como ejemplo, realice un seguimiento de todos los procedimientos llamados al evaluar la expresión (magnitud z), donde z es el objeto que se muestra en la figura  [2.24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.24) . En particular, ¿cuántas veces se invoca apply-generic ? ¿A qué procedimiento se envía en cada caso?

**Ejercicio 2.78.**  Los procedimientos internos del paquete scheme-number no son, en esencia, más que llamadas a los procedimientos primitivos + , - , etc. No fue posible utilizar los primitivos del lenguaje directamente porque nuestro sistema de etiquetas de tipo requiere que cada objeto de datos tenga un tipo asociado a él. De hecho, sin embargo, todas las implementaciones de Lisp tienen un sistema de tipos, que utilizan internamente. Los predicados primitivos como symbol? y number? determinan si los objetos de datos tienen tipos particulares. Modifique las definiciones de type-tag , contents y attached-tag de la sección  [2.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.2) para que nuestro sistema genérico aproveche el sistema de tipos interno de Scheme. Es decir, el sistema debería funcionar como antes, excepto que los números ordinarios deberían representarse simplemente como números de Scheme en lugar de como pares cuyo car es el símbolo scheme-number .

**Ejercicio 2.79.**  Defina un predicado de igualdad genérico equ? que pruebe la igualdad de dos números e instálelo en el paquete aritmético genérico. Esta operación debería funcionar para números ordinarios, números racionales y números complejos.

**Ejercicio 2.80.**  Defina un predicado genérico =zero? que compruebe si su argumento es cero e instálelo en el paquete aritmético genérico. Esta operación debería funcionar para números ordinarios, racionales y complejos.

**[2.5.2 Combinación de datos de diferentes tipos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.5.2)**

Hemos visto cómo definir un sistema aritmético unificado que abarque números ordinarios, números complejos, números racionales y cualquier otro tipo de número que se nos ocurra, pero hemos ignorado un tema importante. Las operaciones que hemos definido hasta ahora tratan los diferentes tipos de datos como si fueran completamente independientes. Por lo tanto, hay paquetes separados para sumar, por ejemplo, dos números ordinarios o dos números complejos. Lo que aún no hemos considerado es el hecho de que tiene sentido definir operaciones que cruzan los límites de los tipos, como la suma de un número complejo a un número ordinario. Nos hemos esforzado mucho por introducir barreras entre las partes de nuestros programas para que puedan desarrollarse y entenderse por separado. Nos gustaría introducir las operaciones de tipos cruzados de alguna manera cuidadosamente controlada, para poder admitirlas sin violar seriamente los límites de nuestros módulos.

Una forma de manejar operaciones entre tipos es diseñar un procedimiento diferente para cada posible combinación de tipos para los que la operación sea válida. Por ejemplo, podríamos extender el paquete complex-number para que proporcione un procedimiento para sumar números complejos a números ordinarios e instalarlo en la tabla usando la etiqueta (complex scheme-number) : [49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_283)

*;; para ser incluido en el paquete complejo*  
(define (agregar-complejo-a-esquema z x)   
  (crea-a-partir-de-imag-real (+ (parte-real z) x)   
                       (parte-imag z)))   
(pone 'agregar '(número-de-esquema-complejo)   
     (lambda (z x) (etiqueta (agregar-complejo-a-esquema z x))))

Esta técnica funciona, pero es engorrosa. Con un sistema de este tipo, el coste de introducir un nuevo tipo no es sólo la construcción del paquete de procedimientos para ese tipo, sino también la construcción e instalación de los procedimientos que implementan las operaciones entre tipos. Esto puede suponer fácilmente mucho más código del que se necesita para definir las operaciones sobre el propio tipo. El método también socava nuestra capacidad de combinar paquetes separados de forma aditiva, o al menos de limitar el grado en que los implementadores de los paquetes individuales deben tener en cuenta otros paquetes. Por ejemplo, en el ejemplo anterior, parece razonable que el manejo de operaciones mixtas sobre números complejos y números ordinarios debería ser responsabilidad del paquete de números complejos. Sin embargo, la combinación de números racionales y números complejos podría ser realizada por el paquete complejo, por el paquete racional o por algún tercer paquete que utilice operaciones extraídas de estos dos paquetes. Formular políticas coherentes sobre la división de responsabilidades entre paquetes puede ser una tarea abrumadora al diseñar sistemas con muchos paquetes y muchas operaciones entre tipos.

**[Coerción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_284)**

En la situación general de operaciones completamente no relacionadas que actúan sobre tipos completamente no relacionados, implementar operaciones explícitas entre tipos, por engorroso que pueda ser, es lo mejor que se puede esperar. Afortunadamente, normalmente podemos hacerlo mejor aprovechando la estructura adicional que puede estar latente en nuestro sistema de tipos. A menudo, los diferentes tipos de datos no son completamente independientes, y puede haber formas por las cuales los objetos de un tipo pueden verse como si fueran de otro tipo. Este proceso se llama *coerción* . Por ejemplo, si se nos pide que combinemos aritméticamente un número ordinario con un número complejo, podemos ver el número ordinario como un número complejo cuya parte imaginaria es cero. Esto transforma el problema en el de combinar dos números complejos, que puede ser manejado de la manera ordinaria por el paquete complex-arithmetic.

En general, podemos implementar esta idea diseñando procedimientos de coerción que transformen un objeto de un tipo en un objeto equivalente de otro tipo. A continuación se muestra un procedimiento de coerción típico, que transforma un número ordinario dado en un número complejo con esa parte real y la parte imaginaria cero:

(define (número-de-esquema->complejo n)   
  (hace-complejo-a-partir-de-imagen-real (contenido n) 0))

Instalamos estos procedimientos de coerción en una tabla de coerción especial, indexada bajo los nombres de los dos tipos:

(put-coercion 'esquema-numero 'esquema-complejo numero->complejo)

(Suponemos que existen procedimientos de coerción put y get disponibles para manipular esta tabla). Generalmente, algunas de las ranuras de la tabla estarán vacías, porque no es posible convertir un objeto de datos arbitrario de cada tipo en todos los demás tipos. Por ejemplo, no hay forma de convertir un número complejo arbitrario en un número ordinario, por lo que no se incluirá ningún procedimiento complejo->número de esquema general en la tabla.

Una vez que se ha configurado la tabla de coerción, podemos manejar la coerción de una manera uniforme modificando el procedimiento  apply-generic de la sección [2.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.3) . Cuando se nos pide que apliquemos una operación, primero verificamos si la operación está definida para los tipos de argumentos, tal como antes. Si es así, enviamos al procedimiento que se encuentra en la tabla de operaciones y tipos. De lo contrario, intentamos la coerción. Para simplificar, consideramos solo el caso en el que hay dos argumentos. [50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_285) Verificamos la tabla de coerción para ver si los objetos del primer tipo pueden ser coaccionados al segundo tipo. Si es así, coaccionamos el primer argumento e intentamos la operación nuevamente. Si los objetos del primer tipo no pueden, en general, ser coaccionados al segundo tipo, intentamos la coerción al revés para ver si hay una manera de coaccionar el segundo argumento al tipo del primer argumento. Finalmente, si no hay una manera conocida de coaccionar ninguno de los tipos al otro tipo, nos damos por vencidos. Este es el procedimiento:

(define (apply-generic op . args)   
  (let ((type-tags (map type-tag args)))   
    (let ((proc (get op type-tags)))   
      (if proc   
          (apply proc (map contents args))   
          (if (= (length args) 2)   
              (let ((type1 (car type-tags))   
                    (type2 (cadr type-tags))   
                    (a1 (car args))   
                    (a2 (cadr args)))   
                (let ((t1->t2 (get-coercion type1 type2))   
                      (t2->t1 (get-coercion type2 type1)))   
                  (cond (t1->t2   
                         (apply-generic op (t1->t2 a1) a2))   
                        (t2->t1   
                         (apply-generic op a1 (t2->t1 a2)))   
                        (else   
                         (error "No hay método para estos tipos"   
                                (list op etiquetas-de-tipo))))))   
              (error "No hay método para estos tipos"   
                     (lista op etiquetas-de-tipo)))))))

Este esquema de coerción tiene muchas ventajas sobre el método de definir operaciones explícitas entre tipos, como se ha señalado anteriormente. Aunque todavía necesitamos escribir procedimientos de coerción para relacionar los tipos (posiblemente *n* 2 procedimientos para un sistema con *n* tipos), necesitamos escribir sólo un procedimiento para cada par de tipos en lugar de un procedimiento diferente para cada conjunto de tipos y cada operación genérica. [51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_286) Lo que estamos contando aquí es el hecho de que la transformación adecuada entre tipos depende sólo de los tipos mismos, no de la operación que se va a aplicar.

Por otra parte, puede haber aplicaciones para las que nuestro esquema de coerción no sea lo suficientemente general. Incluso cuando ninguno de los objetos que se van a combinar se pueda convertir al tipo del otro, puede ser posible realizar la operación convirtiendo ambos objetos a un tercer tipo. Para poder lidiar con tal complejidad y al mismo tiempo preservar la modularidad en nuestros programas, normalmente es necesario construir sistemas que aprovechen una estructura aún mayor en las relaciones entre los tipos, como veremos a continuación.

**[Jerarquías de tipos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_287)**

El esquema de coerción presentado anteriormente se basaba en la existencia de relaciones naturales entre pares de tipos. A menudo hay una estructura más "global" en cómo se relacionan entre sí los diferentes tipos. Por ejemplo, supongamos que estamos construyendo un sistema aritmético genérico para manejar números enteros, números racionales, números reales y números complejos. En un sistema de este tipo, es bastante natural considerar un número entero como un tipo especial de número racional, que a su vez es un tipo especial de número real, que a su vez es un tipo especial de número complejo. Lo que tenemos en realidad es una denominada *jerarquía de tipos* , en la que, por ejemplo, los números enteros son un*subtipo* de números racionales (es decir, cualquier operación que se pueda aplicar a un número racional se puede aplicar automáticamente a un entero). Por el contrario, decimos que los números racionales forman un*supertipo* de números enteros. La jerarquía particular que tenemos aquí es de un tipo muy simple, en la que cada tipo tiene como máximo un supertipo y como máximo un subtipo. Una estructura de este tipo, llamada *torre* , se ilustra en la figura  [2.25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.25) .

|  |
| --- |
|  |
|  |
| **Figura 2.25:**   Una torre de tipos. |

Si tenemos una estructura de torre, podemos simplificar enormemente el problema de agregar un nuevo tipo a la jerarquía, ya que solo necesitamos especificar cómo se incrusta el nuevo tipo en el supertipo que está por encima de él y cómo es el supertipo del tipo que está por debajo de él. Por ejemplo, si queremos agregar un entero a un número complejo, no necesitamos definir explícitamente un procedimiento de coerción especial entero->complejo . En cambio, definimos cómo se puede transformar un entero en un número racional, cómo se transforma un número racional en un número real y cómo se transforma un número real en un número complejo. Luego permitimos que el sistema transforme el entero en un número complejo a través de estos pasos y luego sumamos los dos números complejos.

Podemos rediseñar nuestro procedimiento de aplicación genérica de la siguiente manera: para cada tipo, necesitamos proporcionar un procedimiento de elevación , que "eleva" los objetos de ese tipo un nivel en la torre. Luego, cuando se requiere que el sistema opere sobre objetos de diferentes tipos, puede elevar sucesivamente los tipos inferiores hasta que todos los objetos estén en el mismo nivel en la torre. (Los ejercicios  [2.83](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.83) y   [2.84](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.84) tratan los detalles de la implementación de dicha estrategia).

Otra ventaja de una torre es que podemos implementar fácilmente la noción de que cada tipo "hereda" todas las operaciones definidas en un supertipo. Por ejemplo, si no proporcionamos un procedimiento especial para encontrar la parte real de un entero, deberíamos esperar de todas formas que la parte real se defina para los enteros en virtud del hecho de que los enteros son un subtipo de los números complejos. En una torre, podemos hacer que esto suceda de manera uniforme modificando apply-generic . Si la operación requerida no está definida directamente para el tipo del objeto dado, elevamos el objeto a su supertipo y probamos de nuevo. De este modo, trepamos por la torre, transformando nuestro argumento a medida que avanzamos, hasta que encontramos un nivel en el que se puede realizar la operación deseada o llegamos a la cima (en cuyo caso nos damos por vencidos).

Otra ventaja de una torre sobre una jerarquía más general es que nos da una manera sencilla de "reducir" un objeto de datos a la representación más simple. Por ejemplo, si sumamos 2 + 3 *i* a 4 - 3 *i* , sería bueno obtener la respuesta como el entero 6 en lugar del número complejo 6 + 0 *i* . El ejercicio  [2.85](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.85) analiza una manera de implementar una operación de reducción de este tipo. (El truco es que necesitamos una manera general de distinguir aquellos objetos que se pueden reducir, como 6 + 0 *i* , de aquellos que no, como 6 + 2 *i* .)

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 2.26:**   Relaciones entre tipos de figuras geométricas. |

**[Insuficiencias de las jerarquías](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_288)**

Si los tipos de datos de nuestro sistema se pueden organizar de forma natural en una torre, esto simplifica enormemente los problemas de tratar con operaciones genéricas sobre diferentes tipos, como hemos visto. Desafortunadamente, este no suele ser el caso. La Figura  [2.26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.26) ilustra una disposición más compleja de tipos mixtos, que muestra relaciones entre diferentes tipos de figuras geométricas. Vemos que, en general,Un tipo puede tener más de un subtipo. Por ejemplo, los triángulos y los cuadriláteros son ambos subtipos de polígonos. Además, un tipo puede tener más de un supertipo. Por ejemplo, un triángulo rectángulo isósceles puede considerarse como un triángulo isósceles o como un triángulo rectángulo. Este problema de los supertipos múltiples es particularmente espinoso, ya que significa que no hay una forma única de "elevar" un tipo en la jerarquía. Encontrar el supertipo "correcto" en el que aplicar una operación a un objeto puede implicar una búsqueda considerable a través de toda la red de tipos por parte de un procedimiento como apply-generic . Dado que generalmente hay múltiples subtipos para un tipo, existe un problema similar al forzar un valor "hacia abajo" en la jerarquía de tipos. Tratar con un gran número de tipos interrelacionados mientras se conserva la modularidad en el diseño de sistemas grandes es muy difícil y es un área de mucha investigación actual. [52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_289)

**Ejercicio 2.81.**  Louis Reasoner ha notado que apply-generic puede intentar convertir los argumentos en tipos de los demás incluso si ya tienen el mismo tipo. Por lo tanto, razona, necesitamos poner procedimientos en la tabla de conversión para "convertir" los argumentos de cada tipo en su propio tipo. Por ejemplo, además de la conversión scheme-number->complex que se muestra arriba, haría lo siguiente:

(define (número-de-esquema->número-de-esquema n) n)  
(define (complejo->complejo z) z)   
(put-coercion 'esquema-numero 'esquema-numero   
              esquema->esquema-numero)   
(put-coercion 'complejo 'complejo complejo->complejo)

a. Con los procedimientos de coerción de Louis instalados, ¿qué sucede si se llama a apply-generic con dos argumentos de tipo scheme-number o dos argumentos de tipo complex para una operación que no se encuentra en la tabla para esos tipos? Por ejemplo, supongamos que hemos definido una operación de exponenciación genérica:

(define (exp x y) (aplica-genérico 'exp x y))

y he puesto un procedimiento para exponenciación en el paquete Scheme-number pero no en ningún otro paquete:

*;; lo siguiente se agregó al paquete Scheme-number*  
(put 'exp '(scheme-number scheme-number)   
     (lambda (x y) (tag (expt x y))))  *; usando el primitivo  expt*

¿Qué sucede si llamamos a exp con dos números complejos como argumentos?

b. ¿Tiene razón Louis al afirmar que se debía hacer algo con respecto a la coerción con argumentos del mismo tipo, o el método apply-generic funciona correctamente tal como está?

c. Modifique apply-generic para que no intente la coerción si los dos argumentos tienen el mismo tipo.

**Ejercicio 2.82.**  Muestre cómo generalizar apply-generic para manejar la coerción en el caso general de múltiples argumentos. Una estrategia es intentar coaccionar todos los argumentos al tipo del primer argumento, luego al tipo del segundo argumento, y así sucesivamente. Dé un ejemplo de una situación en la que esta estrategia (y también la versión de dos argumentos dada anteriormente) no sea lo suficientemente general. (Sugerencia: considere el caso en el que hay algunas operaciones de tipo mixto adecuadas presentes en la tabla que no se intentarán).

**Ejercicio 2.83.**  Supongamos que está diseñando un sistema aritmético genérico para trabajar con la torre de tipos que se muestra en la figura  [2.25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_2.25) : entero, racional, real, complejo. Para cada tipo (excepto complejo), diseñe un procedimiento que eleve los objetos de ese tipo un nivel en la torre. Muestre cómo instalar una operación de elevación genérica que funcione para cada tipo (excepto complejo).

**Ejercicio 2.84.**  Utilizando la operación  raise del ejercicio [2.83](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.83) , modifique el procedimiento apply-generic de modo que fuerce a sus argumentos a tener el mismo tipo mediante el método de elevación sucesiva, como se explica en esta sección. Deberá idear una forma de probar cuál de los dos tipos está más arriba en la torre. Hágalo de una manera que sea "compatible" con el resto del sistema y que no genere problemas al agregar nuevos niveles a la torre.

**Ejercicio 2.85.**  En esta sección se mencionó un método para ``simplificar'' un objeto de datos bajándolo en la torre de tipos lo más lejos posible. Diseñe un procedimiento drop que logre esto para la torre descrita en el ejercicio  [2.83](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.83) . La clave es decidir, de alguna manera general, si un objeto puede ser bajado. Por ejemplo, el número complejo 1.5 + 0 *i* puede ser bajado hasta real , el número complejo 1 + 0 *i* puede ser bajado hasta entero , y el número complejo 2 + 3 *i* no puede ser bajado en absoluto. Aquí hay un plan para determinar si un objeto puede ser bajado: Comience por definir una operación genérica project que ``empuja'' un objeto hacia abajo en la torre. Por ejemplo, proyectar un número complejo implicaría descartar la parte imaginaria. Luego, un número puede ser descartado si, cuando lo proyectamos y elevamos el resultado de nuevo al tipo con el que comenzamos, terminamos con algo igual a lo que comenzamos. Muestre cómo implementar esta idea en detalle, escribiendo un procedimiento drop que descarte un objeto lo más lejos posible. Necesitará diseñar las distintas operaciones de proyección [53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_295) e instalar el proyecto como una operación genérica en el sistema. También necesitará hacer uso de un predicado de igualdad genérico, como se describe en el ejercicio  [2.79](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.79) . Finalmente, use drop para reescribir apply-generic del ejercicio  [2.84](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.84) de modo que ``simplifique'' sus respuestas.

**Ejercicio 2.86.**   Supongamos que queremos manejar números complejos cuyas partes reales, partes imaginarias, magnitudes y ángulos pueden ser números ordinarios, números racionales u otros números que queramos añadir al sistema. Describa e implemente los cambios necesarios en el sistema para adaptarlo a esto. Deberá definir operaciones como seno y coseno que sean genéricas para números ordinarios y números racionales.

**[2.5.3 Ejemplo: Álgebra simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_2.5.3)**

La manipulación de expresiones algebraicas simbólicas es un proceso complejo que ilustra muchos de los problemas más difíciles que surgen en el diseño de sistemas a gran escala. Una expresión algebraica, enEn general, se puede considerar como una estructura jerárquica, un árbol de operadores aplicados a operandos. Podemos construir expresiones algebraicas comenzando con un conjunto de objetos primitivos, como constantes y variables, y combinándolos por medio de operadores algebraicos, como la suma y la multiplicación. Como en otros lenguajes, formamos abstracciones que nos permiten referirnos a objetos compuestos en términos simples. Abstracciones típicas en álgebra simbólica son ideas como combinación lineal, polinomio, función racional o función trigonométrica. Podemos considerarlas como "tipos" compuestos, que a menudo son útiles para dirigir el procesamiento de expresiones. Por ejemplo, podríamos describir la expresión



como un polinomio en *x* con coeficientes que son funciones trigonométricas de polinomios en *y* cuyos coeficientes son números enteros.

No intentaremos desarrollar aquí un sistema completo de manipulación algebraica. Estos sistemas son programas sumamente complejos que incorporan un profundo conocimiento algebraico y algoritmos elegantes. Lo que haremos será observar una parte simple pero importante de la manipulación algebraica: la aritmética de polinomios. Ilustraremos los tipos de decisiones que enfrenta el diseñador de un sistema de este tipo y cómo aplicar las ideas de datos abstractos y operaciones genéricas para ayudar a organizar este esfuerzo.

**[Aritmética sobre polinomios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_297)**

Nuestra primera tarea al diseñar un sistema para realizar operaciones aritméticas con polinomios es decidir qué es exactamente un polinomio. Los polinomios normalmente se definen en relación con ciertas variables (la*indeterminados* del polinomio). Para simplificar, nos limitaremos a polinomios que tengan solo un indeterminado (*polinomios univariados* ). [54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_298) Definiremos un polinomio como una suma de términos, cada uno de los cuales es un coeficiente, una potencia de lo indeterminado o un producto de un coeficiente y una potencia de lo indeterminado. Un coeficiente se define como una expresión algebraica que no depende de lo indeterminado del polinomio. Por ejemplo,



es un polinomio simple en *x* , y



es un polinomio en *x* cuyos coeficientes son polinomios en *y* .

Ya estamos eludiendo algunas cuestiones espinosas. ¿El primero de estos polinomios es el mismo que el polinomio 5 *y* 2 + 3 *y* + 7, o no? Una respuesta razonable podría ser "sí, si estamos considerando un polinomio puramente como una función matemática, pero no, si estamos considerando un polinomio como una forma sintáctica". El segundo polinomio es algebraicamente equivalente a un polinomio en *y* cuyos coeficientes son polinomios en *x* . ¿Nuestro sistema debería reconocer esto, o no? Además, hay otras formas de representar un polinomio - por ejemplo, como un producto de factores, o (para un polinomio univariado) como el conjunto de raíces, o como una lista de los valores del polinomio en un conjunto específico de puntos. [55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_299) Podemos pulir estas cuestiones decidiendo que en nuestro sistema de manipulación algebraica un "polinomio" será una forma sintáctica particular, no su significado matemático subyacente.

Ahora debemos considerar cómo proceder para realizar operaciones aritméticas con polinomios. En este sistema simple, consideraremos únicamente la suma y la multiplicación. Además, insistiremos en que los dos polinomios que se van a combinar deben tener la misma indeterminación.

Abordaremos el diseño de nuestro sistema siguiendo la conocida disciplina de abstracción de datos. Representaremos polinomios utilizando una estructura de datos llamada*poli* , que consta de una variable y unacolección de términos. Suponemos que tenemos selectores variable y term-list que extraen esas partes de un poly y un constructor make-poly que ensambla un poly a partir de una variable dada y una lista de términos. Una variable será solo un símbolo, por lo que podemos usar el¿misma variable? procedimiento de la sección  [2.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.2) para comparar variables.Los siguientes procedimientos definen la suma y multiplicación de polinomios:

(define (add-poly p1 p2)   
  (if (misma-variable? (variable p1) (variable p2))   
      (make-poly (variable p1)   
                 (add-terms (term-list p1)   
                            (term-list p2)))   
      (error "Los polígonos no están en la misma variable -- ADD-POLY"   
             (list p1 p2))))  
(define (mul-poly p1 p2)   
  (if (misma-variable? (variable p1) (variable p2))   
      (make-poly (variable p1)   
                 (mul-terms (lista-términos p1)   
                            (lista-términos p2)))   
      (error "Los polinomios no están en la misma variable -- MUL-POLY"   
             (lista p1 p2))))

Para incorporar polinomios a nuestro sistema aritmético genérico, necesitamos proporcionarles etiquetas de tipo. Usaremos la etiqueta polynomial e instalaremos las operaciones apropiadas en los polinomios etiquetados en la tabla de operaciones. Incorporaremos todo nuestro código en un procedimiento de instalación para el paquete polynomial, similar a los de la sección  [2.5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.5.1) :

(define (install-polynomial-package)   
  *;; procedimientos internos*   
  *;; representación de poly*  
  (define (make-poly variable term-list)   
    (cons variable term-list))   
  (define (variable p) (car p))   
  (define (term-list p) (cdr p))   
  < *procedimientos ¿  misma-variable?  y  variable?  de la sección* [*2.3.2*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.2) >   
  *;; representación de términos y listas de términos*  
  < *procedimientos  adjoin-term  ... coeff  del texto de abajo* >   
  
  *;; continúa en la siguiente página*  
  
  (define (add-poly p1 p2)  ... )   
  < *procedimientos usados ​​por  add-poly* >   
  (define (mul-poly p1 p2)  ... )   
  < *procedimientos usados ​​por  mul-poly* >   
  *;; interfaz con el resto del sistema*  
  (define (tag p) (attach-tag 'polynomial p))   
  (put 'add '(polynomial polynomial)    
       (lambda (p1 p2) (tag (add-poly p1 p2))))   
  (put 'mul '(polinomio polinomio)    
       (lambda (p1 p2) (tag (mul-poly p1 p2))))   
  (put 'make 'polinomio   
       (lambda (var términos) (tag (make-poly var términos))))   
  'hecho)

La suma de polinomios se realiza término por término. Los términos del mismo orden (es decir, con la misma potencia del indeterminado) deben combinarse. Esto se hace formando un nuevo término del mismo orden cuyo coeficiente es la suma de los coeficientes de los sumandos. Los términos de un sumando para los que no hay términos del mismo orden en el otro sumando simplemente se acumulan en el polinomio suma que se está construyendo.

Para manipular listas de términos, asumiremos que tenemos un constructorthe-empty-termlist que devuelve una lista de términos vacía y un constructortérmino adjunto que adjunta un nuevo término a una lista de términos. También supondremos que tenemos un predicado¿lista de términos vacía? que indica si una lista de términos dada está vacía, un selectorprimer término que extrae el término de orden más alto de una lista de términos y un selectorrest-terms que devuelve todos los términos excepto el de orden más alto. Para manipular los términos, supondremos que tenemos un constructortérmino de creación que construye un término con el orden y coeficiente dados, y selectoresorden ycoeff que devuelven, respectivamente, el orden y el coeficiente del término. Estas operaciones nos permiten considerar tanto los términos como las listas de términos como abstracciones de datos, de cuyas representaciones concretas podemos ocuparnos por separado.

Aquí está el procedimiento que construye la lista de términos para la suma de dos polinomios: [56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_300)

(define (agregar-términos L1 L2)   
  (cond ((lista-de-términos-vacía? L1) L2)   
        ((lista-de-términos-vacía? L2) L1)   
        (else   
         (let ((t1 (primer-término L1)) (t2 (primer-término L2)))   
           (cond ((> (orden t1) (orden t2))   
                  (término-adjunto   
                   t1 (agregar-términos (resto-términos L1) L2)))   
                 ((< (orden t1) (orden t2))   
                  (término-adjunto   
                   t2 (agregar-términos L1 (resto-términos L2))))   
                 (else   
                  (término-adjunto   
                   (hacer-término (orden t1)   
                              (agregar (coef t1) (coef t2)))   
                   (términos-add (resto-términos L1)   
                              (resto-términos L2)))))))))

El punto más importante a tener en cuenta aquí es que utilizamos el procedimiento de adición genérico.add para sumar los coeficientes de los términos que se combinan. Esto tiene consecuencias importantes, como veremos a continuación.

Para multiplicar dos listas de términos, multiplicamos cada término de la primera lista por todos los términos de la otra lista, utilizando repetidamente mul-term-by-all-terms , que multiplica un término dado por todos los términos de una lista de términos dada. Las listas de términos resultantes (una por cada término de la primera lista) se acumulan en una suma. Al multiplicar dos términos se forma un término cuyo orden es la suma de los órdenes de los factores y cuyo coeficiente es el producto de los coeficientes de los factores:

(define (términos múltiples L1 L2)   
  (si (lista-de-términos-vacía? L1)   
      (la-lista-de-términos-vacía)   
      (términos-adicionales (término-múltiple-por-todos-los-términos (primer-término L1) L2)   
                 (términos-múltiple (resto-términos L1) L2))))   
(define (término-múltiple-por-todos-los-términos t1 L)   
  (si (lista-de-términos-vacía? L)   
      (la-lista-de-términos-vacía)   
      (let ((t2 (primer-término L)))   
        (término-adjunto   
         (término-creación (+ (orden t1) (orden t2))   
                    (mul (coef t1) (coef t2)))   
         (término-múltiple-por-todos-los-términos t1 (resto-términos L))))))

Esto es realmente todo lo que hay que saber sobre la suma y multiplicación de polinomios.Tenga en cuenta que, dado que operamos sobre términos utilizando los procedimientos genéricos add y mul , nuestro paquete polinomial puede manejar automáticamente cualquier tipo de coeficiente que conozca el paquete aritmético genérico. Si incluimos unmecanismo de coerción como uno de los discutidos en la sección  [2.5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.5.2) , entonces también podemos manejar automáticamente operaciones en polinomios de diferentes tipos de coeficientes, como



Debido a que instalamos los procedimientos de suma y multiplicación de polinomios add-poly y mul-poly en el sistema aritmético genérico como las operaciones add y mul para el tipo polynomial , nuestro sistema también puede manejar automáticamente operaciones polinomiales como



La razón es que cuando el sistema intenta combinar coeficientes, los enviará a través de add y mul . Dado que los coeficientes son en sí mismos polinomios (en *y* ), estos se combinarán utilizando add-poly y mul-poly . El resultado es una especie de"recursión dirigida por datos" en la que, por ejemplo, una llamada a mul-poly dará como resultado llamadas recursivas a mul-poly para multiplicar los coeficientes. Si los coeficientes de los coeficientes fueran en sí mismos polinomios (como se podría usar para representar polinomios en tres variables), la dirección de los datos garantizaría que el sistema continuara a través de otro nivel de llamadas recursivas, y así sucesivamente a través de tantos niveles como dicte la estructura de los datos. [57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_301)

[**Representación de listas de términos**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_302)

Finalmente, debemos afrontar la tarea de implementar una buena representación para listas de términos. Una lista de términos es, en efecto, un conjunto de coeficientes ordenados por el orden del término. Por lo tanto, cualquiera de los métodos para representar conjuntos, como se discutió en la sección  [2.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.3) , se puede aplicar a esta tarea. Por otro lado, nuestros procedimientos add-terms y mul-terms siempre acceden a las listas de términos secuencialmente desde el orden más alto al más bajo. Por lo tanto, utilizaremos algún tipo de representación de lista ordenada.

¿Cómo debemos estructurar la lista que representa una lista de términos? Una consideración es la "densidad" de los polinomios que pretendemos manipular. Se dice que un polinomio es*densa* si tiene coeficientes distintos de cero en términos de la mayoría de los órdenes. Si tiene muchos términos cero se dice que es*escaso* . Por ejemplo,



es un polinomio denso, mientras que



Es escaso.

Las listas de términos de polinomios densos se representan de manera más eficiente como listas de coeficientes. Por ejemplo, *A* arriba se representaría bien como (1 2 0 3 -2 -5) . El orden de un término en esta representación es la longitud de la sublista que comienza con el coeficiente de ese término, decrementado en 1. [58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_303) Esta sería una representación terrible para un polinomio disperso como *B* : Habría una lista gigante de ceros puntuada por unos pocos términos solitarios distintos de cero. Una representación más razonable de la lista de términos de un polinomio disperso es como una lista de los términos distintos de cero, donde cada término es una lista que contiene el orden del término y el coeficiente para ese orden. En un esquema de este tipo, el polinomio *B* se representa de manera eficiente como ((100 1) (2 2) (0 1)) . Como la mayoría de las manipulaciones polinómicas se realizan en polinomios dispersos, utilizaremos este método. Supondremos que las listas de términos se representan como listas de términos, ordenados desde el término de mayor orden hasta el de menor orden. Una vez que hayamos tomado esta decisión, implementar los selectores y constructores para términos y listas de términos es sencillo: [59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_304)

(define (término adjunto término lista de términos)   
  (si (=cero? (coeficiente término))   
      lista de términos   
      (cons término lista de términos)))  
(define (la-lista-de-términos-vacía) '())  
(define (primer término lista de términos) (auto lista de términos))  
(define (rest-terms lista-de-términos) (cdr lista-de-términos))  
(define (¿lista-de-términos-vacía? lista-de-términos) (¿lista-de-términos-nula?))  
(define (coeficiente de orden del término de creación) (coeficiente de orden de lista))  
(definir (término de orden) (término de automóvil))  
(definir (término coef) (término cadr))

donde =cero? es como se define en el ejercicio  [2.80](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.80) . (Véase también el ejercicio  [2.87](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.87) a continuación).

Los usuarios del paquete polinomial crearán polinomios (etiquetados) mediante el procedimiento:

(define (make-polinomio var términos)   
  ((obtiene 'make 'polinomio) var términos))

**Ejercicio 2.87.**  Instalar =zero? para polinomios en el paquete aritmético genérico. Esto permitirá que adjoin-term funcione para polinomios con coeficientes que sean polinomios en sí mismos.

**Ejercicio 2.88.**  Amplíe el sistema polinomial para incluir la resta de polinomios. (Sugerencia: puede resultarle útil definir una operación de negación genérica).

**Ejercicio 2.89.**   Defina procedimientos que implementen la representación de lista de términos descrita anteriormente como apropiada para polinomios densos.

**Ejercicio 2.90.**   Supongamos que queremos tener un sistema polinómico que sea eficiente tanto para polinomios dispersos como densos. Una forma de hacerlo es permitir ambos tipos de representaciones de listas de términos en nuestro sistema. La situación es análoga al ejemplo de números complejos de la sección  [2.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4) , donde permitimos representaciones tanto rectangulares como polares. Para ello, debemos distinguir diferentes tipos de listas de términos y hacer que las operaciones sobre las listas de términos sean genéricas. Rediseñe el sistema polinómico para implementar esta generalización. Este es un esfuerzo importante, no un cambio local.

**Ejercicio 2.91.**  Un polinomio univariante se puede dividir por otro para producir un cociente polinómico y un resto polinómico. Por ejemplo,



La división se puede realizar mediante una división larga. Es decir, se divide el término de mayor orden del dividendo por el término de mayor orden del divisor. El resultado es el primer término del cociente. A continuación, se multiplica el resultado por el divisor, se resta ese resultado del dividendo y se obtiene el resto de la respuesta dividiendo recursivamente la diferencia por el divisor. Se detiene cuando el orden del divisor supera el orden del dividendo y se declara que el dividendo es el resto. Además, si el dividendo llega a ser cero, se devuelve cero tanto como cociente como como resto.

Podemos diseñar un procedimiento div-poly basado en el modelo de add-poly y mul-poly . El procedimiento comprueba si los dos polys tienen la misma variable. Si es así, div-poly elimina la variable y pasa el problema a div-terms , que realiza la operación de división en listas de términos. Div-poly finalmente vuelve a unir la variable al resultado proporcionado por div-terms . Es conveniente diseñar div-terms para calcular tanto el cociente como el resto de una división. Div-terms puede tomar dos listas de términos como argumentos y devolver una lista de la lista de términos del cociente y la lista de términos del resto.

Complete la siguiente definición de términos div completando las expresiones faltantes. Use esto para implementar div-poly , que toma dos polys como argumentos y devuelve una lista de los polys de cociente y resto.

(define (div-terms L1 L2)   
  (if (vacía-lista-de-términos? L1)   
      (list (la-lista-de-términos-vacía) (la-lista-de-términos-vacía))   
      (let ((t1 (primer-término L1))   
            (t2 (primer-término L2)))   
        (if (> (orden t2) (orden t1))   
            (list (la-lista-de-términos-vacía) L1)   
            (let ((new-c (div (coef t1) (coef t2)))   
                  (new-o (- (orden t1) (orden t2))))   
              (let ((resto-del-resultado   
                     < *calcular resto del resultado recursivamente* >   
                     ))   
                < *formar resultado completo* >   
                ))))))

**[Jerarquías de tipos en álgebra simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_310)**

Nuestro sistema polinómico ilustra cómo los objetos de un tipo (polinomios) pueden ser, de hecho, objetos complejos que tienen objetos de muchos tipos diferentes como partes. Esto no plantea ninguna dificultad real a la hora de definir operaciones genéricas. Sólo necesitamos instalar operaciones genéricas apropiadas para realizar las manipulaciones necesarias de las partes de los tipos compuestos. De hecho, vimos que los polinomios forman una especie de "abstracción recursiva de datos", en el sentido de que las partes de un polinomio pueden ser, en sí mismas, polinomios. Nuestras operaciones genéricas y nuestro estilo de programación dirigida por datos pueden manejar esta complicación sin muchos problemas.

Por otra parte, el álgebra polinómica es un sistema en el que los tipos de datos no se pueden organizar de forma natural en una torre. Por ejemplo, es posible tener polinomios en *x* cuyos coeficientes sean polinomios en *y* . También es posible tener polinomios en *y* cuyos coeficientes sean polinomios en *x* . Ninguno de estos tipos está "por encima" del otro de forma natural, pero a menudo es necesario sumar elementos de cada conjunto. Hay varias formas de hacer esto. Una posibilidad es convertir un polinomio al tipo del otro expandiendo y reorganizando los términos de modo que ambos polinomios tengan la misma variable principal. Se puede imponer una estructura tipo torre sobre esto ordenando las variables y, por lo tanto, convirtiendo siempre cualquier polinomio en un"forma canónica" con la variable de mayor prioridad dominante y las variables de menor prioridad enterradas en los coeficientes. Esta estrategia funciona bastante bien, excepto que la conversión puede expandir un polinomio innecesariamente, lo que dificulta su lectura y quizás hace que sea menos eficiente trabajar con él. La estrategia de la torre ciertamente no es natural para este dominio ni para ningún dominio donde el usuario pueda inventar nuevos tipos dinámicamente usando tipos antiguos en varias formas de combinación, como funciones trigonométricas, series de potencias e integrales.

No debería sorprendernos que el controlLa coerción es un problema grave en el diseño de sistemas de manipulación algebraica a gran escala. Gran parte de la complejidad de dichos sistemas tiene que ver con las relaciones entre los distintos tipos. De hecho, es justo decir que todavía no comprendemos por completo la coerción. De hecho, todavía no comprendemos por completo el concepto de tipo de datos. Sin embargo, lo que sabemos nos proporciona principios de estructuración y modularidad poderosos para respaldar el diseño de sistemas grandes.

**Ejercicio 2.92.**   Imponiendo un orden a las variables, extienda el paquete polinomial de modo que la suma y multiplicación de polinomios funcione para polinomios con distintas variables. (¡Esto no es fácil!)

**[Ejercicio extendido: Funciones racionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_312)**

Podemos ampliar nuestro sistema aritmético genérico para incluir *funciones racionales* . Éstas son «fracciones» cuyo numerador y denominador son polinomios, como



El sistema debe poder sumar, restar, multiplicar y dividir funciones racionales, y realizar cálculos tales como:



(Aquí la suma se ha simplificado eliminando los factores comunes. La «multiplicación cruzada» ordinaria habría producido un polinomio de cuarto grado sobre un polinomio de quinto grado).

Si modificamos nuestro paquete aritmético-racional para que utilice operaciones genéricas, entonces hará lo que queremos, excepto el problema de reducir fracciones a sus términos más bajos.

**Ejercicio 2.93.**   Modifique el paquete aritmético-racional para utilizar operaciones genéricas, pero cambie make-rat de modo que no intente reducir fracciones a su mínima expresión. Pruebe su sistema llamando a make-rational en dos polinomios para producir una función racional.

(define p1 (hacer-polinomio 'x '((2 1)(0 1))))   
(define p2 (hacer-polinomio 'x '((3 1)(0 1))))   
(define rf (hacer-racional p2 p1))

Ahora suma rf a sí misma, usando add . Observarás que este procedimiento de suma no reduce las fracciones a su mínima expresión.

Podemos reducir las fracciones polinómicas a su mínima expresión utilizando la misma idea que utilizamos con los números enteros: modificando make-rat para dividir tanto el numerador como el denominador por su máximo común divisor. La noción deEl "máximo común divisor" tiene sentido para los polinomios. De hecho, podemos calcular el MCD de dos polinomios utilizando esencialmente el mismo algoritmo de Euclides que funciona para los números enteros. [60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_314) La versión entera es

(define (mcd a b)   
  (si (= b 0)   
      a   
      (mcd b (resto a b))))

Usando esto, podríamos hacer la modificación obvia para definir una operación MCD que funcione en listas de términos:

(define (términos-mcd a b)   
  (si (lista-de-términos-vacía? b)   
      a   
      (términos-mcd b (términos-resto a b))))

donde remainder-terms selecciona el componente restante de la lista devuelta por la operación de división de lista de términos div-terms que se implementó en el ejercicio  [2.91](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.91) .

**Ejercicio 2.94.**  Usando div-terms , implemente el procedimiento remainder-terms y úselo para definir gcd-terms como se indicó anteriormente. Ahora escriba un procedimiento gcd-poly que calcule el MCD polinomial de dos polinomios. (El procedimiento debería indicar un error si los dos polinomios no están en la misma variable). Instale en el sistema una operación genérica maximum-common-divisor que se reduzca a gcd-poly para polinomios y a mcd ordinario para números ordinarios. Como prueba, intente

(define p1 (haz-polinomio 'x'((4 1) (3 -1) (2 -2) (1 2))))   
(define p2 (haz-polinomio 'x'((3 1) (1 -1))))   
(máximo-común-divisor p1 p2)

y comprueba tu resultado manualmente.

**Ejercicio 2.95.**   Defina *P* 1 , *P* 2 y *P* 3 como los polinomios







Ahora definamos *Q* 1 como el producto de *P* 1 y *P* 2 y *Q* 2 como el producto de *P* 1 y *P* 3 , y utilicemos el máximo común divisor (ejercicio  [2.94](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.94) ) para calcular el MCD de *Q* 1 y *Q* 2 . Nótese que la respuesta no es la misma que *P* 1 . Este ejemplo introduce operaciones no enteras en el cálculo, lo que causa dificultades con el algoritmo MCD. [61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_317) Para entender lo que está sucediendo, intentemos rastrear los términos del MCD mientras calculamos el MCD o intentemos realizar la división a mano.

Podemos resolver el problema expuesto en el ejercicio  [2.95](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.95) si utilizamos la siguiente modificación del algoritmo MCD (que realmente sólo funciona en el caso de polinomios con coeficientes enteros). Antes de realizar cualquier división polinómica en el cálculo del MCD, multiplicamos el dividendo por un factor constante entero, elegido para garantizar que no surjan fracciones durante el proceso de división. Nuestra respuesta diferirá entonces del MCD real por un factor constante entero, pero esto no importa en el caso de reducir funciones racionales a su mínima expresión; el MCD se utilizará para dividir tanto el numerador como el denominador, por lo que el factor constante entero se cancelará.

Más precisamente, si *P* y *Q* son polinomios, sea *O* 1 el orden de *P* (es decir, el orden del término más grande de *P* ) y sea *O* 2 el orden de *Q* . Sea *c* el coeficiente principal de *Q* . Entonces se puede demostrar que, si multiplicamos *P* por el*factor entero c* 1+ *O* 1 - *O* 2 , el polinomio resultante se puede dividir por *Q* utilizando el algoritmo de términos div sin introducir ninguna fracción. La operación de multiplicar el dividendo por esta constante y luego dividir se denomina a veces*pseudodivisión* de *P* por *Q.* El resto de la división se llama *pseudoresto* .

**Ejercicio 2.96.**   a. Implemente el procedimiento pseudoremainder-terms , que es igual que remainder-terms excepto que multiplica el dividendo por el factor enterotizador descrito anteriormente antes de llamar a div-terms . Modifique gcd-terms para utilizar pseudoremainder-terms y verifique que maximum-common-divisor ahora produce una respuesta con coeficientes enteros en el ejemplo del ejercicio  [2.95](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.95) .

b. El MCD ahora tiene coeficientes enteros, pero son mayores que los de *P* 1 . Modifique los términos del MCD para que elimine los factores comunes de los coeficientes de la respuesta dividiendo todos los coeficientes por su máximo común divisor (entero).

Así pues, aquí se explica cómo reducir una función racional a sus términos más bajos:

* Calcular el MCD del numerador y el denominador, utilizando la versión de términos de MCD del ejercicio  [2.96](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.96) .
* Cuando obtengas el MCD, multiplica tanto el numerador como el denominador por el mismo factor entero antes de dividir por el MCD, de modo que la división por el MCD no introduzca ningún coeficiente no entero. Como factor puedes usar el coeficiente principal del MCD elevado a la potencia 1 + *O* 1 - *O* 2 , donde *O* 2 es el orden del MCD y *O* 1 es el máximo de los órdenes del numerador y el denominador. Esto garantizará que la división del numerador y el denominador por el MCD no introduzca ninguna fracción.
* El resultado de esta operación será un numerador y un denominador con coeficientes enteros. Los coeficientes normalmente serán muy grandes debido a todos los factores enteros, por lo que el último paso es eliminar los factores redundantes calculando el máximo común divisor (entero) de todos los coeficientes del numerador y del denominador y dividiendo por este factor.

**Ejercicio 2.97.**   a. Implemente este algoritmo como un procedimiento reduce-terms que toma dos listas de términos n y d como argumentos y devuelve una lista nn , dd , que son n y d reducidas a los términos más bajos a través del algoritmo dado anteriormente. También escriba un procedimiento reduce-poly , análogo a add-poly , que verifique si los dos polys tienen la misma variable. Si es así, reduce-poly quita la variable y pasa el problema a reduce-terms , luego vuelve a unir la variable a las dos listas de términos proporcionadas por reduce-terms .

b. Defina un procedimiento análogo a reduce-terms que haga lo que el make-rat original hacía para los números enteros:

(define (reduce-enteros n d)   
  (sea ((g (mcd n d)))   
    (lista (/ n g) (/ d g))))

y definir reduce como una operación genérica que llama a apply-generic para enviar a reduce-poly (para argumentos polinomiales ) o a reduce-integers (para argumentos scheme-number ). Ahora puede hacer fácilmente que el paquete aritmética racional reduzca fracciones a su mínima expresión haciendo que make-rat llame a reduce antes de combinar el numerador y el denominador dados para formar un número racional. El sistema ahora maneja expresiones racionales en números enteros o polinomiales. Para probar su programa, pruebe el ejemplo al comienzo de este ejercicio extendido:

(define p1 (hacer-polinomio 'x' ((1 1)(0 1))))   
(define p2 (hacer-polinomio 'x' ((3 1)(0 -1))))   
(define p3 (hacer-polinomio 'x' ((1 1))))   
(define p4 (hacer-polinomio 'x' ((2 1)(0 -1))))   
  
(define rf1 (hacer-racional p1 p2))   
(define rf2 (hacer-racional p3 p4))   
  
(agrega rf1 rf2)

Vea si obtiene la respuesta correcta, reducida correctamente a sus términos más bajos.

El cálculo del MCD es el núcleo de cualquier sistema que realice operaciones con funciones racionales. El algoritmo utilizado anteriormente, aunque matemáticamente sencillo, es extremadamente lento. La lentitud se debe en parte a la gran cantidad de operaciones de división y en parte al enorme tamaño de los coeficientes intermedios generados por las pseudodivisiones. Una de las áreas activas en el desarrollo de sistemas de manipulación algebraica es el diseño de mejores algoritmos para calcular MCD polinómicos. [62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_320)

[También](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_283) tenemos que proporcionar un procedimiento casi idéntico para manejar los tipos(esquema-número complejo).

[50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_285) Véase el ejercicio [2.82](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.82)para generalizaciones.

[51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_286) Si somos inteligentes, normalmente podemos arreglárnoslas con menos de*n*2procedimientos de coerción. Por ejemplo, si sabemos cómo convertir del tipo 1 al tipo 2 y del tipo 2 al tipo 3, entonces podemos usar este conocimiento para convertir del tipo 1 al tipo 3. Esto puede reducir en gran medida el número de procedimientos de coerción que necesitamos proporcionar explícitamente cuando agregamos un nuevo tipo al sistema. Si estamos dispuestos a incorporar la cantidad necesaria de sofisticación a nuestro sistema, podemos hacer que busque en el "grafo" de relaciones entre tipos y genere automáticamente aquellos procedimientos de coerción que se pueden inferir de los que se proporcionan explícitamente.

[52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_289) Esta afirmación, que también aparece en la primera edición de este libro, es tan cierta ahora como lo era cuando la escribimos hace doce años. Desarrollar un marco general útil para expresar las relaciones entre diferentes tipos de entidades (lo que los filósofos llaman "ontología") parece insolublemente difícil. La principal diferencia entre la confusión que existía hace diez años y la confusión que existe ahora es que ahora una variedad de teorías ontológicas inadecuadas se han incorporado a una plétora de lenguajes de programación correspondientemente inadecuados. Por ejemplo, gran parte de la complejidad deLos lenguajes de programación orientados a objetos (y las diferencias sutiles y confusas entre los lenguajes orientados a objetos contemporáneos) se centran en el tratamiento de operaciones genéricas sobre tipos interrelacionados. Nuestra propia discusión de los objetos computacionales en el capítulo 3 evita estos problemas por completo. Los lectores familiarizados con la programación orientada a objetos notarán que tenemos mucho que decir en el capítulo 3 sobre el estado local, pero ni siquiera mencionamos las "clases" o la "herencia". De hecho, sospechamos que estos problemas no pueden abordarse adecuadamente en términos de diseño de lenguajes de computadora únicamente, sin recurrir también al trabajo sobre representación del conocimiento y razonamiento automatizado.

[53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_295) Un número real se puede proyectar a un entero usando laprimitiva redonda , que devuelve el entero más cercano a su argumento.

[54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_298) Por otra parte, permitiremos polinomios cuyos coeficientes sean en sí mismos polinomios en otras variables. Esto nos dará esencialmente el mismo poder de representación que un sistema multivariado completo, aunque conduce a problemas de coerción, como se analiza más adelante.

[55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_299) Para los polinomios univariados, dar el valor de un polinomio en un conjunto dado de puntos puede ser una representación particularmente buena. Esto hace que la aritmética polinómica sea extremadamente simple. Para obtener, por ejemplo, la suma de dos polinomios representados de esta manera, solo necesitamos sumar los valores de los polinomios en los puntos correspondientes. Para volver a transformar a una representación más familiar, podemos usar laFórmula de interpolación de Lagrange, que muestra cómo recuperar los coeficientes de un polinomio de grado *n* dados los valores del polinomio en *n* + 1 puntos.

[56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_300) Esta operación es muy similar a launión de conjuntosque desarrollamos en el ejercicio  [2.62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.62). De hecho, si pensamos en los términos del polinomio como un conjunto ordenado según la potencia del indeterminado, entonces el programa que produce la lista de términos para una suma es casi idéntico ala operación de unión de conjuntos.

[57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_301) Para que esto funcione sin problemas, también deberíamos añadir a nuestro sistema aritmético genérico la capacidad de convertir un ``número'' en un polinomio considerándolo como un polinomio de grado cero cuyo coeficiente es el número. Esto es necesario si vamos a realizar operaciones como



lo cual requiere sumar el coeficiente *y* + 1 al coeficiente 2.

[58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_303) En estos ejemplos polinómicos, suponemos que hemos implementado el sistema aritmético genérico utilizando el mecanismo de tipos sugerido en el ejercicio [2.78](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.78). Por lo tanto, los coeficientes que son números ordinarios se representarán como los números mismos en lugar de como pares cuyocares el símboloscheme-number.

[59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_304) Aunque asumimos que las listas de términos están ordenadas, hemos implementadoadjoin-termpara simplementeincorporarel nuevo término a la lista de términos existente. Podemos hacerlo siempre que garanticemos que los procedimientos (comoadd-terms) que usanadjoin-termsiempre lo llamen con un término de orden superior al que aparece en la lista. Si no quisiéramos hacer tal garantía, podríamos haber implementadoadjoin-termpara que fuera similar aladjoin-setpara la representación de conjuntos en forma de lista ordenada (ejercicio [2.61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.61)).

[60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_314) El hecho de queEl algoritmo de Euclides funciona para polinomios y se formaliza en álgebra diciendo que los polinomios forman una especie de dominio algebraico llamado*Anillo euclidiano* . Un anillo euclidiano es un dominio que admite la adición, la resta y la multiplicación conmutativa, junto con una forma de asignar a cada elemento *x* del anillo una «medida» entera positiva *m* ( *x* ) con las propiedades de que *m* ( *x y* ) > *m* ( *x* ) para cualquier *x* e y *distinto* de cero y que, dados cualesquiera *x* e *y* , existe un *q* tal que *y* = *q x* + *r* y *r* = 0 o *m* ( *r* ) < *m* ( *x* ). Desde un punto de vista abstracto, esto es lo que se necesita para demostrar que el algoritmo de Euclides funciona. Para el dominio de los números enteros, la medida *m* de un número entero es el valor absoluto del propio número entero. Para el dominio de los polinomios, la medida de un polinomio es su grado.

[61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_317) En una implementación como la de MIT Scheme, esto produce un polinomio que es de hecho un divisor de*Q*1y*Q*2, pero con coeficientes racionales. En muchos otros sistemas Scheme, en los que la división de números enteros puede producir números decimales de precisión limitada, es posible que no obtengamos un divisor válido.

[62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_320) Un método extremadamente eficiente y elegante para calcularEl MCD polinomial fue descubierto porRichard Zippel (1979). El método es un algoritmo probabilístico, al igual que la prueba rápida de primalidad que analizamos en el capítulo 1. El libro de Zippel (1993) describe este método, junto con otras formas de calcular MCD polinómicos.

[**Capítulo 3**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_3)

[**Modularidad, objetos y estado**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_3)

|  |
| --- |
| METRO  (Aunque cambia, permanece inmóvil.)  Heráclito  Plus Ã§a change, plus c'est la même chose.  Alfonso Karr |

En los capítulos anteriores se introdujeron los elementos básicos a partir de los cuales se construyen los programas. Vimos cómo se combinan los procedimientos primitivos y los datos primitivos para construir entidades compuestas, y aprendimos que la abstracción es vital para ayudarnos a lidiar con la complejidad de los sistemas grandes. Pero estas herramientas no son suficientes para diseñar programas. La síntesis eficaz de programas también requiere principios organizativos que puedan guiarnos en la formulación del diseño general de un programa. En particular, necesitamos estrategias que nos ayuden a estructurar sistemas grandes de manera que sean *modulares* , es decir, que se puedan dividir "naturalmente" en partes coherentes que se puedan desarrollar y mantener por separado.

Una estrategia de diseño poderosa, que es particularmente apropiada para la construcción de programas para modelar sistemas físicos, es basar la estructura de nuestros programas en la estructura del sistema que se está modelando. Para cada objeto del sistema, construimos un objeto computacional correspondiente. Para cada acción del sistema, definimos una operación simbólica en nuestro modelo computacional. Nuestra esperanza al usar esta estrategia es que extender el modelo para acomodar nuevos objetos o nuevas acciones no requerirá cambios estratégicos en el programa, solo la adición de los nuevos análogos simbólicos de esos objetos o acciones. Si hemos tenido éxito en la organización de nuestro sistema, entonces para agregar una nueva característica o depurar una antigua tendremos que trabajar solo en una parte localizada del sistema.

En gran medida, entonces, la forma en que organizamos un programa grande está determinada por nuestra percepción del sistema que se va a modelar. En este capítulo, investigaremos dos estrategias organizacionales importantes que surgen de dos "visiones del mundo" bastante diferentes de la estructura de los sistemas. La primera estrategia organizacional se concentra en*objetos* , considerando un sistema grande como una colección de objetos distintos cuyo comportamiento puede cambiar con el tiempo. Una estrategia organizacional alternativa se concentra en los*flujos* de información que fluyen en el sistema, de forma muy similar a como un ingeniero eléctrico ve un sistema de procesamiento de señales.

Tanto el enfoque basado en objetos como el enfoque de procesamiento de flujos plantean importantes problemas lingüísticos en la programación. Con los objetos, debemos preocuparnos por cómo un objeto computacional puede cambiar y, sin embargo, mantener su identidad. Esto nos obligará a abandonar nuestro antiguo modelo de computación por sustitución (sección  [1.1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.5) ) en favor de un modelo más mecanicista pero menos manejable teóricamente.*modelo de entorno* de computación. Las dificultades de tratar con objetos, cambios e identidades son una consecuencia fundamental de la necesidad de lidiar con el tiempo en nuestros modelos computacionales. Estas dificultades se vuelven aún mayores cuando permitimos la posibilidad de ejecución concurrente de programas. El enfoque de flujo se puede explotar al máximo cuando desacoplamos el tiempo simulado en nuestro modelo del orden de los eventos que tienen lugar en la computadora durante la evaluación. Lo lograremos utilizando una técnica conocida como*evaluación* retrasada

[**3.1 Asignación y Estado Local**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_3.1)

Habitualmente consideramos que el mundo está poblado por objetos independientes, cada uno de los cuales tiene un estado que cambia con el tiempo. Se dice que un objeto "tiene estado" si su comportamiento está influenciado por su historia. Una cuenta bancaria, por ejemplo, tiene estado en el sentido de que la respuesta a la pregunta "¿Puedo retirar $100?" depende del historial de transacciones de depósito y retiro. Podemos caracterizar el estado de un objeto por uno o más*Variables de estado* , que entre ellas mantienen suficiente información sobre el historial para determinar el comportamiento actual del objeto. En un sistema bancario simple, podríamos caracterizar el estado de una cuenta por un saldo actual en lugar de recordar el historial completo de transacciones de la cuenta.

En un sistema compuesto por muchos objetos, estos rara vez son completamente independientes. Cada uno puede influir en los estados de los demás mediante interacciones que sirven para acoplar las variables de estado de un objeto con las de otros objetos. De hecho, la idea de que un sistema está compuesto por objetos separados es más útil cuando las variables de estado del sistema se pueden agrupar en subsistemas estrechamente acoplados que solo están débilmente acoplados a otros subsistemas.

Esta visión de un sistema puede ser un marco poderoso para organizar modelos computacionales del sistema. Para que un modelo de este tipo sea modular, debe descomponerse en objetos computacionales que modelen los objetos reales del sistema. Cada objeto computacional debe tener sus propias *variables de estado locales* que describan el estado del objeto real. Dado que los estados de los objetos del sistema que se modela cambian con el tiempo, las variables de estado de los objetos computacionales correspondientes también deben cambiar. Si elegimos modelar el flujo de tiempo en el sistema por el tiempo transcurrido en la computadora, entonces debemos tener una forma de construir objetos computacionales cuyo comportamiento cambie a medida que se ejecutan nuestros programas. En particular, si deseamos modelar variables de estado por nombres simbólicos ordinarios en el lenguaje de programación, entonces el lenguaje debe proporcionar una*operador de asignación* que nos permite cambiar el valor asociado a un nombre.

**[3.1.1 Variables de estado locales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.1.1)**

Para ilustrar lo que queremos decir con tener un objeto computacional con un estado variable en el tiempo, modelemos la situación de retirar dinero de una cuenta bancaria. Lo haremos usando un procedimiento retirar , que toma como argumento una cantidad a retirar. Si hay suficiente dinero en la cuenta para realizar el retiro, entonces retirar debe devolver el saldo restante después del retiro. De lo contrario, retirar debe devolver el mensaje *Fondos insuficientes.* Por ejemplo, si comenzamos con $100 en la cuenta, deberíamos obtener la siguiente secuencia de respuestas usando retirar :

(retirar 25)   
*75*  
(retirar 25)   
*50*  
(retirar 60)   
*"Fondos insuficientes"*  
(retirar 15)   
*35*

Observe que la expresión (retirar 25) , evaluada dos veces, arroja valores diferentes. Este es un nuevo tipo de comportamiento para un procedimiento. Hasta ahora, todos nuestros procedimientos podían considerarse como especificaciones para calcular funciones matemáticas. Una llamada a un procedimiento calculaba el valor de la función aplicada a los argumentos dados, y dos llamadas al mismo procedimiento con los mismos argumentos siempre producían el mismo resultado. [1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_321)

Para implementar retirar , podemos usar una variable balance para indicar el saldo de dinero en la cuenta y definir retirar como un procedimiento que accede a balance . El procedimiento retirar verifica si balance es al menos tan grande como el monto solicitado . Si es así, retirar disminuye balance por monto y devuelve el nuevo valor de balance . De lo contrario, retirar devuelve el mensaje *Fondos insuficientes* . Estas son las definiciones de balance y retirar :

(definir saldo 100)  
  
(define (monto a retirar)   
  (si (>= monto del saldo)   
      (begin (set! balance (- monto del saldo))   
             balance)   
      "Fondos insuficientes"))

La disminución del saldo se logra mediante la expresión

(establecer! saldo (- importe del saldo))

Aquí se utiliza la forma especial set!, cuya sintaxis es

(establecer! < *nombre* > < *nuevo-valor* >)

Aquí < *nombre* > es un símbolo y < *nuevo-valor* > es cualquier expresión. Set! cambia < *nombre* > para que su valor sea el resultado obtenido al evaluar < *nuevo-valor* >. En el caso que nos ocupa, estamos cambiando saldo para que su nuevo valor sea el resultado de restar cantidad al valor anterior de saldo . [2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_322)

Withdraw también utiliza la forma especial begin para hacer que se evalúen dos expresiones en el caso en que la prueba if sea verdadera: primero se decrementa el saldo y luego se devuelve el valor del saldo . En general, evaluar la expresión

(comienza < *exp 1* > < *exp 2* >  ... < *exp k* >)

hace que las expresiones < *exp 1* > a < *exp k* > se evalúen en secuencia y que el valor de la expresión final < *exp k* > se devuelva como el valor de toda la forma inicial . [3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_323)

Aunque retirar funciona como se desea, la variable balance presenta un problema. Como se especificó anteriormente, balance es un nombre definido en el entorno global y es de libre acceso para ser examinado o modificado por cualquier procedimiento. Sería mucho mejor si pudiéramos de alguna manera hacer que balance sea interno a retirar , de modo que retirar fuera el único procedimiento que pudiera acceder a balance directamente y cualquier otro procedimiento pudiera acceder a balance solo indirectamente (a través de llamadas a retirar ). Esto modelaría con mayor precisión la noción de que balance es una variable de estado local utilizada por retirar para realizar un seguimiento del estado de la cuenta.

Podemos hacer que el saldo sea interno para retirar reescribiendo la definición de la siguiente manera:

(define new-withdraw   
  (let ((saldo 100))   
    (lambda (monto)   
      (si (>= monto del saldo)   
          (begin (set! saldo (- monto del saldo))   
                 saldo)   
          "Fondos insuficientes"))))

Lo que hemos hecho aquí es utilizar let para establecer un entorno con una variable local balance , vinculada al valor inicial 100. Dentro de este entorno local, utilizamos lambda para crear un procedimiento que toma amount como argumento y se comporta como nuestro procedimiento de retiro anterior . Este procedimiento, devuelto como resultado de evaluar la expresión let , es new-withdraw , que se comporta exactamente de la misma manera que retreat pero cuya variable balance no es accesible por ningún otro procedimiento. [4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_324)

La combinación de set! con variables locales es la técnica de programación general que utilizaremos para construir objetos computacionales con estado local. Desafortunadamente, el uso de esta técnica plantea un problema serio: cuando presentamos por primera vez los procedimientos, también presentamos el modelo de sustitución de evaluación (sección  [1.1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.5) ) para proporcionar una interpretación de lo que significa la aplicación de un procedimiento. Dijimos que la aplicación de un procedimiento debería interpretarse como la evaluación del cuerpo del procedimiento con los parámetros formales reemplazados por sus valores. El problema es que, tan pronto como introducimos la asignación en nuestro lenguaje, la sustitución ya no es un modelo adecuado de aplicación de procedimientos. (Veremos por qué esto es así en la sección  [3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.3) ). Como consecuencia, técnicamente en este punto no tenemos forma de entender por qué el procedimiento new-withdraw se comporta como se afirmó anteriormente. Para comprender realmente un procedimiento como new-withdraw , necesitaremos desarrollar un nuevo modelo de aplicación de procedimientos. En la sección  [3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.2) presentaremos dicho modelo, junto con una explicación de set! y las variables locales. Primero, sin embargo, examinamos algunas variaciones sobre el tema establecido por new-withdraw .

El siguiente procedimiento, make-withdraw [,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_325) crea "procesadores de retiro". El parámetro formal balance en make-withdraw especifica la cantidad inicial de dinero en la cuenta.

(define (make-withdraw balance)   
  (lambda (amount)   
    (if (>= balance amount)   
        (begin (set! balance (- balance amount))   
               balance)   
        "Fondos insuficientes")))

Make-withdraw se puede utilizar de la siguiente manera para crear dos objetos W1 y W2 :

(define W1 (hacer-retirar 100))   
(define W2 (hacer-retirar 100))   
(W1 50)   
*50*  
(W2 70)   
*30*  
(W2 40)   
*"Fondos insuficientes"*  
(W1 40)   
*10*

Observe que W1 y W2 son objetos completamente independientes, cada uno con su propia variable de estado local balance . Los retiros de uno no afectan al otro.

También podemos crear objetos que gestionen depósitos y retiros, y así podemos representar cuentas bancarias simples. A continuación, se muestra un procedimiento que devuelve un ``objeto de cuenta bancaria'' con un saldo inicial especificado:

(define (make-account balance)   
  (define (monto de retiro)   
    (if (>= monto de saldo)   
        (begin (set! balance (- monto de saldo))   
               saldo)   
        "Fondos insuficientes"))   
  (define (monto de depósito)   
    (set! balance (+ monto de saldo))   
    saldo)   
  (define (dispatch m)   
    (cond ((eq? m 'withdraw) retirar)   
          ((eq? m 'deposit) depositar)   
          (else (error "Solicitud desconocida -- MAKE-ACCOUNT"   
                       m))))   
  dispatch)

Cada llamada a make-account configura un entorno con una variable de estado local balance . Dentro de este entorno, make-account define los procedimientos deposit y retreat que acceden a balance y un procedimiento adicional dispatch que toma un ``mensaje'' como entrada y devuelve uno de los dos procedimientos locales. El procedimiento dispatch en sí se devuelve como el valor que representa el objeto bank-count . Este es precisamente el*estilo de programación de paso de mensajes* que vimos en la sección  [2.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.3) , aunque aquí lo estamos usando junto con la capacidad de modificar variables locales.

La cuenta Make-account se puede utilizar de la siguiente manera:

(define acc (make-account 100))   
((acc 'retirar) 50)   
*50*  
((acc 'retirar) 60)   
*"Fondos insuficientes"*  
((acc 'depositar) 40)   
*90*  
((acc 'retirar) 60)   
*30*

Cada llamada a acc devuelve el procedimiento de depósito o retiro definido localmente , que luego se aplica al monto especificado . Como fue el caso con make-withdraw , otra llamada a make-account

(define acc2 (crear cuenta 100))

producirá un objeto de cuenta completamente separado, que mantiene su propio saldo local .

**Ejercicio 3.1.**   Una*El acumulador* es un procedimiento que se llama repetidamente con un único argumento numérico y acumula sus argumentos en una suma. Cada vez que se lo llama, devuelve la suma acumulada actual. Escriba un procedimientomake-accumulator que genera acumuladores, cada uno de los cuales mantiene una suma independiente. La entrada a make-accumulator debe especificar el valor inicial de la suma; por ejemplo

(define A (hacer acumulador 5))   
(A 10)   
*15*  
(A 10)   
*25*

**Ejercicio 3.2.**   En aplicaciones de prueba de software, es útil poder contar la cantidad de veces que se llama a un procedimiento determinado durante el transcurso de un cálculo. Escriba un procedimientomake-monitored que toma como entrada un procedimiento, f , que a su vez toma una entrada. El resultado devuelto por make-monitored es un tercer procedimiento, digamos mf , que lleva un registro de la cantidad de veces que ha sido llamado manteniendo un contador interno. Si la entrada a mf es el símbolo especial how-many-calls? , entonces mf devuelve el valor del contador. Si la entrada es el símbolo especial reset-count , entonces mf reinicia el contador a cero. Para cualquier otra entrada, mf devuelve el resultado de llamar a f en esa entrada e incrementa el contador. Por ejemplo, podríamos hacer una versión monitoreada del procedimiento sqrt :

(define s (make-monitored sqrt))   
  
(s 100)   
*10*  
  
(s '¿cuántas-llamadas?)   
*1*

**Ejercicio 3.3.**  Modifique el procedimiento de creación de cuentas para que cree cuentas protegidas con contraseña. Es decir, make-account debe tomar un símbolo como argumento adicional, como en

(define acc (make-account 100 'contraseña-secreta))

El objeto de cuenta resultante debe procesar una solicitud solo si está acompañada de la contraseña con la que se creó la cuenta y, de lo contrario, debe devolver una queja:

((acc 'contraseña-secreta' retirar) 40)   
*60*  
  
((acc 'otra-contraseña' depositar) 50)   
*"Contraseña incorrecta"*

**Ejercicio 3.4.**   Modifique el procedimiento make-account del ejercicio  [3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.3) agregando otra variable de estado local de modo que, si se accede a una cuenta más de siete veces consecutivas con una contraseña incorrecta, se invoque el procedimiento call-the-cops .

**[3.1.2 Los beneficios de introducir la asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.1.2)**

Como veremos, la introducción de la asignación en nuestro lenguaje de programación nos lleva a una maraña de difíciles cuestiones conceptuales. Sin embargo, considerar los sistemas como colecciones de objetos con estado local es una técnica poderosa para mantener un diseño modular. Como ejemplo simple, considere el diseño de un procedimiento rand que, siempre que se lo llama, devuelve un entero elegido al azar.

No está del todo claro qué se entiende por "elegido al azar". Lo que presumiblemente queremos es que las llamadas sucesivas a rand produzcan una secuencia de números que tenga propiedades estadísticas de distribución uniforme. No analizaremos aquí métodos para generar secuencias adecuadas. En cambio, supongamos que tenemos un procedimiento rand-update que tiene la propiedad de que si empezamos con un número dado *x* 1 y formamos

*x* 2  = (rand-actualizar  *x* 1 )   
*x* 3  = (rand-actualizar  *x* 2 )

entonces la secuencia de valores *x* 1 , *x* 2 , *x* 3 , ... , tendrá las propiedades estadísticas deseadas. [6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_330)

Podemos implementar rand como un procedimiento con una variable de estado local x que se inicializa con un valor fijo random-init . Cada llamada a rand calcula rand-update del valor actual de x , lo devuelve como un número aleatorio y también lo almacena como el nuevo valor de x .

(define rand   
  (let ((x random-init))   
    (lambda ()   
      (set! x (rand-update x))   
      x)))

Por supuesto, podríamos generar la misma secuencia de números aleatorios sin usar la asignación simplemente llamando a rand-update directamente. Sin embargo, esto significaría que cualquier parte de nuestro programa que usara números aleatorios tendría que recordar explícitamente el valor actual de x para pasarlo como argumento a rand-update . Para darse cuenta de lo molesto que sería esto, considere usar números aleatorios para implementar una técnica llamada*Simulación de Monte Carlo* .

El método de Monte Carlo consiste en elegir experimentos de muestra al azar de un conjunto grande y luego hacer deducciones sobre la base de las probabilidades estimadas a partir de la tabulación de los resultados de esos experimentos. Por ejemplo, podemos aproximarutilizando el hecho de que 6/ 2 es la probabilidad de que dos números enteros elegidos al azar no tengan factores en común; es decir, que su máximo común divisor sea 1. [7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_331) Para obtener la aproximación a , realizamos una gran cantidad de experimentos. En cada experimento elegimos dos números enteros al azar y realizamos una pruebapara ver si su MCD es 1. La fracción de veces que se pasa la prueba nos da nuestra estimación de 6/ 2 , y de esto obtenemos nuestra aproximación a .

El núcleo de nuestro programa es un procedimiento monte-carlo , que toma como argumentos la cantidad de veces que se debe intentar un experimento, junto con el experimento, representado como un procedimiento sin argumentos que devolverá verdadero o falso cada vez que se ejecute. Monte-carlo ejecuta el experimento durante la cantidad designada de ensayos y devuelve un número que indica la fracción de los ensayos en los que se encontró que el experimento era verdadero.

(definir (estimar-pi ensayos)   
  (sqrt (/ 6 (ensayos de monte-carlo prueba-cesaro))))  
(define (prueba de cesaro)   
   (= (mcd (rand) (rand)) 1))  
(define (ensayos de montecarlo experimento)   
  (define (iter ensayos-restantes ensayos-pasados)   
    (cond ((= ensayos-restantes 0)   
           (/ ensayos-pasados ​​ensayos))   
          ((experimento)   
           (iter (- ensayos-restantes 1) (+ ensayos-pasados ​​1)))   
          (else   
           (iter (- ensayos-restantes 1) ensayos-pasados))))   
  (iter ensayos 0))

Ahora intentemos el mismo cálculo usando rand-update directamente en lugar de rand , la forma en que nos veríamos obligados a proceder si no usáramos la asignación para modelar el estado local:

(define (estimate-pi ensayos)   
  (sqrt (/ 6 (random-mcd-test ensayos random-init))))   
(define (random-mcd-test ensayos initial-x)   
  (define (iter ensayos-restantes ensayos-pasados ​​x)   
    (let ((x1 (rand-update x)))   
      (let ((x2 (rand-update x1)))   
        (cond ((= ensayos-restantes 0)      
               (/ ensayos-pasados ​​ensayos))   
              ((= (mcd x1 x2) 1)   
               (iter (- ensayos-restantes 1)   
                     (+ ensayos-pasados ​​1)   
                     x2))   
              (else   
               (iter (- ensayos-restantes 1)   
                     ensayos-pasados   
                     ​​x2))))))   
  (iter ensayos 0 initial-x))

Aunque el programa sigue siendo simple, revela algunas dolorosas brechas de modularidad. En nuestra primera versión del programa, utilizando rand , podemos expresar el método de Monte Carlo directamente como un procedimiento general de Monte Carlo que toma como argumento un procedimiento de experimento arbitrario . En nuestra segunda versión del programa, sin estado local para el generador de números aleatorios, random-gcd-test debe manipular explícitamente los números aleatorios x1 y x2 y reciclar x2 a través del bucle iterativo como la nueva entrada para rand-update . Este manejo explícito de los números aleatorios entrelaza la estructura de acumulación de resultados de pruebas con el hecho de que nuestro experimento particular utiliza dos números aleatorios, mientras que otros experimentos de Monte Carlo pueden utilizar uno o tres números aleatorios. Incluso el procedimiento de nivel superior estimate-pi tiene que preocuparse por proporcionar un número aleatorio inicial. El hecho de que las entrañas del generador de números aleatorios se filtren a otras partes del programa nos dificulta aislar la idea de Monte Carlo para que pueda aplicarse a otras tareas. En la primera versión del programa, la asignación encapsula el estado del generador de números aleatorios dentro del procedimiento rand , de modo que los detalles de la generación de números aleatorios permanecen independientes del resto del programa.

El fenómeno general que ilustra el ejemplo de Monte Carlo es el siguiente: desde el punto de vista de una parte de un proceso complejo, las otras partes parecen cambiar con el tiempo. Tienen estados locales ocultos que varían con el tiempo. Si deseamos escribir programas informáticos cuya estructura refleje esta descomposición, creamos objetos computacionales (como cuentas bancarias y generadores de números aleatorios) cuyo comportamiento cambia con el tiempo. Modelamos el estado con variables de estado locales y modelamos los cambios de estado con asignaciones a esas variables.

Resulta tentador concluir este debate diciendo que, al introducir la asignación y la técnica de ocultar el estado en las variables locales, podemos estructurar los sistemas de una manera más modular que si todos los estados tuvieran que manipularse explícitamente, pasando parámetros adicionales. Desafortunadamente, como veremos, la historia no es tan sencilla.

**Ejercicio 3.5.**  *La integración de Monte Carlo* es un método para estimar integrales definidas por medio de la simulación de Monte Carlo. Considere calcular el área de una región del espacio descrita por un predicado *P* ( *x* , *y* ) que es verdadero para los puntos ( *x* , *y* ) en la región y falso para los puntos que no están en la región. Por ejemplo, la región contenida dentro de un círculo de radio 3 centrado en (5, 7) se describe por el predicado que prueba si ( *x* - 5) 2 + ( *y* - 7) 2 < 3 2 . Para estimar el área de la región descrita por dicho predicado, comience por elegir un rectángulo que contenga la región. Por ejemplo, un rectángulo con esquinas diagonalmente opuestas en (2, 4) y (8, 10) contiene el círculo anterior. La integral deseada es el área de esa porción del rectángulo que se encuentra en la región. Podemos estimar la integral eligiendo al azar puntos ( *x* , *y* ) que se encuentren en el rectángulo y probando *P* ( *x* , *y* ) para cada punto para determinar si el punto se encuentra en la región. Si probamos esto con muchos puntos, entonces la fracción de puntos que se encuentran en la región debería dar una estimación de la proporción del rectángulo que se encuentra en la región. Por lo tanto, multiplicar esta fracción por el área de todo el rectángulo debería producir una estimación de la integral.

Implementar la integración de Monte Carlo como un procedimientoestimación-integral que toma como argumentos un predicado P , límites superior e inferior x1 , x2 , y1 e y2 para el rectángulo, y la cantidad de ensayos a realizar para producir la estimación. Su procedimiento debe utilizar el mismo procedimiento de Montecarlo que se utilizó anteriormente para estimar . Utilice su estimación-integral para producir una estimación de midiendo el área de un círculo unitario.

Le resultará útil contar con un procedimiento que devuelva un número elegido al azar de un rango determinado. El siguiente procedimiento aleatorio dentro de un rango implementa esto en términos del procedimiento  aleatorio utilizado en la sección [1.2.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.6) , que devuelve un número no negativo menor que su entrada. [8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_333)

(define (aleatorio dentro del rango bajo alto)   
  (let ((rango (- alto bajo)))   
    (+ bajo (rango aleatorio))))

**Ejercicio 3.6.**  Resulta útil poder reiniciar un generador de números aleatorios para producir una secuencia a partir de un valor dado. Diseñe un nuevo procedimiento rand que se llame con un argumento que sea el símbolo generate o el símbolo reset y que se comporte de la siguiente manera: (rand 'generate) produce un nuevo número aleatorio; ((rand 'reset) < *new-value* >) reinicia la variable de estado interna al < *new-value* > designado. De este modo, al reiniciar el estado, se pueden generar secuencias repetibles. Son muy útiles para probar y depurar programas que utilizan números aleatorios.

**[3.1.3 Los costos de la introducción de la cesión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.1.3)**

Como hemos visto, la operación set! nos permite modelar objetos que tienen estado local. Sin embargo, esta ventaja tiene un precio. Nuestro lenguaje de programación ya no puede interpretarse en términos del modelo de sustitución de aplicación de procedimientos que presentamos en la sección  [1.1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.5) . Además, ningún modelo simple con propiedades matemáticas "agradables" puede ser un marco adecuado para tratar con objetos y asignaciones en lenguajes de programación.

Mientras no utilicemos asignaciones, dos evaluaciones del mismo procedimiento con los mismos argumentos producirán el mismo resultado, de modo que los procedimientos pueden considerarse como funciones matemáticas computacionales. La programación sin el uso de asignaciones, como hicimos a lo largo de los dos primeros capítulos de este libro, se conoce como*programación funcional* .

Para entender cómo la cesión complica las cosas, considere una versión simplificada del procedimiento  de creación y retiro de la sección [3.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.1) que no se molesta en verificar si hay un monto insuficiente:

(define (make-simplified-withdraw balance)   
  (lambda (amount)   
    (set! balance (- balance amount))   
    balance))   
(define W (make-simplified-withdraw 25))   
(W 20)   
*5*  
(W 10)   
*- 5*

Compare este procedimiento con el siguiente procedimiento make-decrementer , que no utiliza set !:

(define (make-decrementer balance)   
  (lambda (monto)   
    (- monto del balance)))

Make-decrementer devuelve un procedimiento que resta su entrada de una cantidad designada balance , pero no hay un efecto acumulado en llamadas sucesivas, como con make-simplified-withdraw :

(define D (hacer decrementar 25))   
(D 20)   
*5*  
(D 10)   
*15*

Podemos utilizar el modelo de sustitución para explicar cómo funciona el método make-decrementer . Por ejemplo, analicemos la evaluación de la expresión

((hacer-decrementar 25) 20)

Primero simplificamos el operador de la combinación sustituyendo 25 por balance en el cuerpo de make-decrementer . Esto reduce la expresión a

((lambda (cantidad) (- 25 cantidad)) 20)

Ahora aplicamos el operador sustituyendo 20 por cantidad en el cuerpo de la expresión lambda :

(- 25 20)

La respuesta final es 5.

Observe, sin embargo, lo que sucede si intentamos un análisis de sustitución similar con make-simplified-withdraw :

((hacer-simplificado-retirar 25) 20)

Primero simplificamos el operador sustituyendo 25 por balance en el cuerpo de make-simplified-withdraw . Esto reduce la expresión a [9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_335)

((lambda (cantidad) (establecer! balance (- 25 cantidad)) 25) 20)

Ahora aplicamos el operador sustituyendo 20 por cantidad en el cuerpo de la expresión lambda :

(¡establecer! equilibrio (- 25 20)) 25

Si nos adhiriéramos al modelo de sustitución, tendríamos que decir que el significado de la aplicación del procedimiento es primero establecer el balance en 5 y luego devolver 25 como el valor de la expresión. Esto nos da la respuesta incorrecta. Para obtener la respuesta correcta, tendríamos que distinguir de alguna manera la primera aparición de balance (¡antes del efecto del conjunto! ) de la segunda aparición de balance (¡después del efecto del conjunto! ), y el modelo de sustitución no puede hacer esto.

El problema aquí es que la sustitución se basa en última instancia en la noción de que los símbolos en nuestro lenguaje son esencialmente nombres para valores. Pero tan pronto como introducimos set! y la idea de que el valor de una variable puede cambiar, una variable ya no puede ser simplemente un nombre. Ahora una variable de alguna manera se refiere a un lugar donde se puede almacenar un valor, y el valor almacenado en este lugar puede cambiar. En la sección  [3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.2) veremos cómo los entornos desempeñan este papel de "lugar" en nuestro modelo computacional.

**[Misma identidad y cambio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_336)**

El problema que surge aquí es más profundo que el mero fracaso de un modelo de computación en particular. En cuanto introducimos cambios en nuestros modelos computacionales, muchas nociones que antes eran sencillas se vuelven problemáticas. Consideremos el concepto de que dos cosas son "lo mismo".

Supongamos que llamamos a make-decrementer dos veces con el mismo argumento para crear dos procedimientos:

(define D1 (make-decrementer 25))   
(define D2 (make-decrementer 25))

¿D1 y D2 son lo mismo? Una respuesta aceptable es sí, porque D1 y D2 tienen el mismo comportamiento computacional: cada uno es un procedimiento que resta su entrada de 25. De hecho, D1 podría sustituirse por D2 en cualquier cálculo sin cambiar el resultado.

Compare esto con hacer dos llamadas a make-simplified-withdraw :

(define W1 (hacer-simplificado-retirar 25))   
(define W2 (hacer-simplificado-retirar 25))

¿ Son W1 y W2 lo mismo? Seguramente no, porque las llamadas a W1 y W2 tienen efectos distintos, como lo demuestra la siguiente secuencia de interacciones:

(1.º de marzo de 2012)   
*5*  
(1.º de marzo de 2012)   
*- 15*  
(2.º de marzo de 2012)   
*5*

Aunque W1 y W2 son "iguales" en el sentido de que ambos se crean evaluando la misma expresión (make-simplified-withdraw 25) , no es cierto que W1 pueda sustituirse por W2 en cualquier expresión sin cambiar el resultado de evaluar la expresión.

Un lenguaje que admite el concepto de que ``igual puede sustituirse por igual'' en una expresión sin cambiar el valor de la expresión se dice que es*referencialmente transparente* . La transparencia referencial se viola cuando incluimos set! en nuestro lenguaje de programación. Esto hace que sea complicado determinar cuándo podemos simplificar expresiones mediante la sustitución de expresiones equivalentes. En consecuencia, el razonamiento sobre programas que utilizan la asignación se vuelve drásticamente más difícil.

Una vez que renunciamos a la transparencia referencial, la noción de lo que significa que los objetos computacionales sean “lo mismo” se vuelve difícil de capturar de una manera formal. De hecho, el significado de “lo mismo” en el mundo real que modelan nuestros programas no es claro en sí mismo. En general, podemos determinar que dos objetos aparentemente idénticos son de hecho “el mismo” solo modificando un objeto y luego observando si el otro objeto ha cambiado de la misma manera. Pero ¿cómo podemos saber si un objeto ha “cambiado” si no es observando el “mismo” objeto dos veces y viendo si alguna propiedad del objeto difiere de una observación a la siguiente? Por lo tanto, no podemos determinar el “cambio” sin alguna noción *a priori* de “igualdad”, y no podemos determinar la igualdad sin observar los efectos del cambio.

Como ejemplo de cómo surge este problema en la programación, considere la situación en la que Peter y Paul tienen una cuenta bancaria con $100 en ella. Hay una diferencia sustancial entre modelar esto como

(define peter-acc (crear-cuenta 100))   
(define paul-acc (crear-cuenta 100))

y modelarlo como

(define peter-acc (crear-cuenta 100))   
(define paul-acc peter-acc)

En la primera situación, las dos cuentas bancarias son distintas. Las transacciones que haga Peter no afectarán la cuenta de Paul, y viceversa. En la segunda situación, sin embargo, hemos definido paul-acc como *lo mismo* que peter-acc . En efecto, Peter y Paul tienen ahora una cuenta bancaria conjunta, y si Peter hace un retiro de peter-acc, Paul observará menos dinero en paul-acc . Estas dos situaciones similares pero distintas pueden causar confusión al construir modelos computacionales. Con la cuenta compartida, en particular, puede ser especialmente confuso que haya un objeto (la cuenta bancaria) que tenga dos nombres diferentes ( peter-acc y paul-acc ); si estamos buscando todos los lugares en nuestro programa donde se puede cambiar paul-acc , debemos recordar mirar también las cosas que cambian peter-acc . [10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_337)

Con referencia a las observaciones anteriores sobre la "identidad" y el "cambio", observe que si Pedro y Pablo sólo pudieran examinar sus saldos bancarios y no pudieran realizar operaciones que cambiaran el saldo, entonces la cuestión de si las dos cuentas son distintas sería discutible. En general, siempre que nunca modifiquemos los objetos de datos, podemos considerar que un objeto de datos compuesto es precisamente la totalidad de sus partes. Por ejemplo, un número racional se determina dando su numerador y su denominador. Pero esta visión ya no es válida en presencia de cambio, donde un objeto de datos compuesto tiene una "identidad" que es algo diferente de las partes que lo componen. Una cuenta bancaria sigue siendo "la misma" cuenta bancaria incluso si cambiamos el saldo haciendo un retiro; a la inversa, podríamos tener dos cuentas bancarias diferentes con la misma información de estado. Esta complicación es una consecuencia, no de nuestro lenguaje de programación, sino de nuestra percepción de una cuenta bancaria como un objeto. Por ejemplo, normalmente no consideramos un número racional como un objeto cambiante con identidad, de modo que pudiéramos cambiar el numerador y seguir teniendo «el mismo» número racional.

[**Los inconvenientes de la programación imperativa**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_338)

A diferencia de la programación funcional, la programación que hace un uso extensivo de la asignación se conoce como*Programación imperativa* . Además de generar complicaciones en los modelos computacionales, los programas escritos en estilo imperativo son susceptibles a errores que no pueden ocurrir en los programas funcionales. Por ejemplo, recuerde el programa factorial iterativo de la sección  [1.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.1) :

(define (factorial n)   
  (define (iter producto contador)   
    (si (> contador n)   
        producto   
        (iter (\* contador producto)   
              (+ contador 1))))   
  (iter 1 1))

En lugar de pasar argumentos en el bucle iterativo interno, podríamos adoptar un estilo más imperativo utilizando la asignación explícita para actualizar los valores de las variables producto y contador :

(define (factorial n)   
  (sea ((producto 1)   
        (contador 1))   
    (define (iter)   
      (si (> contador n)   
          producto   
          (comienza (¡establece! producto (\* contador producto))   
                 (¡establece! contador (+ contador 1))   
                 (iter))))   
    (iter)))

Esto no cambia los resultados producidos por el programa, pero sí introduce una trampa sutil. ¿Cómo decidimos el orden de las asignaciones? Resulta que el programa está escrito correctamente, pero escribir las asignaciones en el orden opuesto

(¡establecer! contador (+ contador 1))   
(¡establecer! producto (\* contador producto))

habría producido un resultado diferente e incorrecto. En general, la programación con asignación nos obliga a considerar cuidadosamente los órdenes relativos de las asignaciones para asegurarnos de que cada instrucción esté usando la versión correcta de las variables que se han cambiado. Este problema simplemente no surge en los programas funcionales. [11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_339) La complejidad de los programas imperativos se vuelve aún peor si consideramos aplicaciones en las que varios procesos se ejecutan simultáneamente. Volveremos a esto en la sección  [3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.4) . Primero, sin embargo, abordaremos el problema de proporcionar un modelo computacional para expresiones que involucran asignación y exploraremos los usos de objetos con estado local en el diseño de simulaciones.

**Ejercicio 3.7.**  Considere los objetos de cuenta bancaria creados por make-account , con la modificación de contraseña descrita en el ejercicio  [3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.3) . Suponga que nuestro sistema bancario requiere la capacidad de crear cuentas conjuntas. Defina un procedimientomake-joint que logra esto. Make-joint debe tomar tres argumentos. El primero es una cuenta protegida con contraseña. El segundo argumento debe coincidir con la contraseña con la que se definió la cuenta para que la operación make-joint continúe. El tercer argumento es una nueva contraseña. Make-joint es para crear un acceso adicional a la cuenta original utilizando la nueva contraseña. Por ejemplo, si peter-acc es una cuenta bancaria con contraseña open-sesame , entonces

(define paul-acc   
  (make-joint peter-acc 'ábrete-sésamo 'rosebud))

permitirá realizar transacciones en peter-acc utilizando el nombre paul-acc y la contraseña rosebud . Es posible que desee modificar su solución del ejercicio  [3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.3) para adaptarla a esta nueva función.

**Ejercicio 3.8.**  Cuando definimos el modelo de evaluación en la sección  [1.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.3) , dijimos que el primer paso para evaluar una expresión es evaluar sus subexpresiones. Pero nunca especificamos el orden en el que se deben evaluar las subexpresiones (por ejemplo, de izquierda a derecha o de derecha a izquierda). Cuando introducimos la asignación, el orden en el que se evalúan los argumentos de un procedimiento puede marcar una diferencia en el resultado. Defina un procedimiento simple f tal que la evaluación de (+ (f 0) (f 1)) devuelva 0 si los argumentos de + se evalúan de izquierda a derecha, pero devuelva 1 si los argumentos se evalúan de derecha a izquierda.

[1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_321) En realidad, esto no es del todo cierto. Una excepción fue lagenerador de números aleatorios en la sección  [1.2.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.6) . Otra excepción involucraba elTablas de operaciones/tipos que introdujimos en la sección  [2.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.3) , donde los valores de dos llamadas a get con los mismos argumentos dependían de llamadas intermedias a put . Por otro lado, hasta que introduzcamos la asignación, no tenemos forma de crear dichos procedimientos nosotros mismos.

[2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_322) El valor de una expresión set! depende de la implementación. Set! debe usarse solo por su efecto, no por su valor.

El nombre set! refleja una convención de nomenclatura utilizada en Scheme: las operaciones que cambian los valores de las variables (o que cambian las estructuras de datos, como veremos en la sección  [3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3) ) reciben nombres que terminan con un signo de exclamación. Esto es similar a la convención de designar predicados con nombres que terminan con un signo de interrogación.

[3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_323) Ya lo hemos utilizadocomienzan implícitamente en nuestros programas, porque en Scheme el cuerpo de un procedimiento puede ser una secuencia de expresiones. Además, la parte <consecuente> *de* cada cláusula en unLa expresión cond puede ser una secuencia de expresiones en lugar de una sola expresión.

[4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_324) En la jerga de los lenguajes de programación, se dice que la variablebalancees*encapsulado* dentro del procedimiento de nueva retirada . La encapsulación refleja el principio general de diseño del sistema conocido como*Principio de ocultación* : Se puede hacer que un sistema sea más modular y robusto protegiendo partes del sistema entre sí; es decir, proporcionando acceso a la información sólo a aquellas partes del sistema que tienen una "necesidad de saber".

[5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_325) A diferencia delmétodo new-withdrawanterior, no tenemos que usarletpara convertirel balanceen una variable local, ya que los parámetros formales ya son locales. Esto quedará más claro después de la discusión del modelo de evaluación del entorno en la sección [3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.2). (Véase también el ejercicio [3.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.10).)

[6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_330) Una forma común de implementarrand-updatees usar la regla de que*x*se actualiza a*ax*+*b*módulo*m*, donde*a*,*b*y*m*son números enteros elegidos apropiadamente. Capítulo 3 deKnuth 1981 incluye una extensa discusión de técnicas para generar secuencias de números aleatorios y establecer sus propiedades estadísticas. Nótese que el procedimiento rand-update calcula una función matemática: dada la misma entrada dos veces, produce la misma salida. Por lo tanto, la secuencia de números producida por rand-update ciertamente no es "aleatoria", si por "aleatoria" insistimos en que cada número en la secuencia no está relacionado con el número anterior. La relación entre la "aleatoriedad real" y la llamada*Las secuencias pseudoaleatorias* , que se producen mediante cálculos bien determinados y, sin embargo, tienen propiedades estadísticas adecuadas, son una cuestión compleja que involucra cuestiones difíciles de matemáticas y filosofía.Kolmogorov, Solomonoff y Chaitin han hecho grandes progresos en el esclarecimiento de estas cuestiones; se puede encontrar un análisis del tema en Chaitin 1975.

[Este](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_331) teorema se debe a E.Cesí ro. Ver apartado 4.5.2 deKnuth 1981 para una discusión y una prueba.

[8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_333) El esquema MIT prevé un procedimiento de este tipo. SiSi a random se le da un entero exacto (como en la sección  [1.2.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.6) ), devuelve un entero exacto, pero si se le da un valor decimal (como en este ejercicio), devuelve un valor decimal.

No sustituimos la ocurrencia debalanceen laset!porque el <*name*> en unset!no se evalúa. Si lo sustituyéramos, obtendríamos [(](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_335) set! 25 (- 25 amount)), lo cual no tiene sentido.

[10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_337) El fenómeno de que un único objeto computacional sea accedido por más de un nombre se conoce como*alias* . La situación de la cuenta bancaria conjunta ilustra un ejemplo muy simple de un alias. En la sección  [3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3) veremos ejemplos mucho más complejos, como estructuras de datos compuestas ``distintas'' que comparten partes. Pueden ocurrir errores en nuestros programas siOlvidamos que un cambio en un objeto también puede, como un "efecto secundario", cambiar un objeto "diferente" porque los dos objetos "diferentes" son en realidad un único objeto que aparece bajo diferentes alias. Estos llamados *errores de efectos secundarios* son tan difíciles de localizar y analizar que algunas personas han propuesto que los lenguajes de programación se diseñen de tal manera que no permitan efectos secundarios o alias.(Lampson y otros, 1981; Morris, Schmidt y Wadler, 1980).

[11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_339) En vista de esto, resulta irónico que la programación introductoria se enseñe con mayor frecuencia en un estilo altamente imperativo. Esto puede ser un vestigio de una creencia, común durante las décadas de 1960 y 1970, de que los programas que invocan procedimientos deben ser inherentemente menos eficientes que los programas que realizan tareas. (Steele (1977)(Esto desacredita este argumento.) Alternativamente, puede reflejar una visión de que la asignación paso a paso es más fácil de visualizar para los principiantes que la llamada a un procedimiento. Cualquiera sea la razón, a menudo carga a los programadores principiantes con preocupaciones del tipo "debería configurar esta variable antes o después de esa otra" que pueden complicar la programación y oscurecer las ideas importantes.

[**3.2 El modelo ambiental de evaluación**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_3.2)

Cuando introdujimos los procedimientos compuestos en el capítulo 1, utilizamos elmodelo de sustitución de evaluación (sección  [1.1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.5) ) para definir lo que se entiende por aplicar un procedimiento a los argumentos:

* Para aplicar un procedimiento compuesto a los argumentos, evalúe el cuerpo del procedimiento con cada parámetro formal reemplazado por el argumento correspondiente.

Una vez que admitimos la asignación en nuestro lenguaje de programación, dicha definición ya no es adecuada. En particular, la sección  [3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.3) argumentó que, en presencia de la asignación, una variable ya no puede considerarse simplemente como un nombre para un valor. Más bien, una variable debe designar de alguna manera un "lugar" en el que se puedan almacenar valores. En nuestro nuevo modelo de evaluación, estos lugares se mantendrán en estructuras llamadas*entornos* .

Un entorno es una secuencia de*marcos* . Cada marco es una tabla (posiblemente vacía) de*enlaces* , que asocian los nombres de las variables con sus valores correspondientes. (Un único marco puede contener como máximo un enlace para cualquier variable). Cada marco también tiene un puntero a su*entorno envolvente* , a menos que, a los efectos de la discusión, se considere que el marco es*mundial* . El*El valor de una variable* con respecto a un entorno es el valor dado por la vinculación de la variable en el primer marco del entorno que contiene una vinculación para esa variable. Si ningún marco de la secuencia especifica una vinculación para la variable, se dice que la variable es*libre* en el medio ambiente.

|  |
| --- |
| Fondo negro con letras blancas  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.1:**   Una estructura de entorno simple. |

La figura  [3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.1) muestra una estructura de entorno simple que consta de tres marcos, etiquetados I, II y III. En el diagrama, A, B, C y D son punteros a entornos. C y D apuntan al mismo entorno. Las variables z y x están ligadas en el marco II, mientras que y y x están ligadas en el marco I. El valor de x en el entorno D es 3. El valor de x con respecto al entorno B también es 3. Esto se determina de la siguiente manera: Examinamos el primer marco en la secuencia (marco III) y no encontramos un enlace para x , por lo que procedemos al entorno D que lo encierra y encontramos el enlace en el marco I. Por otro lado, el valor de x en el entorno A es 7, porque el primer marco en la secuencia (marco II) contiene un enlace de x a 7. Con respecto al entorno A, se dice que el enlace de x a 7 en el marco II es*sombrea* la unión de x a 3 en el cuadro I.

El entorno es crucial para el proceso de evaluación, porque determina el contexto en el que se debe evaluar una expresión. De hecho, se podría decir que las expresiones en un lenguaje de programación no tienen, en sí mismas, ningún significado. Más bien, una expresión adquiere un significado solo con respecto a un entorno en el que se evalúa. Incluso la interpretación de una expresión tan sencilla como (+ 1 1) depende de la comprensión de que uno está operando en un contexto en el que + es el símbolo de la adición. Por lo tanto, en nuestro modelo de evaluación siempre hablaremos de evaluar una expresión con respecto a un entorno. Para describir las interacciones con el intérprete, supondremos que hay unentorno global, que consta de un único marco (sin entorno envolvente) que incluye valores para los símbolos asociados con los procedimientos primitivos. Por ejemplo, la idea de que + es el símbolo de la suma se capta diciendo que el símbolo + está ligado en el entorno global al procedimiento primitivo de suma.

**[3.2.1 Las reglas de evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.2.1)**

La especificación general de cómo el intérprete evalúa una combinación sigue siendo la misma que cuando la presentamos por primera vez en la sección  [1.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.3) :

* Para evaluar una combinación:

1. Evalúa las subexpresiones de la combinación. [12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_342)

2. Aplique el valor de la subexpresión del operador a los valores de las subexpresiones del operando.

El modelo ambiental de evaluación reemplaza al modelo de sustitución al especificar lo que significa aplicar un procedimiento compuesto a los argumentos.

En el modelo de evaluación del entorno, un procedimiento siempre es un par formado por un código y un puntero a un entorno. Los procedimientos se crean de una sola manera: evaluando una expresión lambda .Esto produce un procedimiento cuyo código se obtiene del texto de la expresión lambda y cuyo entorno es el entorno en el que se evaluó la expresión lambda para producir el procedimiento. Por ejemplo, considere la definición del procedimiento

(define (cuadrado x)   
  (\* x x))

evaluado en el entorno global. La sintaxis de definición del procedimiento es simplemente azúcar sintáctico para una expresión lambda implícita subyacente . Habría sido equivalente haber utilizado

(define cuadrado   
  (lambda (x) (\* x x)))

que evalúa (lambda (x) (\* xx)) y vincula el cuadrado al valor resultante, todo en el entorno global.

La figura  [3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.2) muestra el resultado de evaluar esta expresión de definición . El objeto de procedimiento es un par cuyo código especifica que el procedimiento tiene un parámetro formal, a saber, x , y un cuerpo de procedimiento (\* xx) . La parte del entorno del procedimiento es un puntero al entorno global, ya que ese es el entorno en el que se evaluó la expresión lambda para producir el procedimiento. Se ha agregado un nuevo enlace, que asocia el objeto de procedimiento con el símbolo cuadrado , al marco global. En general, define crea definiciones agregando enlaces a los marcos.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.2:**   Estructura del entorno producida al evaluar (define (cuadrado x) (\* xx)) en el entorno global. |

Ahora que hemos visto cómo se crean los procedimientos, podemos describir cómo se aplican. El modelo de entorno especifica: Para aplicar un procedimiento a los argumentos, cree un nuevo entorno que contenga un marco que vincule los parámetros a los valores de los argumentos. El entorno que encierra este marco es el entorno especificado por el procedimiento. Ahora, dentro de este nuevo entorno, evalúe el cuerpo del procedimiento.

Para mostrar cómo se sigue esta regla, la figura  [3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.3) ilustra la estructura del entorno creada al evaluar la expresión (cuadrado 5) en el entorno global, donde cuadrado es el procedimiento generado en la figura  [3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.2) . La aplicación del procedimiento da como resultado la creación de un nuevo entorno, etiquetado E1 en la figura, que comienza con un marco en el que x , el parámetro formal para el procedimiento, está ligado al argumento 5. El puntero que apunta hacia arriba desde este marco muestra que el entorno que encierra el marco es el entorno global. El entorno global se elige aquí, porque este es el entorno que se indica como parte del objeto de procedimiento cuadrado . Dentro de E1, evaluamos el cuerpo del procedimiento, (\* xx) . Dado que el valor de x en E1 es 5, el resultado es (\* 5 5) , o 25.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.3:**   Entorno creado al evaluar (cuadrado 5) en el entorno global. |

El modelo de entorno de aplicación de procedimientos se puede resumir en dos reglas:

* Un objeto de procedimiento se aplica a un conjunto de argumentos mediante la construcción de un marco, vinculando los parámetros formales del procedimiento a los argumentos de la llamada y, a continuación, evaluando el cuerpo del procedimiento en el contexto del nuevo entorno construido. El nuevo marco tiene como entorno envolvente la parte del entorno del objeto de procedimiento que se está aplicando.
* Un procedimiento se crea evaluando una expresión lambda relativa a un entorno determinado. El objeto de procedimiento resultante es un par formado por el texto de la expresión lambda y un puntero al entorno en el que se creó el procedimiento.

También especificamos que definir un símbolo usando define crea un enlace en el marco del entorno actual y asigna al símbolo el valor indicado. [13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_343) Finalmente, especificamos el comportamiento de set! , la operación que nos obligó a introducir el modelo de entorno en primer lugar. Evaluar la expresión (set! < *variable* > < *valor* >) en algún entorno ubica el enlace de la variable en el entorno y cambia ese enlace para indicar el nuevo valor. Es decir, uno encuentra el primer marco en el entorno que contiene un enlace para la variable y modifica ese marco. Si la variable no está enlazada en el entorno, entonces set! señala un error.

Estas reglas de evaluación, aunque considerablemente más complejas que el modelo de sustitución, siguen siendo razonablemente sencillas. Además, el modelo de evaluación, aunque abstracto, proporciona una descripción correcta de cómo el intérprete evalúa las expresiones. En el capítulo 4 veremos cómo este modelo puede servir como modelo para implementar un intérprete funcional. Las siguientes secciones desarrollan los detalles del modelo mediante el análisis de algunos programas ilustrativos.

[**3.2.2 Aplicación de procedimientos simples**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_3.2.2)

Cuando introdujimos el modelo de sustitución en la sección  [1.1.5,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.5) mostramos cómo la combinación (f 5) evalúa 136, dadas las siguientes definiciones de procedimiento:

(define (cuadrado x)   
  (\* x x))   
(define (suma de cuadrados x y)   
  (+ (cuadrado x) (cuadrado y)))   
(define (f a)   
  (suma de cuadrados (+ a 1) (\* a 2)))

Podemos analizar el mismo ejemplo utilizando el modelo de entorno. La figura  [3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.4) muestra los tres objetos de procedimiento creados mediante la evaluación de las definiciones de f , cuadrado y suma de cuadrados en el entorno global. Cada objeto de procedimiento consta de un código, junto con un puntero al entorno global.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.4:**   Objetos de procedimiento en el marco global. |

En la figura  [3.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.5) vemos la estructura del entorno creada al evaluar la expresión (f 5) . La llamada a f crea un nuevo entorno E1 que comienza con un marco en el que a , el parámetro formal de f , está ligado al argumento 5. En E1, evaluamos el cuerpo de f :

(suma de cuadrados (+ a 1) (\* a 2))

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.5:**   Entornos creados mediante la evaluación de (f 5) utilizando los procedimientos de la figura  [3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.4) . |

Para evaluar esta combinación, primero evaluamos las subexpresiones. La primera subexpresión, suma-de-cuadrados , tiene un valor que es un objeto de procedimiento. (Observe cómo se encuentra este valor: primero buscamos en el primer marco de E1, que no contiene ningún enlace para suma-de-cuadrados . Luego procedemos al entorno envolvente, es decir, el entorno global, y encontramos el enlace que se muestra en la figura  [3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.4) ). Las otras dos subexpresiones se evalúan aplicando las operaciones primitivas + y \* para evaluar las dos combinaciones (+ a 1) y (\* a 2) para obtener 6 y 10, respectivamente.

Ahora aplicamos el objeto de procedimiento suma-de-cuadrados a los argumentos 6 y 10. Esto da como resultado un nuevo entorno E2 en el que los parámetros formales x e y están ligados a los argumentos. Dentro de E2 evaluamos la combinación (+ (cuadrado x) (cuadrado y)) . Esto nos lleva a evaluar (cuadrado x) , donde cuadrado se encuentra en el marco global y x es 6. Una vez más, configuramos un nuevo entorno, E3, en el que x está ligado a 6, y dentro de este evaluamos el cuerpo de cuadrado , que es (\* xx) . También como parte de la aplicación de suma-de-cuadrados , debemos evaluar la subexpresión (cuadrado y) , donde y es 10. Esta segunda llamada a cuadrado crea otro entorno, E4, en el que x , el parámetro formal de cuadrado , está ligado a 10. Y dentro de E4 debemos evaluar (\* xx) .

El punto importante a observar es que cada llamada a square crea un nuevo entorno que contiene un enlace para x . Aquí podemos ver cómo los diferentes marcos sirven para mantener separadas las diferentes variables locales, todas llamadas x . Observe que cada marco creado por square apunta al entorno global, ya que este es el entorno indicado por el objeto de procedimiento square .

Después de evaluar las subexpresiones, se devuelven los resultados. Los valores generados por las dos llamadas a square se suman mediante sum-of-squares y este resultado se devuelve mediante f . Dado que aquí nos centraremos en las estructuras del entorno, no nos detendremos en cómo se pasan estos valores devueltos de una llamada a otra; sin embargo, este también es un aspecto importante del proceso de evaluación y volveremos a él en detalle en el capítulo 5.

**Ejercicio 3.9.**  En la sección  [1.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.1) utilizamos el modelo de sustitución para analizar dos procedimientos para calcular factoriales, una versión recursiva

(define (factorial n)   
  (si (= n 1)   
      1   
      (\* n (factorial (- n 1)))))

y una versión iterativa

(define (factorial n)   
  (iterador de hechos 1 1 n))   
(define (iterador de hechos producto contador recuento máximo)   
  (si (> contador recuento máximo)   
      producto   
      (iterador de hechos (\* contador producto)   
                 (+ contador 1)   
                 recuento máximo)))

Muestre las estructuras ambientales creadas al evaluar (factorial 6) utilizando cada versión del procedimiento factorial . [14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_345)

**[3.2.3 Los marcos como repositorio del estado local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.2.3)**

Podemos recurrir al modelo de entorno para ver cómo se pueden utilizar los procedimientos y la asignación para representar objetos con estado local. Como ejemplo, considere el ``procesador de retiro'' de la sección  [3.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.1) creado al llamar al procedimiento

(define (make-withdraw balance)   
  (lambda (amount)   
    (if (>= balance amount)   
        (begin (set! balance (- balance amount))   
               balance)   
        "Fondos insuficientes")))

Describamos la evaluación de

(definir W1 (hacer-retirar 100))

seguido por

(W1 50)   
*50*

La figura  [3.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.6) muestra el resultado de definir el procedimiento make-withdraw en el entorno global. Esto produce un objeto de procedimiento que contiene un puntero al entorno global. Hasta ahora, esto no es diferente de los ejemplos que ya hemos visto, excepto que el cuerpo del procedimiento es en sí mismo una expresión lambda .

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.6:**   Resultado de la definición de make-withdraw en el entorno global. |

La parte interesante del cálculo ocurre cuando aplicamos el procedimiento make-withdraw a un argumento:

(definir W1 (hacer-retirar 100))

Comenzamos, como es habitual, configurando un entorno E1 en el que el parámetro formal balance está ligado al argumento 100. Dentro de este entorno, evaluamos el cuerpo de make-withdraw , es decir, la expresión lambda . Esto construye un nuevo objeto de procedimiento, cuyo código es el especificado por la lambda y cuyo entorno es E1, el entorno en el que se evaluó la lambda para producir el procedimiento. El objeto de procedimiento resultante es el valor devuelto por la llamada a make-withdraw . Este está ligado a W1 en el entorno global, ya que la propia define se está evaluando en el entorno global. La figura  [3.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.7) muestra la estructura del entorno resultante.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.7:**   Resultado de la evaluación (define W1 (make-withdraw 100)) . |

Ahora podemos analizar qué sucede cuando se aplica W1 a un argumento:

(W1 50)   
*50*

Comenzamos construyendo un marco en el que cantidad , el parámetro formal de W1 , está ligado al argumento 50. El punto crucial que hay que observar es que este marco tiene como entorno envolvente no el entorno global, sino el entorno E1, porque este es el entorno que especifica el objeto de procedimiento W1 . Dentro de este nuevo entorno, evaluamos el cuerpo del procedimiento:

(si (>= monto del saldo)   
    (inicio (establecer! saldo (- monto del saldo))   
           saldo)   
    "Fondos insuficientes")

La estructura del entorno resultante se muestra en la figura  [3.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.8) . La expresión que se está evaluando hace referencia tanto a amount como a balance . amount se encontrará en el primer marco del entorno, mientras que balance se encontrará siguiendo el puntero del entorno circundante hasta E1.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.8:**   Entornos creados al aplicar el objeto de procedimiento W1 . |

Cuando se ejecuta set!, se cambia el enlace de balance en E1. Al completarse la llamada a W1 , balance es 50 y el marco que contiene balance todavía está apuntado por el objeto de procedimiento W1 . El marco que enlaza a amount (en el que ejecutamos el código que cambió balance ) ya no es relevante, ya que la llamada de procedimiento que lo construyó ha finalizado y no hay punteros a ese marco desde otras partes del entorno. La próxima vez que se llame a W1 , esto creará un nuevo marco que enlaza a amount y cuyo entorno envolvente es E1. Vemos que E1 sirve como el ``lugar'' que contiene la variable de estado local para el objeto de procedimiento W1 . La figura  [3.9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.9) muestra la situación después de la llamada a W1 .

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.9:**   Entornos después de la llamada a W1 . |

Observe lo que sucede cuando creamos un segundo objeto ``withdraw'' haciendo otra llamada a make-withdraw :

(definir W2 (hacer-retirar 100))

Esto produce la estructura de entorno de la figura  [3.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.10) , que muestra que W2 es un objeto de procedimiento, es decir, un par con algún código y un entorno. El entorno E2 para W2 fue creado por la llamada a make-withdraw . Contiene un marco con su propio enlace local para balance . Por otro lado, W1 y W2 tienen el mismo código: el código especificado por la expresión lambda en el cuerpo de make-withdraw . [15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_346) Vemos aquí por qué W1 y W2 se comportan como objetos independientes. Las llamadas a W1 hacen referencia a la variable de estado balance almacenada en E1, mientras que las llamadas a W2 hacen referencia a la variable de estado balance almacenada en E2. Por lo tanto, los cambios en el estado local de un objeto no afectan al otro objeto.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.10:**   Uso de (define W2 (make-withdraw 100)) para crear un segundo objeto. |

**Ejercicio 3.10.**   En el procedimiento make-withdraw se crea la variable local balance como parámetro de make-withdraw . También podríamos crear la variable local state de forma explícita, utilizando let , de la siguiente manera:

(define (make-withdraw initial-amount)   
  (let ((balance initial-amount))   
    (lambda (amount)   
      (if (>= balance amount)   
          (begin (set! balance (- balance amount))   
                 balance)   
          "Fondos insuficientes"))))

Recuerde de la sección  [1.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.2) que let es simplemente un azúcar sintáctico para una llamada a un procedimiento:

(let ((< *var* > < *exp* >)) < *cuerpo* >)

se interpreta como una sintaxis alternativa para

((lambda (< *var* >) < *cuerpo* >) < *exp* >)

Utilice el modelo de entorno para analizar esta versión alternativa de make-withdraw , dibujando figuras como las de arriba para ilustrar las interacciones.

(define W1 (hacer-retirar 100))   
  
(W1 50)   
  
(define W2 (hacer-retirar 100))

Demuestre que las dos versiones de make-withdraw crean objetos con el mismo comportamiento. ¿En qué se diferencian las estructuras del entorno de las dos versiones?

**[3.2.4 Definiciones internas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.2.4)**

La sección  [1.1.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.8) introdujo la idea de que los procedimientos pueden tener definiciones internas, lo que conduce a una estructura de bloques como en el siguiente procedimiento para calcular raíces cuadradas:

(define (sqrt x)   
  (define (¿suficientemente bueno? conjetura)   
    (< (abs (- (conjetura al cuadrado) x)) 0.001))   
  (define (mejora conjetura)   
    (conjetura promedio (/ x conjetura)))   
  (define (sqrt-iter conjetura)   
    (si (¿suficientemente bueno? conjetura)   
        conjetura   
        (sqrt-iter (mejora conjetura))))   
  (sqrt-iter 1.0))

Ahora podemos utilizar el modelo de entorno para ver por qué estas definiciones internas se comportan como se desea. La figura  [3.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.11) muestra el punto en la evaluación de la expresión (sqrt 2) donde se ha llamado por primera vez al procedimiento interno ¿suficientemente bueno? con un valor de conjetura igual a 1.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.11:**   Procedimiento Sqrt con definiciones internas. |

Observe la estructura del entorno. Sqrt es un símbolo en el entorno global que está vinculado a un objeto de procedimiento cuyo entorno asociado es el entorno global. Cuando se llamó a sqrt , se formó un nuevo entorno E1, subordinado al entorno global, en el que el parámetro x está vinculado a 2. El cuerpo de sqrt se evaluó entonces en E1. Dado que la primera expresión en el cuerpo de sqrt es

(define (¿suficientemente bueno? adivina)   
  (< (abs (- (cuadrado adivinado) x)) 0,001))

Al evaluar esta expresión se definió el procedimiento ¿suficientemente bueno? en el entorno E1. Para ser más precisos, se agregó el símbolo ¿suficientemente bueno? al primer marco de E1, vinculado a un objeto de procedimiento cuyo entorno asociado es E1. De manera similar, improve y sqrt-iter se definieron como procedimientos en E1. Para mayor concisión, la figura  [3.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.11) muestra solo el objeto de procedimiento para ¿suficientemente bueno ?.

Después de definir los procedimientos locales, se evaluó la expresión (sqrt-iter 1.0) , todavía en el entorno E1. Por lo tanto, se llamó al objeto de procedimiento vinculado a sqrt-iter en E1 con 1 como argumento. Esto creó un entorno E2 en el que guess , el parámetro de sqrt-iter , está vinculado a 1. A su vez, sqrt-iter llamó a good-enough? con el valor de guess (de E2) como argumento para good-enough? . Esto creó otro entorno, E3, en el que guess (el parámetro de good-enough? ) está vinculado a 1. Aunque sqrt-iter y good-enough? tienen ambos un parámetro llamado guess , estas son dos variables locales distintas ubicadas en marcos diferentes. Además, E2 y E3 tienen ambos E1 como su entorno envolvente, porque los procedimientos sqrt-iter y good-enough? tienen ambos E1 como parte de su entorno. Una consecuencia de esto es que el símbolo x que aparece en el cuerpo de ¿suficientemente bueno? hará referencia a la vinculación de x que aparece en E1, es decir, el valor de x con el que se llamó al procedimiento sqrt original . El modelo de entorno explica así las dos propiedades clave que hacen que las definiciones de procedimientos locales sean una técnica útil para modularizar programas:

* Los nombres de los procedimientos locales no interfieren con los nombres externos al procedimiento que los contiene, porque los nombres de los procedimientos locales estarán ligados en el marco que el procedimiento crea cuando se ejecuta, en lugar de estar ligados en el entorno global.
* Los procedimientos locales pueden acceder a los argumentos del procedimiento que los incluye, simplemente utilizando los nombres de los parámetros como variables libres. Esto se debe a que el cuerpo del procedimiento local se evalúa en un entorno que está subordinado al entorno de evaluación del procedimiento que los incluye.

**Ejercicio 3.11.**  En la sección  [3.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.2.3) vimos cómo el modelo de entorno describía el comportamiento de los procedimientos con estado local. Ahora hemos visto cómo funcionan las definiciones internas. Un procedimiento de paso de mensajes típico contiene ambos aspectos. Considere el procedimiento de cuenta bancaria de la sección  [3.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.1) :

(define (make-account balance)   
  (define (monto de retiro)   
    (if (>= monto de saldo)   
        (begin (set! balance (- monto de saldo))   
               saldo)   
        "Fondos insuficientes"))   
  (define (monto de depósito)   
    (set! balance (+ monto de saldo))   
    saldo)   
  (define (dispatch m)   
    (cond ((eq? m 'withdraw) retirar)   
          ((eq? m 'deposit) depositar)   
          (else (error "Solicitud desconocida -- MAKE-ACCOUNT"   
                       m))))   
  dispatch)

Muestra la estructura del entorno generada por la secuencia de interacciones.

(define acc (crear-cuenta 50))   
  
((acc 'depositar) 40)   
*90*  
  
((acc 'retirar) 60)   
*30*

¿Dónde se guarda el estado local de la cuenta ? Supongamos que definimos otra cuenta

(define acc2 (crear cuenta 100))

¿Cómo se mantienen separados los estados locales de las dos cuentas? ¿Qué partes de la estructura del entorno comparten acc y acc2 ?

[12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_342) La asignación introduce una sutileza en el paso 1 de la regla de evaluación. Como se muestra en el ejercicio [3.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.8), la presencia de la asignación nos permite escribir expresiones que producirán valores diferentes según el orden en que se escriban las subexpresiones en una combinación.Se evalúan. Por lo tanto, para ser precisos, deberíamos especificar un orden de evaluación en el paso 1 (por ejemplo, de izquierda a derecha o de derecha a izquierda). Sin embargo, este orden siempre debería considerarse un detalle de implementación y nunca se deberían escribir programas que dependan de un orden particular. Por ejemplo, un compilador sofisticado podría optimizar un programa variando el orden en el que se evalúan las subexpresiones.

[13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_343) Si ya existe una vinculación para la variable en el marco actual, entonces se cambia la vinculación. Esto es conveniente porque permite la redefinición de símbolos; sin embargo, también significa quedefinepara cambiar valores, y esto plantea los problemas de asignación sin usar explícitamente¡Listo! Por este motivo, algunas personas prefieren redefinir símbolos existentes para señalar errores o advertencias.

[14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_345) El modelo de entorno no aclarará nuestra afirmación en la sección [1.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.1)de que el intérprete puede ejecutar un procedimiento comofact-iteren una cantidad constante de espacio utilizando la recursión de cola. Hablaremos de la recursión de cola cuandoTrataremos la estructura de control del intérprete en la sección  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4) .

[15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_346) El hecho de queW1yW2compartan el mismo código físico almacenado en la computadora o de que cada uno conserve una copia del código es un detalle de la implementación. En el caso del intérprete que implementamos en el capítulo 4, el código es, de hecho, compartido.

[**3.3 Modelado con datos mutables**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_3.3)

El capítulo 2 se ocupó de los datos compuestos como un medio para construir objetos computacionales que tienen varias partes, con el fin de modelar objetos del mundo real que tienen varios aspectos. En ese capítulo introdujimos la disciplina de la abstracción de datos, según la cual las estructuras de datos se especifican en términos de constructores, que crean objetos de datos, y selectores, que acceden a las partes de los objetos de datos compuestos. Pero ahora sabemos que hay otro aspecto de los datos que el capítulo 2 no abordó. El deseo de modelar sistemas compuestos de objetos que tienen un estado cambiante nos lleva a la necesidad de modificar objetos de datos compuestos, así como de construirlos y seleccionarlos. Para modelar objetos compuestos con estado cambiante, diseñaremos abstracciones de datos que incluyan, además de selectores y constructores, operaciones llamadas*mutadores* , que modifican los objetos de datos. Por ejemplo, para modelar un sistema bancario es necesario cambiar los saldos de las cuentas. Por lo tanto, una estructura de datos para representar cuentas bancarias podría admitir una operación

(establecer-saldo! < *cuenta* > < *nuevo-valor* >)

que cambia el saldo de la cuenta designada al nuevo valor designado. Los objetos de datos para los que se definen mutadores se conocen como *objetos de datos mutables* .

En el capítulo 2 se introdujeron los pares como un "pegamento" de uso general para sintetizar datos compuestos. Comenzamos esta sección definiendo mutadores básicos para pares, de modo que los pares puedan servir como bloques de construcción para construir objetos de datos mutables. Estos mutadores mejoran en gran medida el poder de representación de los pares, lo que nos permite construir estructuras de datos distintas de las secuencias y árboles con los que trabajamos en la sección  [2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2) . También presentamos algunos ejemplos de simulaciones en las que se modelan sistemas complejos como colecciones de objetos con estado local.

**[3.3.1 Estructura de lista mutable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.3.1)**

Las operaciones básicas sobre pares - cons , car y cdr - se pueden utilizar para construir estructuras de listas y seleccionar partes de ellas, pero no son capaces de modificarlas. Lo mismo se aplica a las operaciones de listas que hemos utilizado hasta ahora, como append y list , ya que se pueden definir en términos de cons , car y cdr . Para modificar estructuras de listas necesitamos nuevas operaciones.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.12:**   Listas x : ((ab) cd) e y : (ef) . |

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.13:**   Efecto de (set-car! xy) en las listas de la figura  [3.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.12) . |

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.14:**   Efecto de (define z (cons y (cdr x))) en las listas de la figura  [3.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.12) . |

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.15:**   Efecto de (set-cdr! xy) en las listas de la figura  [3.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.12) . |

Los mutadores primitivos para pares son set-car! y set-cdr!. Set -car! toma dos argumentos, el primero de los cuales debe ser un par. Modifica este par, reemplazando el puntero car por un puntero al segundo argumento de set-car !. [16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_349)

Como ejemplo, supongamos que x está ligada a la lista ((ab) cd) e y a la lista (ef) como se ilustra en la figura  [3.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.12) . Al evaluar la expresión (set-car! xy) se modifica el par al que está ligada x , reemplazando su car por el valor de y . El resultado de la operación se muestra en la figura  [3.13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.13) . La estructura x ha sido modificada y ahora se imprimiría como ((e f) c d) . Los pares que representan la lista (ab) , identificados por el puntero que fue reemplazado, ahora están separados de la estructura original. [17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_350)

Compare la figura  [3.13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.13) con la figura  [3.14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.14) , que ilustra el resultado de ejecutar (define z (cons y (cdr x))) con x e y enlazadas a las listas originales de la figura  [3.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.12) . La variable z ahora está enlazada a un nuevo par creado por la operación cons ; la lista a la que está enlazada x no cambia.

La operación set-cdr! es similar a set-car! . La única diferencia es que se reemplaza el puntero cdr del par, en lugar del puntero car . El efecto de ejecutar (set-cdr! xy) en las listas de la figura  [3.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.12) se muestra en la figura  [3.15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.15) . Aquí el puntero cdr de x ha sido reemplazado por el puntero a (ef) . Además, la lista (cd) , que solía ser el cdr de x , ahora está separada de la estructura.

Cons construye una nueva estructura de lista creando nuevos pares, mientras que set-car! y set-cdr! modifican los pares existentes. De hecho, podríamos implementar cons en términos de los dos mutadores, junto con un procedimiento get-new-pair , que devuelve un nuevo par que no es parte de ninguna estructura de lista existente. Obtenemos el nuevo par, establecemos sus punteros car y cdr a los objetos designados y devolvemos el nuevo par como resultado de cons . [18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_351)

(define (cons x y)   
  (let ((new (get-new-pair)))   
    (set-car! new x)   
    (set-cdr! new y)   
    new))

**Ejercicio 3.12.**  El siguiente procedimiento para añadir listas se introdujo en la sección  [2.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.1) :

(define (añade x y)   
  (si (nulo? x)   
      y   
      (cons (car x) (añade (cdr x) y))))

Append forma una nueva lista al unir sucesivamente los elementos de x con y . El procedimiento append! es similar a append , pero es un mutador en lugar de un constructor. Agrega las listas empalmándolas, modificando el par final de x de modo que su cdr sea ahora y . (Es un error llamar a append! con una  x vacía ).

(define (append! x y)   
  (set-cdr! (last-pair x) y)   
  x)

Aquí last-pair es un procedimiento que devuelve el último par en su argumento:

(define (último par x)   
  (si (nulo? (cdr x))   
      x   
      (último par (cdr x))))

Considere la interacción

(define x (lista 'a' b))   
(define y (lista 'c' d))   
(define z (añade x y))   
z   
*(a b c d)*  
(cdr x)   
< *respuesta* >   
(define w (añade! x y))   
w   
*(a b c d)*  
(cdr x)   
< *respuesta* >

¿Cuáles son las *respuestas* que faltan ? Dibuje diagramas de caja y puntero para explicar su respuesta.

**Ejercicio 3.13.**  Considere el siguiente procedimiento de ciclo de creación , que utiliza el procedimiento de último par definido en el ejercicio  [3.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.12) :

(define (ciclo-de-hacer x)   
  (establece-cdr! (último-par x) x)   
  x)

Dibuje un diagrama de caja y puntero que muestre la estructura z creada por

(define z (make-cycle (lista 'a 'b 'c)))

¿Qué sucede si intentamos calcular (último par z) ?

**Ejercicio 3.14.**   El siguiente procedimiento es bastante útil, aunque poco claro:

(define (misterio x)   
  (define (bucle x y)   
    (si (nulo? x)   
        y   
        (let ((temp (cdr x)))   
          (establecer-cdr! x y)   
          (bucle temp x))))   
  (bucle x '()))

El bucle utiliza la variable ``temporal'' temp para guardar el valor anterior del cdr de x , ya que el set-cdr! en la siguiente línea destruye el cdr . Explique lo que hace mystery en general. Suponga que v está definido por (define v (list 'a 'b 'c 'd)) . Dibuje el diagrama de caja y puntero que representa la lista a la que está enlazado v . Suponga que ahora evaluamos (define w (mystery v)) . Dibuje diagramas de caja y puntero que muestren las estructuras v y w después de evaluar esta expresión. ¿Qué se imprimirían como los valores de v y w ?

**[Compartir e identidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_355)**

[En la sección 3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_3.1.3) mencionamos  los problemas teóricos de "igualdad" y "cambio" que surgen con la introducción de la asignación. Estos problemas surgen en la práctica cuando se *comparten* pares individuales entre diferentes objetos de datos. Por ejemplo, considere la estructura formada por

(define x (lista 'a 'b))   
(define z1 (cons x x))

Como se muestra en la figura  [3.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.16) , z1 es un par cuyos car y cdr apuntan al mismo par x . Esta compartición de x por parte de car y cdr de z1 es una consecuencia de la forma sencilla en que se implementa cons . En general, el uso de cons para construir listas dará como resultado una estructura interconectada de pares en la que muchos pares individuales son compartidos por muchas estructuras diferentes.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.16:**   La lista z1 formada por (cons xx) . |

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.17:**   La lista z2 formada por (cons (list 'a 'b) (list 'a 'b)) . |

A diferencia de la figura  [3.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.16) , la figura  [3.17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.17) muestra la estructura creada por

(define z2(cons(lista 'a'b) (lista 'a'b)))

En esta estructura, los pares en las dos listas (ab) son distintos, aunque los símbolos reales son compartidos. [19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_356)

Cuando se piensa en una lista, z1 y z2 representan "la misma" lista, ((ab) ab) . En general, la compartición es completamente indetectable si operamos en listas utilizando solo cons , car y cdr . Sin embargo, si permitimos mutadores en la estructura de la lista, la compartición se vuelve significativa. Como ejemplo de la diferencia que puede generar la compartición, considere el siguiente procedimiento, que modifica el car de la estructura a la que se aplica:

(define (establece-a-wow! x)   
  (establece-auto! (auto x) 'wow)   
  x)

Aunque z1 y z2 son ``la misma'' estructura, aplicarles set-to-wow! produce resultados diferentes. Con z1 , al modificar el auto también se modifica el cdr , porque en z1 el auto y el cdr son el mismo par. Con z2 , el auto y el cdr son distintos, por lo que set-to-wow! modifica solo el auto :

z1   
*((a b) a b)*  
  
(establecido en ¡guau! z1)   
*((guau b) guau b)*  
  
z2   
*((a b) a b)*  
  
(establecido en ¡guau! z2)   
*((guau b) a b)*

Una forma de detectar la compartición en estructuras de listas es utilizar el predicadoeq? , que introdujimos en la sección  [2.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.1) como una forma de probar si dos símbolos son iguales. De manera más general, (eq? xy) prueba si x e y son el mismo objeto (es decir, si x e y son iguales como punteros). Por lo tanto, con z1 y z2 como se definen en las figuras  [3.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.16) y  [3.17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.17) , (eq? (car z1) (cdr z1)) es verdadera y (eq? (car z2) (cdr z2)) es falsa.

Como se verá en las siguientes secciones, podemos aprovechar la compartición para ampliar en gran medida el repertorio de estructuras de datos que se pueden representar por pares. Por otro lado, la compartición también puede ser peligrosa, ya que las modificaciones realizadas a las estructuras también afectarán a otras estructuras que compartan las partes modificadas. Las operaciones de mutación set-car! y set-cdr! deben utilizarse con cuidado; a menos que tengamos una buena comprensión de cómo se comparten nuestros objetos de datos, la mutación puede tener resultados imprevistos. [20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_357)

**Ejercicio 3.15.**   Dibuje diagramas de caja y puntero para explicar el efecto de la función set-to-wow! en las estructuras z1 y z2 anteriores.

**Ejercicio 3.16.**   Ben Bitdiddle decide escribir un procedimiento para contar el número de pares en cualquier estructura de lista. "Es fácil", razona. "El número de pares en cualquier estructura es el número en el car más el número en el cdr más uno más para contar el par actual". Entonces Ben escribe el siguiente procedimiento:

(define (count-pares x)   
  (si (no (pair? x))   
      0   
      (+ (count-pares (car x))   
         (count-pares (cdr x))   
         1)))

Demuestre que este procedimiento no es correcto. En particular, dibuje diagramas de caja y puntero que representen estructuras de lista formadas por exactamente tres pares para los cuales el procedimiento de Ben devolvería 3; devolvería 4; devolvería 7; nunca devolvería nada.

**Ejercicio 3.17.**   Diseñe una versión correcta del procedimiento de conteo de pares del ejercicio  [3.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.16) que devuelva el número de pares distintos en cualquier estructura. (Sugerencia: recorra la estructura, manteniendo una estructura de datos auxiliar que se utiliza para llevar un registro de los pares que ya se han contado).

**Ejercicio 3.18.**  Escriba un procedimiento que examine una lista y determine si contiene un ciclo, es decir, si un programa que intentara encontrar el final de la lista tomando cdr sucesivos entraría en un bucle infinito. Ejercicio  [3.13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.13) construyó tales listas.

**Ejercicio 3.19.**   Repita el ejercicio  [3.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.18) utilizando un algoritmo que ocupe solo una cantidad constante de espacio. (Esto requiere una idea muy ingeniosa).

**[La mutación es solo una asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_363)**

Cuando introdujimos datos compuestos, observamos en la sección  [2.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.3) que los pares pueden representarse puramente en términos de procedimientos:

(define (cons x y)   
  (define (despacha m)   
    (cond ((eq? m 'car) x)   
          ((eq? m 'cdr) y)   
          (else (error "Operación indefinida -- CONS" m))))   
  despacha)  
(define (coche z) (z 'coche))  
(define (cdr z) (z 'cdr))

La misma observación es válida para los datos mutables. Podemos implementar objetos de datos mutables como procedimientos utilizando la asignación y el estado local. Por ejemplo, podemos extender la implementación del par anterior para manejar set-car! y set-cdr! de una manera análoga a la forma en que implementamos cuentas bancarias utilizando make-account en la sección  [3.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.1) :

(define (cons x y)   
  (define (set-x! v) (set! x v))   
  (define (set-y! v) (set! y v))   
  (define (dispatch m)   
    (cond ((eq? m 'car) x)   
          ((eq? m 'cdr) y)   
          ((eq? m 'set-car!) set-x!)   
          ((eq? m 'set-cdr!) set-y!)   
          (else (error "Operación indefinida -- CONS" m))))   
  dispatch)  
(define (coche z) (z 'coche))  
(define (cdr z) (z 'cdr))  
(define (set-car! z nuevo-valor)   
  ((z 'set-car!) nuevo-valor)   
  z)  
(define (set-cdr! z nuevo-valor)   
  ((z 'set-cdr!) nuevo-valor)   
  z)

La asignación es todo lo que se necesita, teóricamente, para explicar el comportamiento de los datos mutables. Tan pronto como admitimos set! en nuestro lenguaje, planteamos todas las cuestiones, no sólo de la asignación, sino de los datos mutables en general. [21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_364)

**Ejercicio 3.20.**   Dibuje diagramas de entorno para ilustrar la evaluación de la secuencia de expresiones.

(define x (cons 1 2))   
(define z (cons x x))   
(conjunto-auto! (cdr z) 17)   
(auto x)   
*17*

utilizando la implementación procedimental de pares dada anteriormente. (Compare el ejercicio  [3.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.11) ).

**[3.3.2 Representación de colas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.3.2)**

Los mutadores set-car! y set-cdr! nos permiten usar pares para construir estructuras de datos que no se pueden construir con cons , car y cdr solos. Esta sección muestra cómo usar pares para representar una estructura de datos llamada cola. La sección  [3.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.3) mostrará cómo representar estructuras de datos llamadas tablas.

Una *cola* es una secuencia en la que los elementos se insertan en un extremo (llamado*parte trasera* de la cola) y se eliminan del otro extremo (elLa figura  [3.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.18) muestra una cola inicialmente vacía en la que se insertan los elementos a y b . Luego se elimina a , se insertan c y d *, y se elimina* b . Debido a que los elementos siempre se eliminan en el orden en el que se insertan, a una cola a veces se la denomina*Búfer FIFO* (primero en entrar, primero en salir).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Operación | Cola resultante | | (define q (hacer cola)) |  | | (insertar-cola! q 'a) | a | | (insertar-cola! q 'b) | desde | | (¡borrar cola! q) | b | | (insertar-cola! q 'c) | antes de Cristo | | (insertar-cola! q 'd) | bcd | | (¡borrar cola! q) | cd | |
|  |
| **Figura 3.18:**   Operaciones de cola. |

En términos de abstracción de datos, podemos considerar una cola como definida por el siguiente conjunto de operaciones:

* un constructor:  
  (make-queue)  
  devuelve una cola vacía (una cola que no contiene elementos).
* dos selectores:  
  (cola-vacía? < *cola* >)  
  prueba si la cola está vacía.  
  (front-queue < *queue* >)  
  devuelve el objeto al principio de la cola, señalando un error si la cola está vacía; no modifica la cola.
* dos mutadores:  
  (insert-queue! < *cola* > < *elemento* >)  
  inserta el elemento al final de la cola y devuelve la cola modificada como su valor.  
  (delete-queue! < *queue* >)  
  elimina el elemento al principio de la cola y devuelve la cola modificada como su valor, señalando un error si la cola está vacía antes de la eliminación.

Como una cola es una secuencia de elementos, ciertamente podríamos representarla como una lista ordinaria; el frente de la cola sería el carro de la lista, insertar un elemento en la cola equivaldría a agregar un nuevo elemento al final de la lista, y eliminar un elemento de la cola sería simplemente tomar el cdr de la lista. Sin embargo, esta representación es ineficiente, porque para insertar un elemento debemos escanear la lista hasta que lleguemos al final. Como el único método que tenemos para escanear una lista es mediante operaciones cdr sucesivas , este escaneo requiere ( *n* ) pasos para una lista de *n* elementos. Una simple modificación a la representación de la lista supera esta desventaja al permitir que las operaciones de la cola se implementen de modo que requieran (1) pasos; es decir, de modo que el número de pasos necesarios sea independiente de la longitud de la cola.

La dificultad de la representación en forma de lista surge de la necesidad de escanear para encontrar el final de la lista. La razón por la que necesitamos escanear es que, aunque la forma estándar de representar una lista como una cadena de pares nos proporciona fácilmente un puntero al principio de la lista, no nos da ningún puntero fácilmente accesible al final. La modificación que evita el inconveniente es representar la cola como una lista, junto con un puntero adicional que indica el par final de la lista. De esa manera, cuando vamos a insertar un elemento, podemos consultar el puntero posterior y así evitar escanear la lista.

Una cola se representa, entonces, como un par de punteros, front-ptr y rear-ptr , que indican, respectivamente, el primer y el último par de una lista ordinaria. Como nos gustaría que la cola fuera un objeto identificable, podemos utilizar cons para combinar los dos punteros. De este modo, la cola en sí misma será el cons de los dos punteros. La figura  [3.19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.19) ilustra esta representación.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.19:**   Implementación de una cola como una lista con punteros frontales y posteriores. |

Para definir las operaciones de cola utilizamos los siguientes procedimientos, que nos permiten seleccionar y modificar los punteros frontal y posterior de una cola:

(define (cola front-ptr) (cola de autos))  
(define (cola rear-ptr) (cola cdr))  
(define (set-front-ptr! elemento de la cola) (set-car! elemento de la cola))  
(define (set-rear-ptr! elemento de cola) (set-cdr! elemento de cola))

Ahora podemos implementar las operaciones de cola reales. Consideraremos que una cola está vacía si su puntero frontal es la lista vacía:

(define (cola-vacía? cola) (nulo? (cola-ptr-frontal)))

El constructor make-queue devuelve, como una cola inicialmente vacía, un par cuyo car y cdr son ambos la lista vacía:

(define (make-queue) (cons '() '()))

Para seleccionar el elemento que se encuentra al frente de la cola, devolvemos el coche del par indicado por el puntero delantero:

(define (cola-frontal)   
  (si (¿cola-vacía?)   
      (error "FRONT llamado con una cola vacía" cola)   
      (car (cola-ptr-frontal))))

Para insertar un elemento en una cola, seguimos el método cuyo resultado se indica en la figura  [3.20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.20) . Primero creamos un nuevo par cuyo car es el elemento a insertar y cuyo cdr es la lista vacía. Si la cola estaba vacía inicialmente, fijamos los punteros frontal y posterior de la cola a este nuevo par. En caso contrario, modificamos el par final de la cola para que apunte al nuevo par, y también fijamos el puntero posterior al nuevo par.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.20:**   Resultado del uso de (insert-queue! q 'd) en la cola de la figura  [3.19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.19) . |

(define (insert-queue! elemento de cola)   
  (let ((new-pair (cons item '())))   
    (cond ((empty-queue? cola)   
           (set-front-ptr! cola nuevo-par)   
           (set-rear-ptr! cola nuevo-par)   
           cola)   
          (else   
           (set-cdr! (rear-ptr cola) nuevo-par)   
           (set-rear-ptr! cola nuevo-par)   
           cola))))

Para eliminar el elemento que se encuentra al principio de la cola, simplemente modificamos el puntero frontal para que ahora apunte al segundo elemento de la cola, que se puede encontrar siguiendo el puntero cdr del primer elemento (ver figura  [3.21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.21) ): [22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_366)

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.21:**   Resultado del uso de (delete-queue! q) en la cola de la figura  [3.20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.20) . |

(define (delete-queue! queue)   
  (cond ((empty-queue? queue)   
         (error "DELETE! llamado con una cola vacía" queue))   
        (else   
         (set-front-ptr! queue (cdr (front-ptr queue)))   
         queue)))

**Ejercicio 3.21.**   Ben Bitdiddle decide probar la implementación de la cola descrita anteriormente. Escribe los procedimientos en el intérprete de Lisp y procede a probarlos:

(define q1 (make-queue))   
(insert-queue! q1 'a)   
*((a) a)*  
(insert-queue! q1 'b)   
*((a b) b)*  
(delete-queue! q1)   
*((b) b)*  
(delete-queue! q1)   
*(() b)*

``¡Está todo mal!'', se queja. ``La respuesta del intérprete muestra que el último elemento se inserta en la cola dos veces. Y cuando borro ambos elementos, el segundo b sigue ahí, por lo que la cola no está vacía, aunque se supone que debería estarlo.'' Eva Lu Ator sugiere que Ben ha entendido mal lo que está sucediendo. ``No es que los elementos estén entrando en la cola dos veces'', explica. ``Es sólo que la impresora estándar de Lisp no sabe cómo interpretar la representación de la cola. Si quieres ver la cola impresa correctamente, tendrás que definir tu propio procedimiento de impresión para colas.'' Explique de qué está hablando Eva Lu. En particular, muestre por qué los ejemplos de Ben producen los resultados impresos que producen. Defina un procedimientocola de impresión que toma una cola como entrada e imprime la secuencia de elementos en la cola.

**Ejercicio 3.22.**  En lugar de representar una cola como un par de punteros, podemos construir una cola como un procedimiento con estado local. El estado local estará compuesto por punteros al principio y al final de una lista ordinaria. Por lo tanto, el procedimiento de creación de cola tendrá la forma

(define (make-queue)   
  (let ((front-ptr  ... )   
        (rear-ptr  ... ))   
    < *definiciones de procedimientos internos* >   
    (define (dispatch m)  ... )   
    dispatch))

Complete la definición de make-queue y proporcione implementaciones de las operaciones de cola utilizando esta representación.

**Ejercicio 3.23.**  Una *deque* («cola de doble extremo») es una secuencia en la que se pueden insertar y eliminar elementos tanto en la parte delantera como en la trasera. Las operaciones sobre deques son el constructor make-deque , el predicado empty-deque? , los selectores front-deque y rear-deque , y los mutadores front-insert-deque! , rear-insert-deque! , front-delete-deque! y rear-delete-deque! . Muestre cómo representar deques usando pares y proporcione implementaciones de las operaciones. [23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_370) Todas las operaciones deben realizarse en (1) pasos.

**[3.3.3 Representación de tablas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.3.3)**

Cuando estudiamos varias formas de representar conjuntos en el capítulo 2, mencionamos en la sección  [2.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.3) la tarea de mantener una tabla de registros indexados mediante claves de identificación. En la implementación de la programación dirigida por datos en la sección  [2.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.3) , hicimos un uso extensivo de tablas bidimensionales, en las que la información se almacena y recupera utilizando dos claves. Aquí vemos cómo construir tablas como estructuras de lista mutables.

En primer lugar, consideramos una tabla unidimensional, en la que cada valor se almacena bajo una clave única. Implementamos la tabla como una lista de registros, cada uno de los cuales se implementa como un par que consta de una clave y el valor asociado. Los registros se unen para formar una lista por pares cuyos autos apuntan a registros sucesivos. Estos pares de unión se denominan*columna vertebral* de la tabla. Para tener un lugar que podamos cambiar cuando agregamos un nuevo registro a la tabla, construimos la tabla como una*lista encabezada* . Una lista encabezada tiene un par de columnas principales especial al principio, que contiene un ``registro'' ficticio, en este caso el símbolo \*table\* elegido arbitrariamente . La figura  [3.22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.22) muestra el diagrama de caja y puntero para la tabla.

a: 1   
b: 2   
c: 3

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.22:**   Una tabla representada como una lista encabezada. |

Para extraer información de una tabla, utilizamos el procedimiento lookup , que toma una clave como argumento y devuelve el valor asociado (o false si no hay ningún valor almacenado bajo esa clave). Lookup se define en términos de la operación assoc , que espera una clave y una lista de registros como argumentos. Tenga en cuenta que assoc nunca ve el registro ficticio. Assoc devuelve el registro que tiene la clave dada como su car . [24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_371) Lookup luego verifica que el registro resultante devuelto por assoc no sea false y devuelve el valor (el cdr ) del registro.

(define (tabla de claves de búsqueda)   
  (let ((registro (clave de asociación (tabla cdr))))   
    (si registro   
        (registro cdr)   
        falso)))  
(define (clave de asociación registros)   
  (cond ((¿nulo? registros) falso)   
        ((¿igual? clave (registros caar)) (registros car))   
        (else (clave de asociación (registros cdr)))))

Para insertar un valor en una tabla bajo una clave específica, primero usamos assoc para ver si ya existe un registro en la tabla con esta clave. Si no, formamos un nuevo registro al insertar la clave con el valor y lo insertamos al principio de la lista de registros de la tabla, después del registro ficticio. Si ya existe un registro con esta clave, establecemos el cdr de este registro en el nuevo valor designado. El encabezado de la tabla nos proporciona una ubicación fija para modificar con el fin de insertar el nuevo registro. [25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_372)

(define (insert! clave valor tabla)   
  (let ((registro (clave asociada (cdr tabla))))   
    (if registro   
        (set-cdr! registro valor)   
        (set-cdr! tabla   
                  (cons (cons clave valor) (cdr tabla)))))   
  'ok)

Para construir una nueva tabla, simplemente creamos una lista que contenga el símbolo \*tabla\* :

(define (crear-tabla)   
  (lista '\*tabla\*))

**[Tablas bidimensionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_373)**

En una tabla bidimensional, cada valor está indexado por dos claves. Podemos construir una tabla de este tipo como una tabla unidimensional en la que cada clave identifica una subtabla. La figura  [3.23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.23) muestra el diagrama de caja y puntero para la tabla.

Matemáticas:   
    +: 43   
    -: 45   
    \*: 42   
Letras:   
    a: 97   
    b: 98

que tiene dos subtablas. (Las subtablas no necesitan un símbolo de encabezado especial, ya que la clave que identifica la subtabla cumple este propósito).

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.23:**   Una tabla bidimensional. |

Cuando buscamos un elemento, utilizamos la primera clave para identificar la subtabla correcta. Luego, utilizamos la segunda clave para identificar el registro dentro de la subtabla.

(define (búsqueda clave-1 clave-2 tabla)   
  (let ((subtabla (clave asociada-1 (cdr tabla))))   
    (if subtabla   
        (let ((registro (clave asociada-2 (cdr subtabla))))   
          (if registro   
              (cdr registro)   
              falso))   
        falso)))

Para insertar un nuevo elemento bajo un par de claves, utilizamos assoc para ver si hay una subtabla almacenada bajo la primera clave. Si no, construimos una nueva subtabla que contenga el registro único ( key-2 , value ) y lo insertamos en la tabla bajo la primera clave. Si ya existe una subtabla para la primera clave, insertamos el nuevo registro en esta subtabla, utilizando el método de inserción para tablas unidimensionales descrito anteriormente:

(define (insert! key-1 key-2 value table)   
  (let ((subtable (assoc key-1 (cdr table))))   
    (if subtable   
        (let ((record (assoc key-2 (cdr subtable))))   
          (if record   
              (set-cdr! record value)   
              (set-cdr! subtable   
                        (cons (cons key-2 value)   
                              (cdr subtable)))))   
        (set-cdr! table   
                  (cons (list key-1   
                              (cons key-2 value))   
                        (cdr table)))))   
  'ok)

**[Creando tablas locales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_374)**

Las operaciones de búsqueda e inserción definidas anteriormente toman la tabla como argumento. Esto nos permite utilizar programas que acceden a más de una tabla. Otra forma de trabajar con varias tablas es tener procedimientos de búsqueda e inserción independientes para cada tabla. Podemos hacerlo representando una tabla de manera procedimental, como un objeto que mantiene una tabla interna como parte de su estado local. Cuando se envía un mensaje apropiado, este "objeto de tabla" proporciona el procedimiento con el que operar en la tabla interna. A continuación se muestra un generador de tablas bidimensionales representado de esta manera:

(define (crear-tabla)   
  (let ((tabla-local (lista '\*tabla\*)))   
    (define (búsqueda clave-1 clave-2)   
      (let ((subtabla (asociación clave-1 (cdr tabla-local))))   
        (if subtabla   
            (let ((registro (asociación clave-2 (cdr subtabla))))   
              (if registro   
                  (cdr registro)   
                  false))   
            false)))   
    (define (insertar! clave-1 clave-2 valor)   
      (let ((subtabla (asociación clave-1 (cdr tabla-local))))   
        (if subtabla   
            (let ((registro (asociación clave-2 (cdr subtabla))))   
              (if registro   
                  (establecer-cdr! registro valor)   
                  (establecer-cdr! subtabla   
                            (cons (cons clave-2 valor)   
                                  (cdr subtabla)))))   
            (establecer-cdr! tabla-local   
                      (cons (lista clave-1   
                                  (cons clave-2 valor))   
                            (cdr tabla-local)))))   
      'ok)       
    (define (enviar m)   
      (cond ((eq? m 'lookup-proc) búsqueda)   
            ((eq? m 'insertar-proc!) insertar!)   
            (else (error "Operación desconocida -- TABLA" m))))   
    enviar))

Usando make-table , podríamos implementar las operaciones get y put utilizadas en la sección  [2.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.3) para programación dirigida por datos, de la siguiente manera:

(definir tabla-de-operaciones (tabla-de-creación))  
(define obtener (tabla-de-operaciones 'proc-buscacion))  
(define put (tabla-de-operaciones 'proc-insertar!))

Get toma como argumentos dos claves y put toma como argumentos dos claves y un valor. Ambas operaciones acceden a la misma tabla local, que está encapsulada dentro del objeto creado por la llamada a make-table .

**Ejercicio 3.24.**  En las implementaciones de tabla anteriores, las claves se prueban para comprobar su igualdad utilizando equal? ​​(llamado por assoc ). Esta no siempre es la prueba adecuada. Por ejemplo, podríamos tener una tabla con claves numéricas en la que no necesitamos una coincidencia exacta con el número que estamos buscando, sino solo un número dentro de cierta tolerancia de este. Diseñe un constructor de tabla make-table que tome como argumento un procedimiento same-key? que se utilizará para probar la ``igualdad'' de claves. Make-table debería devolver un procedimiento de envío que se pueda utilizar para acceder a los procedimientos de búsqueda e inserción adecuados para una tabla local.

**Ejercicio 3.25.**  Generalizando tablas unidimensionales y bidimensionales, se muestra cómo implementar una tabla en la que los valores se almacenan bajo un número arbitrario de claves y se pueden almacenar diferentes valores bajo diferentes números de claves. Los procedimientos lookup e insert! deben tomar como entrada una lista de claves utilizadas para acceder a la tabla.

**Ejercicio 3.26.**  Para buscar en una tabla como la implementada anteriormente, es necesario recorrer la lista de registros. Básicamente, se trata de la representación de lista desordenada de la sección  [2.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.3) . Para tablas grandes, puede ser más eficiente estructurar la tabla de una manera diferente. Describa una implementación de tabla donde los registros (clave, valor) estén organizados utilizando un árbol binario, suponiendo que las claves se pueden ordenar de alguna manera (por ejemplo, numérica o alfabéticamente). (Compare el ejercicio  [2.66](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.66) del capítulo 2).

**Ejercicio 3.27.**  *La memorización* (también llamada *tabulación* ) es una técnica que permite a un procedimiento registrar, en una tabla local, valores que se han calculado previamente. Esta técnica puede marcar una gran diferencia en el rendimiento de un programa. Un procedimiento memorizado mantiene una tabla en la que se almacenan los valores de llamadas anteriores utilizando como claves los argumentos que produjeron los valores. Cuando se le pide al procedimiento memorizado que calcule un valor, primero verifica la tabla para ver si el valor ya está allí y, si es así, simplemente devuelve ese valor. De lo contrario, calcula el nuevo valor de la forma habitual y lo almacena en la tabla. Como ejemplo de memorización, recuerde de la sección  [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) el proceso exponencial para calcular los números de Fibonacci:

(define (fib n)   
  (cond ((= n 0) 0)   
        ((= n 1) 1)   
        (de lo contrario (+ (fib (- n 1))   
                 (fib (- n 2))))))

La versión memorizada del mismo procedimiento es

(define memo-fib   
  (memoizar (lambda (n)   
             (cond ((= n 0) 0)   
                   ((= n 1) 1)   
                   (de lo contrario (+ (memo-fib (- n 1))   
                            (memo-fib (- n 2))))))))

donde el memorizador se define como

(define (memoize f)   
  (let ((tabla (crear-tabla)))   
    (lambda (x)   
      (let ((resultado-calculado-previamente (buscar-tabla-x)))   
        (o resultado-calculado-previamente   
            (let ((resultado (f x)))   
              (insertar! x resultado tabla)   
              resultado))))))

Dibuje un diagrama de entorno para analizar el cálculo de (memo-fib 3) . Explique por qué memo-fib calcula el *n-* ésimo número de Fibonacci en una cantidad de pasos proporcional a *n* . ¿El esquema funcionaría si simplemente hubiéramos definido memo-fib como (memoize fib) ?

**[3.3.4 Un simulador para circuitos digitales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.3.4)**

El diseño de sistemas digitales complejos, como las computadoras, es una actividad de ingeniería importante. Los sistemas digitales se construyen interconectando elementos simples. Aunque el comportamiento de estos elementos individuales es simple, las redes de ellos pueden tener un comportamiento muy complejo. La simulación por computadora de los diseños de circuitos propuestos es una herramienta importante utilizada por los ingenieros de sistemas digitales. En esta sección, diseñamos un sistema para realizar simulaciones lógicas digitales. Este sistema es un ejemplo típico de un tipo de programa llamado*simulación basada en eventos* , en la que las acciones («eventos») desencadenan otros eventos que ocurren en un momento posterior, que a su vez desencadenan más eventos, y así sucesivamente.

Nuestro modelo computacional de un circuito estará compuesto por objetos que corresponden a los componentes elementales a partir de los cuales se construye el circuito. Existen*cables* que transportan*Señales digitales* . Una señal digital puede tener en cualquier momento solo uno de dos valores posibles, 0 y 1. También existen varios tipos de señales digitales.*Cajas de función* , que conectan cables que llevan señales de entrada a otros cables de salida. Estas cajas producen señales de salida calculadas a partir de sus señales de entrada. La señal de salida esretrasado por un tiempo que depende del tipo de cuadro de función. Por ejemplo, un*El inversor* es una caja de función primitiva que invierte su entrada. Si la señal de entrada a un inversor cambia a 0, entonces, un retardo de inversor más tarde, el inversor cambiará su señal de salida a 1. Si la señal de entrada a un inversor cambia a 1, entonces, un retardo de inversor más tarde, el inversor cambiará su señal de salida a 0. Dibujamos un inversor simbólicamente como en la figura  [3.24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.24) .*La compuerta and* , que también se muestra en la figura  [3.24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.24) , es una caja de función primitiva con dos entradas y una salida. Lleva su señal de salida a un valor que es el*lógica y* de las entradas. Es decir, si ambas señales de entrada se convierten en 1, entonces un tiempo de retardo de compuerta and después, la compuerta and forzará que su señal de salida sea 1; de lo contrario, la salida será 0.*or-gate* es una caja de función primitiva de dos entradas similar que lleva su señal de salida a un valor que es el*lógica o* de las entradas. Es decir, la salida será 1 si al menos una de las señales de entrada es 1; en caso contrario, la salida será 0.

|  |
| --- |
|  |
|  |
| **Figura 3.24:**   Funciones primitivas en el simulador lógico digital. |

Podemos conectar funciones primitivas para construir funciones más complejas. Para lograrlo, conectamos las salidas de algunas cajas de función a las entradas de otras cajas de función. Por ejemplo,*El circuito semisumador* que se muestra en la figura  [3.25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.25) consta de una compuerta O, dos compuertas Y y un inversor. Toma dos señales de entrada, A y B, y tiene dos señales de salida, S y C. S se convertirá en 1 siempre que precisamente uno de A y B sea 1, y C se convertirá en 1 siempre que A y B sean ambos 1. Podemos ver en la figura que, debido a los retrasos involucrados, las salidas pueden generarse en diferentes momentos. Muchas de las dificultades en el diseño de circuitos digitales surgen de este hecho.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.25:**   Un circuito semisumador. |

Ahora crearemos un programa para modelar los circuitos lógicos digitales que deseamos estudiar. El programa construirá objetos computacionales que modelen los cables, que "contendrán" las señales. Los cuadros de función se modelarán mediante procedimientos que impongan las relaciones correctas entre las señales.

Un elemento básico de nuestra simulación será un procedimiento llamado make-wire , que construye cables. Por ejemplo, podemos construir seis cables de la siguiente manera:

(define a (hacer-alambre))   
(define b (hacer-alambre))   
(define c (hacer-alambre))   
  
(define d (hacer-alambre))   
(define e (hacer-alambre))   
(define s (hacer-alambre))

Conectamos una caja de función a un conjunto de cables llamando a un procedimiento que construye ese tipo de caja. Los argumentos del procedimiento constructor son los cables que se conectarán a la caja. Por ejemplo, dado que podemos construir puertas and, puertas or e inversores, podemos conectar entre sí el semisumador que se muestra en la figura  [3.25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.25) :

(puerta-o a b d)   
*ok*  
  
(puerta-y a b c)   
*ok*  
  
(inversor c e)   
*ok*  
  
(puerta-y d e s)   
*ok*

Better yet, we can explicitly name this operation by defining a procedure half-adder that constructs this circuit, given the four external wires to be attached to the half-adder:

(define (half-adder a b s c)  
  (let ((d (make-wire)) (e (make-wire)))  
    (or-gate a b d)  
    (and-gate a b c)  
    (inverter c e)  
    (and-gate d e s)  
    'ok))

The advantage of making this definition is that we can use half-adder itself as a building block in creating more complex circuits. Figure [3.26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.26), for example, shows a *full-adder* composed of two half-adders and an or-gate.[26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_379) We can construct a full-adder as follows:

(define (full-adder a b c-in sum c-out)  
  (let ((s (make-wire))  
        (c1 (make-wire))  
        (c2 (make-wire)))  
    (half-adder b c-in s c1)  
    (half-adder a s sum c2)  
    (or-gate c1 c2 c-out)  
    'ok))

Having defined full-adder as a procedure, we can now use it as a building block for creating still more complex circuits. (For example, see exercise [3.30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.30).)

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figure 3.26:**  A full-adder circuit. |

In essence, our simulator provides us with the tools to construct a language of circuits. If we adopt the general perspective on languages with which we approached the study of Lisp in section [1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1), we can say that the primitive function boxes form the primitive elements of the language, that wiring boxes together provides a means of combination, and that specifying wiring patterns as procedures serves as a means of abstraction.

**[Primitive function boxes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_380)**

The primitive function boxes implement the ``forces'' by which a change in the signal on one wire influences the signals on other wires. To build function boxes, we use the following operations on wires:

* (get-signal <*wire*>)  
  returns the current value of the signal on the wire.
* (set-signal! <*wire*> <*new value*>)  
  changes the value of the signal on the wire to the new value.
* (add-action! <*wire*> <*procedure of no arguments*>)  
  asserts that the designated procedure should be run whenever the signal on the wire changes value. Such procedures are the vehicles by which changes in the signal value on the wire are communicated to other wires.

In addition, we will make use of a procedure after-delay that takes a time delay and a procedure to be run and executes the given procedure after the given delay.

Using these procedures, we can define the primitive digital logic functions. To connect an input to an output through an inverter, we use add-action! to associate with the input wire a procedure that will be run whenever the signal on the input wire changes value. The procedure computes the logical-not of the input signal, and then, after one inverter-delay, sets the output signal to be this new value:

(define (inverter input output)  
  (define (invert-input)  
    (let ((new-value (logical-not (get-signal input))))  
      (after-delay inverter-delay  
                   (lambda ()  
                     (set-signal! output new-value)))))  
  (add-action! input invert-input)  
  'ok)  
(define (logical-not s)  
  (cond ((= s 0) 1)  
        ((= s 1) 0)  
        (else (error "Invalid signal" s))))

An and-gate is a little more complex. The action procedure must be run if either of the inputs to the gate changes. It computes the logical-and (using a procedure analogous to logical-not) of the values of the signals on the input wires and sets up a change to the new value to occur on the output wire after one and-gate-delay.

(define (and-gate a1 a2 output)  
  (define (and-action-procedure)  
    (let ((new-value  
           (logical-and (get-signal a1) (get-signal a2))))  
      (after-delay and-gate-delay  
                   (lambda ()  
                     (set-signal! output new-value)))))  
  (add-action! a1 and-action-procedure)  
  (add-action! a2 and-action-procedure)  
  'ok)

**Exercise 3.28.**  Define an or-gate as a primitive function box. Your or-gate constructor should be similar to and-gate.

**Exercise 3.29.**  Another way to construct an or-gate is as a compound digital logic device, built from and-gates and inverters. Define a procedure or-gate that accomplishes this. What is the delay time of the or-gate in terms of and-gate-delay and inverter-delay?

**Exercise 3.30.**  Figure [3.27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.27) shows a *sumador de acarreo* por ondulación formado al unir *n* sumadores completos. Esta es la forma más simple de sumador paralelo para sumar dos números binarios *de n bits. Las entradas A* 1 , A 2 , A 3 , ... , A *n* y B 1 , B 2 , B 3 , ... , B *n* son los dos números binarios que se van a sumar (cada A *k* y B *k* es un 0 o un 1). El circuito genera S 1 , S 2 , S 3 , ... , S *n* , los *n* bits de la suma, y ​​C, el acarreo de la adición. Escriba un procedimiento sumador de acarreo por ondulación que genere este circuito. El procedimiento debe tomar como argumentos tres listas de *n* cables cada una (A *k* , B *k* y S *k* ) y también otro cable C. La principal desventaja del sumador de acarreo por ondulación es la necesidad de esperar a que se propaguen las señales de acarreo. ¿Cuál es el retraso necesario para obtener la salida completa de un sumador de acarreo de ondulación *de n bits, expresado en términos de los retrasos para las puertas and, las puertas or y los inversores?*

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.27:**   Un sumador de acarreo de ondulación para números de *n* bits. |

**[Representando cables](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_384)**

En nuestra simulación, un cable será un objeto computacional con dos variables de estado locales: un valor de señal (que inicialmente se toma como 0) y una colección de procedimientos de acción que se ejecutarán cuando la señal cambie de valor. Implementamos el cable, utilizando el estilo de paso de mensajes, comouna colección de procedimientos locales junto con un procedimiento de despacho que selecciona la operación local apropiada, tal como hicimos con el objeto de cuenta bancaria simple en la sección   [3.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.1) :

(define (make-wire)   
  (let ((señal-valor 0) (procedimientos-de-acción '()))   
    (define (establecer-mi-señal! nuevo-valor)   
      (if (not (= señal-valor nuevo-valor))   
          (begin (establecer! señal-valor nuevo-valor)   
                 (llamar-cada-procedimientos-de-acción))   
          'done))   
    (define (aceptar-procedimiento-de-acción! proc)   
      (establecer! procedimientos-de-acción (cons proc procedimientos-de-acción))   
      (proc))   
    (define (enviar m)   
      (cond ((eq? m 'obtener-señal) valor-señal)   
            ((eq? m 'establecer-señal!) ¡establecer-mi-señal!)   
            ((eq? m 'agregar-acción!) ¡aceptar-procedimiento-de-acción!)   
            (else (error "Operación desconocida -- WIRE" m))))   
    enviar))

El procedimiento local set-my-signal! prueba si el nuevo valor de señal cambia la señal en el cable. Si es así, ejecuta cada uno de los procedimientos de acción, utilizando el siguiente procedimiento call-each , que llama a cada uno de los elementos de una lista de procedimientos sin argumentos:

(define (call-each procedimientos)   
  (if (null? procedimientos)   
      'done   
      (begin   
        ((car procedimientos))   
        (call-each (cdr procedimientos)))))

El procedimiento local accept-action-procedure! agrega el procedimiento dado a la lista de procedimientos que se ejecutarán y luego ejecuta el nuevo procedimiento una vez. (Véase el ejercicio  [3.31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.31) ).

Con el procedimiento de despacho local configurado según lo especificado, podemos proporcionar los siguientes procedimientos para acceder a las operaciones locales en los cables: [27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_385)

(define (obtener-señal cable)   
  (cable 'obtener-señal))  
(define (establecer-señal! cable nuevo-valor)   
  ((cable 'establecer-señal!) nuevo-valor))  
(define (add-action! wire procedimiento-de-acción)   
  ((wire 'add-action!) procedimiento-de-acción))

Los cables, que tienen señales que varían con el tiempo y pueden estar conectados de forma incremental a los dispositivos, son objetos mutables típicos. Los hemos modelado como procedimientos con variables de estado locales que se modifican mediante asignación. Cuando se crea un nuevo cable, se asigna un nuevo conjunto de variables de estado (mediante la expresión let en make-wire ) y se construye y devuelve un nuevo procedimiento de envío , que captura el entorno con las nuevas variables de estado.

Los cables se comparten entre los distintos dispositivos que se han conectado a ellos. Por lo tanto, un cambio realizado por una interacción con un dispositivo afectará a todos los demás dispositivos conectados al cable. El cable comunica el cambio a sus vecinos llamando a los procedimientos de acción que se le proporcionaron cuando se establecieron las conexiones.

[**La agenda**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_386)

Lo único que se necesita para completar el simulador es after-delay . La idea aquí es que mantengamos una estructura de datos, llamada *agenda* , que contiene un cronograma de cosas por hacer. Las siguientes operaciones están definidas para las agendas:

* (make-agenda)  
  devuelve una nueva agenda vacía.
* (empty-agenda? < *agenda* >)  
  es verdadero si la agenda especificada está vacía.
* (first-agenda-item < *agenda* >)  
  devuelve el primer elemento de la agenda.
* (remove-first-agenda-item! < *agenda* >)  
  modifica la agenda eliminando el primer elemento.
* (add-to-agenda! < *time* > < *action* > < *agenda* >)  
  modifica la agenda agregando el procedimiento de acción dado que se ejecutará en el momento especificado.
* (current-time < *agenda* >)  
  devuelve el tiempo de simulación actual.

La agenda particular que utilizamos se denota por the-agenda . El procedimiento after-delay agrega nuevos elementos a the-agenda :

(define (después del retraso retraso acción)   
  (¡agrega-a-la-agenda! (+ retraso (tiempo-actual la-agenda))   
                  acción   
                  la-agenda))

La simulación está controlada por el procedimiento propagate , que opera sobre la agenda y ejecuta cada procedimiento de la agenda en secuencia. En general, a medida que se ejecuta la simulación, se agregarán nuevos elementos a la agenda y propagate continuará la simulación mientras haya elementos en la agenda:

(define (propaga)   
  (if (¿agenda-vacía? la-agenda)   
      'done   
      (let ((primer-elemento (primer-elemento-de-la-agenda la-agenda)))   
        (primer-elemento)   
        (remove-first-agenda-item! la-agenda)   
        (propaga))))

**[Una simulación de muestra](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_387)**

El siguiente procedimiento, que coloca una "sonda" en un cable, muestra el simulador en acción. La sonda le dice al cable que, siempre que su señal cambie de valor, debe imprimir el nuevo valor de la señal, junto con la hora actual y un nombre que identifique al cable:

(define (nombre de la sonda cable)   
  (add-action! cable   
               (lambda ()           
                 (nueva línea)   
                 (mostrar nombre)   
                 (mostrar " ")   
                 (mostrar (hora-actual la-agenda))   
                 (mostrar " Nuevo-valor = ")   
                 (mostrar (obtener-señal cable)))))

Comenzamos inicializando la agenda y especificando retrasos para los cuadros de función primitivos:

(define la agenda (hace la agenda))   
(define retardo inversor 2)   
(define retardo compuerta y 3)   
(define retardo compuerta o 5)

Ahora definimos cuatro cables, colocando sondas en dos de ellos:

(define entrada-1 (make-wire))   
(define entrada-2 (make-wire))   
(define suma (make-wire))   
(define acarreo (make-wire))   
(probe 'suma suma)   
*suma 0 Nuevo-valor = 0*  
(probe 'acarreo acarreo)   
*acarreo 0 Nuevo-valor = 0*

A continuación conectamos los cables en un circuito semisumador (como en la figura  [3.25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.25) ), establecemos la señal en la entrada 1 en 1 y ejecutamos la simulación:

(medio sumador entrada-1 entrada-2 suma acarreo)   
*ok*  
(establecer-señal! entrada-1 1)   
*hecho*  
(propagar)   
*suma 8 Nuevo-valor = 1*   
*hecho*

La señal de suma cambia a 1 en el tiempo 8. Ahora estamos a ocho unidades de tiempo desde el comienzo de la simulación. En este punto, podemos establecer la señal en la entrada 2 en 1 y permitir que los valores se propaguen:

(establecer-señal! entrada-2 1)   
*hecho*  
(propagar)   
*llevar 11 Nuevo-valor = 1*   
*suma 16 Nuevo-valor = 0*   
*hecho*

El acarreo cambia a 1 en el momento 11 y la suma cambia a 0 en el momento 16.

**Ejercicio 3.31.**   El procedimiento interno accept-action-procedure! definido en make-wire especifica que cuando se agrega un nuevo procedimiento de acción a un cable, el procedimiento se ejecuta inmediatamente. Explique por qué es necesaria esta inicialización. En particular, rastree el ejemplo del semisumador en los párrafos anteriores y diga cómo diferiría la respuesta del sistema si hubiéramos definido accept-action-procedure! como

(define (aceptar-acción-procedimiento! proc)   
  (establece! procedimientos-de-acción (cons proc procedimientos-de-acción)))

**[Implementando la agenda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_389)**

Por último, detallamos la estructura de datos de la agenda, que contiene los procedimientos que están programados para su ejecución futura.

La agenda está compuesta de*segmentos de tiempo* . Cada segmento de tiempo es un par que consta de un número (la hora) y uncola (ver ejercicio  [3.32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.32) ) que contiene los procedimientos que están programados para ejecutarse durante ese segmento de tiempo.

(define (make-time-segment cola de tiempo)   
  (cons cola de tiempo))  
(definir (segmento-tiempo s) (coche s))  
(define (segmento-cola s) (cdr s))

Operaremos en las colas de segmentos de tiempo utilizando las operaciones de cola descritas en la sección  [3.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.2) .

La agenda en sí es una tabla unidimensional de segmentos de tiempo. Se diferencia de las tablas descritas en la sección  [3.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.3) en que los segmentos se ordenarán en orden creciente de tiempo. Además, almacenamos los*Hora actual* (es decir, la hora de la última acción procesada) al principio de la agenda. Una agenda recién construida no tiene segmentos de tiempo y tiene una hora actual de [0:28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_390)

(define (crear agenda) (lista 0))  
(definir (agenda actual) (agenda del coche))  
(define (establecer-hora-actual! hora-agenda)   
  (establecer-auto! hora-agenda))  
(definir (segmentos agenda) (cdr agenda))  
(define (set-segments! segmentos de la agenda)   
  (set-cdr! segmentos de la agenda))  
(definir (agenda del primer segmento) (automóvil (agenda de segmentos)))  
(define (agenda de segmentos restantes) (cdr (agenda de segmentos)))

Una agenda está vacía si no tiene segmentos de tiempo:

(define (agenda vacía? agenda)   
  (nulo? (segmentos agenda)))

Para añadir una acción a una agenda, primero comprobamos si la agenda está vacía. Si es así, creamos un segmento de tiempo para la acción y lo instalamos en la agenda. En caso contrario, escaneamos la agenda, examinando el tiempo de cada segmento. Si encontramos un segmento para nuestra hora señalada, añadimos la acción a la cola asociada. Si llegamos a una hora posterior a la que nos han asignado, insertamos un nuevo segmento de tiempo en la agenda justo antes de ella. Si llegamos al final de la agenda, debemos crear un nuevo segmento de tiempo al final.

(define (añadir-a-agenda! tiempo acción agenda)   
  (define (pertenece-antes? segmentos)   
    (o (nulo? segmentos)   
        (< tiempo (segmento-tiempo (segmentos de coche))))   
  (define (hacer-nuevo-segmento-de-tiempo tiempo acción)   
    (let ((q (hacer-cola)))   
      (insertar-cola! q acción)   
      (hacer-segmento-de-tiempo tiempo q)))   
  (define (añadir-a-segmentos! segmentos)   
    (if (= (segmento-tiempo (segmentos de coche)) tiempo)   
        (insertar-cola! (segmento-cola (segmentos de coche))   
                       acción)   
        (let ((resto (cdr segmentos)))   
          (if (pertenece-antes? resto)   
              (establecer-cdr!   
               segmentos   
               (cons (hacer-nuevo-segmento-de-tiempo tiempo acción)   
                     (cdr segmentos)))   
              (añadir-a-segmentos! resto)))))   
  (let ((segmentos (segmentos agenda)))   
    (si (pertenece-antes? segmentos)   
        (establecer-segmentos!   
         agenda   
         (cons (hacer-nuevo-segmento-de-tiempo tiempo acción)   
               segmentos))   
        (agregar-a-segmentos! segmentos))))

El procedimiento que elimina el primer elemento de la agenda elimina el elemento que se encuentra al principio de la cola en el primer segmento de tiempo. Si esta eliminación hace que el segmento de tiempo quede vacío, lo eliminamos de la lista de segmentos: [29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_391)

(define (elimina-primer-elemento-de-agenda! agenda)   
  (let ((q (cola-segmentos (primer-segmento-agenda))))   
    (elimina-cola! q)   
    (if (¿cola-vacia? q)   
        (establece-segmentos! agenda (resto-segmentos-agenda)))))

El primer elemento de la agenda se encuentra al principio de la cola en el primer segmento de tiempo. Siempre que extraemos un elemento, también actualizamos la hora actual: [30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_392)

(define (primer-tema-de-agenda agenda)   
  (if (¿agenda-vacía? agenda)   
      (error "La agenda está vacía -- PRIMER-TEMA-DE-AGENDA")   
      (let ((primer-seg (primer-segmento agenda)))   
        (set-current-time! agenda (segmento-tiempo primer-seg))   
        (front-queue (segmento-cola primer-seg)))))

**Ejercicio 3.32.**   Los procedimientos que se deben ejecutar durante cada segmento de tiempo de la agenda se guardan en una cola. De esta forma, los procedimientos para cada segmento se llaman en el orden en el que se agregaron a la agenda (primero en entrar, primero en salir). Explique por qué se debe utilizar este orden. En particular, trace el comportamiento de una compuerta and cuyas entradas cambian de 0,1 a 1,0 en el mismo segmento y diga cómo diferiría el comportamiento si almacenáramos los procedimientos de un segmento en una lista ordinaria, agregando y eliminando procedimientos solo al principio (último en entrar, primero en salir).

**[3.3.5 Propagación de restricciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.3.5)**

Los programas informáticos se organizan tradicionalmente como cálculos unidireccionales, que realizan operaciones sobre argumentos preestablecidos para producir los resultados deseados. Por otro lado, a menudo modelamos sistemas en términos de relaciones entre cantidades. Por ejemplo, un modelo matemático de una estructura mecánica podría incluir la información de que la deflexión *d* de una varilla de metal está relacionada con la fuerza *F* sobre la varilla, la longitud *L* de la varilla, el área de la sección transversal *A* y el módulo elástico *E* a través de la ecuación



Esta ecuación no es unidireccional. Dadas cuatro de las cantidades, podemos usarla para calcular la quinta. Sin embargo, traducir la ecuación a un lenguaje informático tradicional nos obligaría a elegir una de las cantidades para calcular en función de las otras cuatro. Por lo tanto, un procedimiento para calcular el área *A* no podría usarse para calcular la deflexión *d* , aunque los cálculos de *A* y *d* surjan de la misma ecuación. [31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_394)

En esta sección, esbozamos el diseño de un lenguaje que nos permite trabajar en términos de relaciones. Los elementos primitivos del lenguaje son*Restricciones primitivas* , que establecen que se cumplen ciertas relaciones entre cantidades. Por ejemplo, (sumador abc) especifica que las cantidades *a* , *b* y *c* deben estar relacionadas por la ecuación *a* + *b* = *c* , (multiplicador x y z) expresa la restricción *x y* = *z* , y (constante 3,14 x) dice que el valor de *x* debe ser 3,14.

Nuestro lenguaje proporciona un medio para combinar restricciones primitivas con el fin de expresar relaciones más complejas. Combinamos restricciones mediante la construcción de*Redes de restricciones* , en las que las restricciones se unen mediante*Conectores* . Un conector es un objeto que "alberga" un valor que puede participar en una o más restricciones. Por ejemplo, sabemos que la relación entre las temperaturas Fahrenheit y Celsius es



Se puede pensar en una restricción de este tipo como una red que consta de restricciones primitivas de sumador, multiplicador y constante (figura  [3.28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.28) ). En la figura, vemos a la izquierda una caja multiplicadora con tres terminales, etiquetadas *m* 1, *m* 2 y *p* . Estas conectan el multiplicador al resto de la red de la siguiente manera: El terminal *m* 1 está vinculado a un conector *C* , que mantendrá la temperatura Celsius. El terminal *m* 2 está vinculado a un conector *w* , que también está vinculado a una caja constante que contiene 9. El terminal *p* , que la caja multiplicadora restringe para que sea el producto de *m* 1 y *m* 2 , está vinculado al terminal *p* de otra caja multiplicadora, cuyo *m* 2 está conectado a una constante 5 y cuyo *m* 1 está conectado a uno de los términos de una suma.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.28:**   La relación 9 *C* = 5( *F* - 32) expresada como una red de restricciones. |

El cálculo de una red de este tipo se realiza de la siguiente manera: cuando se le asigna un valor a un conector (por parte del usuario o por una caja de restricción a la que está vinculado), este activa todas sus restricciones asociadas (excepto la restricción que lo acaba de activar) para informarles de que tiene un valor. Cada caja de restricción activada sondea a sus conectores para ver si hay suficiente información para determinar un valor para un conector. Si es así, la caja activa ese conector, que activa entonces todas sus restricciones asociadas, y así sucesivamente. Por ejemplo, en la conversión entre Celsius y Fahrenheit, *w* , *x* e *y* se establecen inmediatamente por las cajas de constantes en 9, 5 y 32, respectivamente. Los conectores activan los multiplicadores y el sumador, que determinan que no hay suficiente información para continuar. Si el usuario (o alguna otra parte de la red) establece *C* en un valor (por ejemplo, 25), se activará el multiplicador más a la izquierda y establecerá *u* en 25 · 9 = 225. Luego, *u* activa el segundo multiplicador, que establece *v* en 45, y *v* activa el sumador, que establece *F* en 77.

**[Utilizando el sistema de restricciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_395)**

Para utilizar el sistema de restricciones para realizar el cálculo de temperatura descrito anteriormente, primero creamos dos conectores, C y F , llamando al constructor make-connector , y vinculamos C y F en una red adecuada:

(define C (make-connector))   
(define F (make-connector))   
(convertidor de grados celsius a fahrenheit C F)   
*ok*

El procedimiento que crea la red se define de la siguiente manera:

(define (convertidor celsius-fahrenheit c f)   
  (sea ((u (conector-hacer))   
        (v (conector-hacer))   
        (w (conector-hacer))   
        (x (conector-hacer))   
        (y (conector-hacer)))   
    (multiplicador c w u)   
    (multiplicador v x u)   
    (sumador v y f)   
    (constante 9 w)   
    (constante 5 x)   
    (constante 32 y)   
    'ok))

Este procedimiento crea los conectores internos u , v , w , x e y , y los vincula como se muestra en la figura  [3.28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.28) utilizando los constructores de restricción primitivos adder , multiplier y constant . Al igual que con el simulador de circuitos digitales de la sección  [3.3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.4) , expresar estas combinaciones de elementos primitivos en términos de procedimientos proporciona automáticamente a nuestro lenguaje un medio de abstracción para objetos compuestos.

Para observar la red en acción, podemos colocar sondas en los conectores C y F , utilizando un procedimiento de sonda similar al que utilizamos para monitorear los cables en la sección  [3.3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.4) . Colocar una sonda en un conector hará que se imprima un mensaje cada vez que se le asigne un valor al conector:

(sonda "temperatura Celsius" C)   
(sonda "temperatura Fahrenheit" F)

A continuación, establecemos el valor de C en 25. (El tercer argumento de set-value! le dice a C que esta directiva proviene del usuario ).

(establecer valor! C 25 'usuario)   
*Sonda: temperatura Celsius = 25*   
*Sonda: temperatura Fahrenheit = 77*   
*hecho*

La sonda en C se activa y comunica el valor. C también propaga su valor a través de la red como se describió anteriormente. Esto establece F en 77, que es comunicado por la sonda en F.

Ahora podemos intentar establecer F en un nuevo valor, digamos 212:

(establecer-valor! F 212 'usuario)   
*Error! Contradicción (77 212)*

El conector se queja de que ha detectado una contradicción: su valor es 77 y alguien está intentando establecerlo en 212. Si realmente queremos reutilizar la red con nuevos valores, podemos indicarle a C que olvide su valor anterior:

(¡olvida el valor! C 'usuario)   
*Sonda: temperatura Celsius = ?*   
*Sonda: temperatura Fahrenheit = ?*   
*hecho*

C descubre que el usuario , que estableció su valor originalmente, ahora está retractándose de ese valor, por lo que C acepta perder su valor, como lo muestra la sonda, e informa al resto de la red sobre este hecho. Esta información finalmente se propaga a F , que ahora descubre que no tiene motivos para seguir creyendo que su propio valor es 77. Por lo tanto, F también renuncia a su valor, como lo muestra la sonda.

Ahora que F no tiene valor, somos libres de establecerlo en 212:

(establecer valor! F 212 'usuario)   
*Sonda: temperatura Fahrenheit = 212*   
*Sonda: temperatura Celsius = 100*   
*hecho*

Este nuevo valor, cuando se propaga a través de la red, obliga a que C tenga un valor de 100, y esto es registrado por la sonda en C. Observe que se está utilizando la misma red para calcular C dado F y para calcular F dado  C. Esta no direccionalidad del cálculo es la característica distintiva de los sistemas basados ​​en restricciones.

**[Implementación del sistema de restricciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_396)**

El sistema de restricciones se implementa a través de objetos procedimentales con estado local, de una manera muy similar al simulador de circuitos digitales de la sección  [3.3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.4) . Aunque los objetos primitivos del sistema de restricciones son algo más complejos, el sistema en general es más simple, ya que no hay preocupación por agendas y retrasos lógicos.

Las operaciones básicas sobre los conectores son las siguientes:

* (has-value? <conector> *)* indica  
  si el conector tiene un valor.
* (get-value < *conector* >)  
  devuelve el valor actual del conector.
* (set-value! < *connector* > < *new-value* > < *informant* >)  
  indica que el informante está solicitando al conector que establezca su valor en el nuevo valor.
* (olvidar-valor! < *conector* > < *retractor* >)  
  le dice al conector que el retractor le solicita que olvide su valor.
* (connect < *connector* > < *new-constraint* >)  
  le dice al conector que participe en la nueva restricción.

Los conectores se comunican con las restricciones mediante los procedimientos inform-about-value , que le dice a la restricción dada que el conector tiene un valor, e inform-about-no-value , que le dice a la restricción que el conector ha perdido su valor.

Adder construye una restricción de suma entre los conectores de sumando a1 y a2 y un conector de suma . Un sumador se implementa como un procedimiento con estado local (el procedimiento se muestra a continuación):

(define (sumador a1 a2 suma)   
  (define (procesar-nuevo-valor)   
    (cond ((y (¿tiene-valor? a1) (¿tiene-valor? a2))   
           (establecer-valor! suma   
                       (+ (obtener-valor a1) (obtener-valor a2))   
                       yo))   
          ((y (¿tiene-valor? a1) (¿tiene-valor? suma))   
           (establecer-valor! a2   
                       (- (obtener-valor suma) (obtener-valor a1))   
                       yo))   
          ((y (¿tiene-valor? a2) (¿tiene-valor? suma))   
           (establecer-valor! a1   
                       (- (obtener-valor suma) (obtener-valor a2))   
                       yo))))   
  (define (procesar-olvidar-valor)   
    (olvidar-valor! suma yo)   
    (olvidar-valor! a1 yo)   
    (olvidar-valor! a2 yo)   
    (procesar-nuevo-valor))   
  (define (yo petición)   
    (cond ((eq? petición 'Tengo-un-valor)     
           (procesar-nuevo-valor))   
          ((eq? solicitud 'perdí-mi-valor)    
           (procesar-olvidar-valor))   
          (de lo contrario    
           (error "Solicitud desconocida -- ADDER" solicitud))))   
  (conectar a1 yo)   
  (conectar a2 yo)   
  (conectar suma yo)   
  yo)

Adder conecta el nuevo sumador a los conectores designados y lo devuelve como su valor. El procedimiento me , que representa al sumador, actúa como un envío a los procedimientos locales. Las siguientes ``interfaces de sintaxis'' (ver nota al pie  [27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#footnote_Temp_385) en la sección  [3.3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.4) ) se utilizan junto con el envío:

(define (restricción de informar sobre el valor)   
  (restricción 'tengo un valor))  
(define (informar sobre la restricción sin valor)   
  (restricción 'perdí mi valor))

El procedimiento local del sumador process-new-value se llama cuando se informa al sumador de que uno de sus conectores tiene un valor. El sumador primero comprueba si a1 y a2 tienen valores. Si es así, le dice a sum que establezca su valor en la suma de los dos sumandos. El argumento informante de set-value! es me , que es el objeto sumador en sí. Si a1 y a2 no tienen valores, entonces el sumador comprueba si quizás a1 y sum tengan valores. Si es así, establece a2 en la diferencia de estos dos. Finalmente, si a2 y sum tienen valores, esto le da al sumador suficiente información para establecer a1 . Si se le dice al sumador que uno de sus conectores ha perdido un valor, solicita que todos sus conectores pierdan ahora sus valores. (Solo se pierden realmente los valores que fueron establecidos por este sumador). Luego ejecuta process-new-value . La razón de este último paso es que uno o más conectores aún pueden tener un valor (es decir, un conector puede haber tenido un valor que no fue establecido originalmente por el sumador) y estos valores pueden necesitar ser propagados nuevamente a través del sumador.

Un multiplicador es muy similar a un sumador. Fijará su producto en 0 si alguno de los factores es 0, incluso si el otro factor no se conoce.

(define (multiplicador m1 m2 producto)   
  (define (proceso-nuevo-valor)   
    (cond ((o (y (¿tiene-valor? m1) (= (obtener-valor m1) 0))   
               (y (¿tiene-valor? m2) (= (obtener-valor m2) 0)))   
           (¡establecer-valor! producto 0 yo))   
          ((y (¿tiene-valor? m1) (¿tiene-valor? m2))   
           (¡establecer-valor! producto   
                       (\* (obtener-valor m1) (obtener-valor m2))   
                       yo))   
          ((y (¿tiene-valor? producto) (¿tiene-valor? m1))   
           (¡establecer-valor! m2   
                       (/ (obtener-valor producto) (obtener-valor m1))   
                       yo))   
          ((y (¿tiene-valor? producto) (¿tiene-valor? m2))   
           (¡establecer-valor! m1   
                       (/ (obtener-valor producto) (obtener-valor m2))   
                       yo))))   
  (define (proceso-olvidar-valor)   
    (¡olvidar-valor! producto yo)   
    (¡olvidar-valor! m1 yo)   
    (¡olvidar-valor! m2 yo)   
    (proceso-nuevo-valor))   
  (define (yo solicitud)   
    (cond ((eq? solicitud 'tengo-un-valor)   
           (proceso-nuevo-valor))   
          ((eq? solicitud 'perdí-mi-valor)   
           (proceso-olvidar-valor))   
          (else   
           (error "Solicitud desconocida -- MULTIPLICADOR" solicitud))))   
  (conectar m1 yo)   
  (conectar m2 yo)   
  (conectar producto yo)   
  yo)

Un constructor constante simplemente establece el valor del conector designado. Cualquier mensaje del tipo "Tengo un valor" o "Perdí mi valor" enviado al cuadro de constante producirá un error.

(define (conector de valor constante)   
  (define (solicitud mía)   
    (error "Solicitud desconocida -- solicitud CONSTANTE")   
  (conectar conector mí)   
  (establecer-valor! valor del conector mí)   
  mí)

Finalmente, una sonda imprime un mensaje sobre la activación o desactivación del conector designado:

(define (nombre del conector de sonda)   
  (define (imprime el valor de la sonda)   
    (nueva línea)   
    (muestra "Sonda: ")   
    (nombre para mostrar)   
    (muestra " = ")   
    (muestra valor))   
  (define (procesa el nuevo valor)   
    (imprime la sonda (obtiene el valor del conector)))   
  (define (procesa el olvido del valor)   
    (imprime la sonda "?"))   
  (define (me request)   
    (cond ((eq? request 'tengo un valor)   
           (procesa el nuevo valor))   
          ((eq? request 'perdí mi valor)   
           (procesa el olvido del valor))   
          (else   
           (error "Solicitud desconocida -- SONDA" solicitud))))   
  (conecta el conector me)   
  me)

**[Representando conectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_397)**

Un conector se representa como un objeto de procedimiento con variables de estado locales valor , el valor actual del conector; informante , el objeto que establece el valor del conector; y restricciones , una lista de las restricciones en las que participa el conector.

(define (make-connector)   
  (let ((value false) (informant false) (constraints '()))   
    (define (set-my-value newval setter)   
      (cond ((not (has-value? me))   
             (set! value newval)   
             (set! informant setter)   
             (for-each-except setter   
                              informar-sobre-valor   
                              restricciones))   
            ((not (= value newval))   
             (error "Contradiction" (list value newval)))   
            (else 'ignored)))   
    (define (forget-my-value retractor)   
      (if (eq? retractor informante)   
          (begin (set! informant false)   
                 (for-each-except retractor   
                                  informar-sobre-ningún-valor   
                                  restricciones))   
          'ignored))   
    (define (connect new-constraint)   
      (if (not (memq new-constraint restricciones))   
          (set! limitations    
                (cons new-constraint restricciones)))   
      (if (has-value? yo)   
          (informar-sobre-valor nueva-restricción))   
      'hecho)   
    (definir (yo solicitud)   
      (cond ((eq? solicitud 'tiene-valor?)   
             (si informante verdadero falso))   
            ((eq? solicitud 'valor) valor)   
            ((eq? solicitud 'establecer-valor!) establecer-mi-valor)   
            ((eq? solicitud 'olvidar) olvidar-mi-valor)   
            ((eq? solicitud 'conectar) conectar)   
            (sino (error "Operación desconocida -- CONECTOR"   
                         solicitud))))   
    yo))

El procedimiento local del conector set-my-value se llama cuando hay una solicitud para establecer el valor del conector. Si el conector no tiene actualmente un valor, establecerá su valor y recordará como informante la restricción que solicitó que se estableciera el valor. [32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_398) Luego, el conector notificará a todas sus restricciones participantes excepto a la restricción que solicitó que se estableciera el valor. Esto se logra utilizando el siguiente iterador, que aplica un procedimiento designado a todos los elementos de una lista excepto a uno determinado:

(define (lista de procedimientos de excepción for-each-except)   
  (define (elementos de bucle)   
    (cond ((¿elementos null?) 'done)   
          ((eq? (elementos de automóvil) excepción) (bucle (elementos de CDR)))   
          (else (procedimiento (elementos de automóvil))   
                (bucle (elementos de CDR)))))   
  (lista de bucles))

Si se le pide a un conector que olvide su valor, ejecuta el procedimiento local forget-my-value , que primero verifica que la solicitud provenga del mismo objeto que estableció el valor originalmente. Si es así, el conector informa a sus restricciones asociadas sobre la pérdida del valor.

El procedimiento local connect añade la nueva restricción designada a la lista de restricciones si no se encuentra ya en ella. Luego, si el conector tiene un valor, informa de ello a la nueva restricción.

El procedimiento del conector me sirve como un envío a los demás procedimientos internos y también representa al conector como un objeto. Los siguientes procedimientos proporcionan una interfaz de sintaxis para el envío:

(define (tiene-valor? conector)   
  (conector 'tiene-valor?))  
(define (conector get-value)   
  (conector 'value))  
(define (establecer-valor! conector nuevo-valor informante)   
  ((conector 'establecer-valor!) nuevo-valor informante))  
(define (olvidar-valor! conector retractor)   
  ((conector 'olvidar) retractor))  
(define (conectar conector nueva-restricción)   
  ((conector 'conectar) nueva-restricción))

**Ejercicio 3.33.**   Utilizando restricciones primitivas de multiplicador, sumador y constante, defina un procedimientopromediador que toma tres conectores a , b y c como entradas y establece la restricción de que el valor de c es el promedio de los valores de a y b .

**Ejercicio 3.34.**  Louis Reasoner quiere construir un dispositivo de restricción con dos terminales, de modo que el valor del conector b en el segundo terminal sea siempre el cuadrado del valor a en el primer terminal. Propone el siguiente dispositivo simple formado por un multiplicador:

(define (al cuadrado a b)   
  (multiplicador a a b))

Esta idea tiene un grave fallo. Explíquelo.

**Ejercicio 3.35.**  Ben Bitdiddle le dice a Louis que una forma de evitar el problema del ejercicio  [3.34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.34) es definir un cuadrado como una nueva restricción primitiva. Complete las partes faltantes en el esquema de Ben para obtener un procedimiento que implemente dicha restricción:

(define (al cuadrado a b)   
  (define (proceso-nuevo-valor)   
    (si (¿tiene-valor? b)   
        (si (< (obtener-valor b) 0)   
            (error "al cuadrado menor que 0 -- AL CUADRADO" (obtener-valor b))   
            < *alternativa1* >)   
        < *alternativa2* >))   
  (define (proceso-olvidar-valor) < *cuerpo1* >)   
  (define (yo solicitud) < *cuerpo2* >)   
  < *resto de la definición* >   
  yo)

**Ejercicio 3.36.**   Supongamos que evaluamos la siguiente secuencia de expresiones en el entorno global:

(define a (make-connector))   
(define b (make-connector))   
(establece-valor! a 10 'usuario)

En algún momento durante la evaluación del conjunto-valor!, se evalúa la siguiente expresión del procedimiento local del conector:

(restricciones de información sobre el valor del establecedor para cada uno excepto)

Dibuje un diagrama de entorno que muestre el entorno en el que se evalúa la expresión anterior.

**Ejercicio 3.37.**   El procedimiento de conversión de grados Celsius a Fahrenheit es engorroso en comparación con un estilo de definición más orientado a la expresión, como

(define (convertidor-celsius-fahrenheit-x)   
  (c+ (c\* (c/ (cv 9) (cv 5))   
          x)   
      (cv 32)))   
(define C (conector-hacer))   
(define F (convertidor-celsius-fahrenheit-C))

Aquí c+ , c\* , etc. son las versiones «constrictivas» de las operaciones aritméticas. Por ejemplo, c+ toma dos conectores como argumentos y devuelve un conector que está relacionado con estos mediante una restricción de suma:

(define (c+ x y)   
  (sea ((z (conector-de-hacer)))   
    (sumador x y z)   
    z))

Defina procedimientos análogos c- , c\* , c/ y cv (valor constante) que nos permitan definir restricciones compuestas como en el ejemplo del convertidor anterior. [33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_404)

[16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_349) Set-car!yset-cdr!devuelven una implementación dependientevalores. Al igual que set!, deben usarse solo por su efecto.

[17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_350) Vemos a partir de esto que las operaciones de mutación en listas pueden crear ``basura'' que no es parte de ninguna estructura accesible. Veremos en la sección [5.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.3.2)que los sistemas de administración de memoria de Lisp incluyen un*recolector de basura* , que identifica y recicla el espacio de memoria utilizado por pares innecesarios.

[18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_351) Get-new-paires una de las operaciones que se deben implementar como parte de la gestión de memoria requerida por una implementación de Lisp. Hablaremos de esto en la sección [5.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.3.1).

[19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_356) Los dos pares son distintos porque cada llamada aconsdevuelve un nuevo par. Los símbolos son compartidos; en Scheme hay un símbolo único con cualquiernombre. Dado que Scheme no ofrece ninguna forma de mutar un símbolo, esta compartición es indetectable. Observe también que la compartición es lo que nos permite comparar símbolos utilizando eq? , que simplemente verifica la igualdad de punteros.

[20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_357) Las sutilezas de tratar con el uso compartido de objetos de datos mutables reflejan los problemas subyacentes de "igualdad" y "cambio" que se plantearon en la sección [3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.3). Allí mencionamos que admitir cambios en nuestro lenguaje requiere que un objeto compuesto tenga una "identidad" que sea algo diferente de las piezas de las que está compuesto. En Lisp, consideramos que esta "identidad" es la cualidad que se prueba medianteeq?, es decir, mediante la igualdad de punteros. Dado que en la mayoría de las implementaciones de Lisp un puntero es esencialmente una dirección de memoria, estamos "resolviendo el problema" de definir la identidad de los objetos al estipular que un objeto de datos "en sí mismo" es la información almacenada en algún conjunto particular de ubicaciones de memoria en la computadora. Esto es suficiente para programas Lisp simples, pero difícilmente es una forma general de resolver el problema de la "igualdad" en los modelos computacionales.

[21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_364) Por otra parte, desde el punto de vista de la implementación, la asignación requiere que modifiquemos el entorno, que es en sí mismo una estructura de datos mutable. Por lo tanto, la asignación y la mutación son equipotentes: cada una puede implementarse en términos de la otra.

[22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_366) Si el primer elemento es el elemento final en la cola, el puntero frontal será la lista vacía después de la eliminación, que marcará la cola como vacía; no necesitamos preocuparnos por actualizar el puntero posterior, que seguirá apuntando al elemento eliminado, porqueempty-queue?solo mira el puntero frontal.

[23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_370) Tenga cuidado de no hacer que el intérprete intente imprimir una estructura que contenga ciclos. (Véase el ejercicio [3.13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.13)).

[24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_371) Debido a queassocusaequal?,puede reconocer claves que son símbolos, números o estructuras de lista.

[25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_372) Por lo tanto, el primer par de columnas es el objeto que representa la tabla ``en sí''; es decir, un puntero a la tabla es un puntero a este par. Este mismo par de columnas siempre inicia la tabla. Si no ordenáramos las cosas de esta manera,insert!tendría que devolver un nuevo valor para el inicio de la tabla cuando añadiera un nuevo registro.

[26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_379) Un sumador completo es un elemento básico del circuito que se utiliza para sumar dos números binarios. Aquí A y B son los bits en las posiciones correspondientes en los dos números que se van a sumar, y C *i n* es el bit de acarreo de la suma un lugar hacia la derecha. El circuito genera SUM, que es el bit de suma en la posición correspondiente, y C *o u t* , que es el bit de acarreo que se propagará hacia la izquierda.

[27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_385) Estos procedimientos son simplemente azúcar sintáctico que permitenos permite usar la sintaxis procedimental ordinaria para acceder a los procedimientos locales de los objetos. Es sorprendente que podamos intercambiar el papel de «procedimientos» y «datos» de una manera tan sencilla. Por ejemplo, si escribimos (wire 'get-signal) pensamos en wire como un procedimiento que se llama con el mensaje get-signal como entrada. Alternativamente, escribir (get-signal wire) nos anima a pensar en wire como un objeto de datos que es la entrada de un procedimiento get-signal . La verdad del asunto es que, en un lenguaje en el que podemos tratar los procedimientos como objetos, no hay una diferencia fundamental entre «procedimientos» y «datos», y podemos elegir nuestra sintaxis que nos permita programar en cualquier estilo que elijamos.

[28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_390) La agenda es unalista encabezada, como las tablas de la sección  [3.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.3) , pero como la lista está encabezada por la hora, no necesitamos un encabezado ficticio adicional (como el símbolo \*table\* usado con las tablas).

[29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_391) Observe que laexpresiónif*<alternative>*. Tal "if"se utiliza para decidir si se debe hacer algo, en lugar de seleccionar entre dos expresiones. Una expresión if devuelve un valor no especificado si el predicado es falso y no hay ninguna *<alternativa>* .

[De esta manera](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_392) , la hora actual siempre será la hora de la última acción procesada. Al almacenar esta hora en la cabecera de la agenda, se garantiza que seguirá estando disponible incluso si se ha eliminado el segmento de tiempo asociado.

[31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_394) La propagación de restricciones apareció por primera vez en el increíblemente progresistaSistema SKETCHPAD deIvan Sutherland (1963). Un hermoso sistema de propagación de restricciones basado en laEl lenguaje Smalltalk fue desarrollado porAlan Borning (1977) enCentro de investigación de Xerox en Palo Alto. Sussman, Stallman y SteelePropagación de restricciones aplicada al análisis de circuitos eléctricos (Sussman y Stallman 1975; Sussman y Steele 1980). TK!Solver (Konopasek y Jayaraman 1984) es un entorno de modelado extenso basado en restricciones.

[32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_398) Eldefinidorpodría no ser una restricción. En nuestro ejemplo de temperatura, usamosusuariocomodefinidor.

[33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_404) El formato orientado a expresiones es conveniente porque evita la necesidad de nombrar las expresiones intermedias en un cálculo. Nuestra formulación original de laEl lenguaje de restricciones es engorroso de la misma manera que muchos lenguajes son engorrosos cuando se trata de operaciones con datos compuestos. Por ejemplo, si quisiéramos calcular el producto ( *a* + *b* ) · ( *c* + *d* ), donde las variables representan vectores, podríamos trabajar en "estilo imperativo", utilizando procedimientos que establecen los valores de los argumentos vectoriales designados pero que no devuelven vectores como valores:

(v-suma a b temp1)   
(v-suma c d temp2)   
(v-prod temp1 temp2 respuesta)

Como alternativa, podríamos tratar con expresiones, utilizando procedimientos que devuelvan vectores como valores, y así evitar mencionar explícitamente temp1 y temp2 :

(define la respuesta (v-prod (v-sum a b) (v-sum c d)))

Dado que Lisp nos permite devolver objetos compuestos como valores de procedimientos, podemos transformar nuestro lenguaje de restricciones de estilo imperativo en un estilo orientado a expresiones como se muestra en este ejercicio. En lenguajes que son pobres en el manejo de objetos compuestos, como Algol, Basic y Pascal (a menos que uno use explícitamente variables de puntero de Pascal), uno normalmente está atascado con el estilo imperativo cuando se manipulan objetos compuestos. Dada la ventaja del formato orientado a expresiones, uno podría preguntarse si hay alguna razón para haber implementado el sistema en estilo imperativo, como lo hicimos en esta sección. Una razón es que el lenguaje de restricciones no orientado a expresiones proporciona un manejador de objetos de restricción (por ejemplo, el valor del procedimiento sumador ) así como de objetos conectores. Esto es útil si deseamos extender el sistema con nuevas operaciones que se comunican con restricciones directamente en lugar de solo indirectamente a través de operaciones en conectores. Aunque es fácil implementar el estilo orientado a expresiones en términos de la implementación imperativa, es muy difícil hacer lo inverso.

[**3.4 Simultaneidad: el tiempo es esencial**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_3.4)

Hemos visto el poder de los objetos computacionales con estado local como herramientas para el modelado. Sin embargo, como advertimos en la sección  [3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.3) , este poder tiene un precio: la pérdida de transparencia referencial, lo que da lugar a una maraña de preguntas sobre la igualdad y el cambio, y la necesidad de abandonar el modelo de evaluación de sustitución en favor del modelo de entorno más complejo.

La cuestión central que se esconde tras la complejidad del estado, la igualdad y el cambio es que al introducir la asignación nos vemos obligados a admitir *el tiempo* en nuestros modelos computacionales. Antes de introducir la asignación, todos nuestros programas eran atemporales, en el sentido de que cualquier expresión que tenga un valor siempre tiene el mismo valor. En cambio, recordemos el ejemplo de modelar los retiros de una cuenta bancaria y devolver el saldo resultante, presentado al principio de la sección  [3.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.1) :

(retirar 25)   
*75*  
(retirar 25)   
*50*

En este caso, las evaluaciones sucesivas de la misma expresión dan como resultado valores diferentes. Este comportamiento surge del hecho de que la ejecución de sentencias de asignación (en este caso, asignaciones a la variable balance ) delimita *momentos en el tiempo* en los que los valores cambian. El resultado de la evaluación de una expresión depende no solo de la expresión en sí, sino también de si la evaluación ocurre antes o después de estos momentos. La construcción de modelos en términos de objetos computacionales con estado local nos obliga a enfrentar el tiempo como un concepto esencial en la programación.

Podemos ir más allá en la estructuración de modelos computacionales para que coincidan con nuestra percepción del mundo físico. Los objetos del mundo no cambian uno a la vez en secuencia, sino que los percibimos como si actuaran *simultáneamente* , todos a la vez. Por lo tanto, a menudo es natural modelar sistemas como colecciones de procesos computacionales que se ejecutan simultáneamente. Así como podemos hacer que nuestros programas sean modulares organizando modelos en términos de objetos con estados locales separados, a menudo es apropiado dividir los modelos computacionales en partes que evolucionan por separado y simultáneamente. Incluso si los programas se van a ejecutar en una computadora secuencial, la práctica de escribir programas como si se fueran a ejecutar simultáneamente obliga al programador a evitar restricciones de tiempo no esenciales y, por lo tanto, hace que los programas sean más modulares.

Además de hacer que los programas sean más modulares, la computación concurrente puede proporcionar una ventaja de velocidad sobre la computación secuencial. Las computadoras secuenciales ejecutan solo una operación a la vez, por lo que la cantidad de tiempo que lleva realizar una tarea es proporcional al número total de operaciones realizadas. [34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_405) Sin embargo, si es posible descomponer un problema en partes que sean relativamente independientes y necesiten comunicarse solo en raras ocasiones, puede ser posible asignar partes a procesadores de computación separados, lo que produce una ventaja de velocidad proporcional al número de procesadores disponibles.

Lamentablemente, las complejidades que introduce la asignación se vuelven aún más problemáticas en presencia de concurrencia. El hecho de la ejecución concurrente, ya sea porque el mundo opera en paralelo o porque lo hacen nuestras computadoras, implica una complejidad adicional en nuestra comprensión del tiempo.

**[3.4.1 La naturaleza del tiempo en sistemas concurrentes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.4.1)**

A primera vista, el tiempo parece sencillo. Es un orden impuesto a los acontecimientos. [35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_406) Para cualquier acontecimiento *A* y *B* , o bien *A* ocurre antes de *B* , *A*  y  *B* son simultáneos, o bien *A* ocurre después de *B.* Por ejemplo, volviendo al ejemplo de la cuenta bancaria, supongamos que Pedro retira 10 dólares y Pablo retira 25 dólares de una cuenta bancaria.cuenta conjunta que inicialmente contiene $100, lo que deja $65 en la cuenta. Dependiendo del orden de los dos retiros, la secuencia de saldos en la cuenta es $100 $90 $65 o $100 $75 $65. En una implementación informática del sistema bancario, esta secuencia cambiante de saldos podría modelarse mediante asignaciones sucesivas a un saldo variable .

Sin embargo, en situaciones complejas, esta perspectiva puede resultar problemática. Supongamos que Pedro y Pablo, y otras personas, acceden a la misma cuenta bancaria a través de una red de cajeros automáticos distribuidos por todo el mundo. La secuencia real de saldos en la cuenta dependerá críticamente de la cronología detallada de los accesos y de los detalles de la comunicación entre los cajeros.

Esta indeterminación en el orden de los acontecimientos puede plantear serios problemas en el diseño de sistemas concurrentes. Por ejemplo, supongamos que los retiros realizados por Pedro y Pablo se implementan como dos procesos separados que comparten una variable común, balance , y cada proceso se especifica mediante el procedimiento dado en la sección  [3.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.1) :

(define (monto a retirar)   
  (si (>= monto del saldo)   
      (begin (set! balance (- monto del saldo))   
             balance)   
      "Fondos insuficientes"))

Si los dos procesos funcionan de forma independiente, Peter podría comprobar el saldo e intentar retirar una cantidad legítima. Sin embargo, Paul podría retirar algunos fondos entre el momento en que Peter comprueba el saldo y el momento en que Peter completa el retiro, invalidando así la prueba de Peter.

Las cosas pueden ser aún peores. Considere la expresión

(establecer! saldo (- importe del saldo))

Se ejecuta como parte de cada proceso de retiro. Consta de tres pasos: (1) acceder al valor de la variable saldo ; (2) calcular el nuevo saldo; (3) establecer el saldo en este nuevo valor. Si los retiros de Peter y Paul ejecutan esta instrucción simultáneamente, entonces los dos retiros podrían intercalar el orden en el que acceden al saldo y lo establecen en el nuevo valor.

El diagrama de tiempo de la figura  [3.29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.29) muestra un orden de eventos en el que el saldo comienza en 100, Peter retira 10, Paul retira 25 y, sin embargo, el valor final del saldo es 75. Como se muestra en el diagrama, la razón de esta anomalía es que la asignación de 75 por parte de Paul al saldo se hace bajo el supuesto de que el valor del saldo que se debe decrementar es 100. Sin embargo, ese supuesto se volvió inválido cuando Peter cambió el saldo a 90. Esto es un fracaso catastrófico para el sistema bancario, porque la cantidad total de dinero en el sistema no se conserva. Antes de las transacciones, la cantidad total de dinero era $100. Después, Peter tiene $10, Paul tiene $25 y el banco tiene $75. [36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_407)

El fenómeno general que se ilustra aquí es que varios procesos pueden compartir una variable de estado común. Lo que complica este proceso es que más de un proceso puede estar intentando manipular el estado compartido al mismo tiempo. En el ejemplo de la cuenta bancaria, durante cada transacción, cada cliente debería poder actuar como si los demás clientes no existieran. Cuando un cliente cambia el saldo de una manera que depende del saldo, debe poder asumir que, justo antes del momento del cambio, el saldo sigue siendo el que él pensaba que era.

**[Comportamiento correcto de programas concurrentes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_408)**

El ejemplo anterior ejemplifica los errores sutiles que pueden introducirse en los programas concurrentes. La raíz de esta complejidad reside en las asignaciones a las variables que se comparten entre los diferentes procesos. Ya sabemos que debemos ser cuidadosos al escribir programas que utilicen set!, porque los resultados de un cálculo dependen del orden en que se producen las asignaciones. [37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_409) Con los procesos concurrentes debemos ser especialmente cuidadosos con las asignaciones, porque es posible que no podamos controlar el orden de las asignaciones realizadas por los diferentes procesos. Si se pueden realizar varios de estos cambios simultáneamente (como cuando dos depositantes acceden a una cuenta conjunta), necesitamos alguna forma de asegurarnos de que nuestro sistema se comporta correctamente. Por ejemplo, en el caso de los retiros de una cuenta bancaria conjunta, debemos asegurarnos de que se conserva el dinero. Para que los programas concurrentes se comporten correctamente, es posible que tengamos que poner algunas restricciones a la ejecución concurrente.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.29:**   Diagrama de tiempo que muestra cómo intercalar el orden de eventos en dos retiros bancarios puede llevar a un saldo final incorrecto. |

Una posible restricción a la concurrencia estipularía que no pueden ocurrir al mismo tiempo dos operaciones que cambien cualquier variable de estado compartida. Este es un requisito extremadamente estricto. Para la banca distribuida, requeriría que el diseñador del sistema se asegure de que solo se pueda realizar una transacción a la vez. Esto sería ineficiente y demasiado conservador. La Figura  [3.30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.30) muestra a Peter y Paul compartiendo una cuenta bancaria, donde Paul también tiene una cuenta privada. El diagrama ilustra dos retiros de la cuenta compartida (uno por parte de Peter y otro por parte de Paul) y un depósito en la cuenta privada de Paul. [38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_410) Los dos retiros de la cuenta compartida no deben ser simultáneos (ya que ambos acceden y actualizan la misma cuenta), y el depósito y el retiro de Paul no deben ser simultáneos (ya que ambos acceden y actualizan el monto en la billetera de Paul). Pero no debería haber ningún problema en permitir que el depósito de Paul a su cuenta privada se realice simultáneamente con el retiro de Peter de la cuenta compartida.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.30:**   Depósitos y retiros simultáneos de una cuenta conjunta en el Banco1 y una cuenta privada en el Banco2. |

Una restricción menos estricta de la concurrencia garantizaría que un sistema concurrente produzca el mismo resultado que si los procesos se hubieran ejecutado secuencialmente en algún orden. Este requisito tiene dos aspectos importantes. En primer lugar, no requiere que los procesos se ejecuten secuencialmente, sino solo que produzcan resultados que sean los mismos *que si* se hubieran ejecutado secuencialmente. Para el ejemplo de la figura  [3.30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.30) , el diseñador del sistema de cuenta bancaria puede permitir con seguridad que el depósito de Paul y el retiro de Peter ocurran simultáneamente, porque el resultado neto será el mismo que si las dos operaciones hubieran ocurrido secuencialmente. En segundo lugar, puede haber más de un posible resultado "correcto" producido por un programa concurrente, porque solo requerimos que el resultado sea el mismo que para *algún* orden secuencial. Por ejemplo, supongamos que la cuenta conjunta de Peter y Paul comienza con $100, y Peter deposita $40 mientras que Paul retira simultáneamente la mitad del dinero de la cuenta. Luego, la ejecución secuencial podría dar como resultado que el saldo de la cuenta sea $70 o $90 (ver ejercicio  [3.38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.38) ). [39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_411)

Aún existen requisitos más débiles para la correcta ejecución de programas concurrentes. Un programa para simularLa difusión (por ejemplo, el flujo de calor en un objeto) puede consistir en una gran cantidad de procesos, cada uno de los cuales representa un pequeño volumen de espacio, que actualizan sus valores simultáneamente. Cada proceso cambia repetidamente su valor al promedio de su propio valor y los valores de sus vecinos. Este algoritmo converge a la respuesta correcta independientemente del orden en que se realicen las operaciones; no hay necesidad de ninguna restricción sobre el uso simultáneo de los valores compartidos.

**Ejercicio 3.38.**   Supongamos que Peter, Paul y Mary comparten una cuenta bancaria conjunta que inicialmente contiene $100. Al mismo tiempo, Peter deposita $10, Paul retira $20 y Mary retira la mitad del dinero de la cuenta, ejecutando los siguientes comandos:

|  |  |
| --- | --- |
| Pedro: | (¡listo! equilibrio (+ equilibrio 10)) |
| Pablo: | (¡listo! balance (- balance 20)) |
| María: | (¡listo! equilibrio (- equilibrio (/ equilibrio 2))) |

a. Enumere todos los diferentes valores posibles para el saldo después de que se hayan completado estas tres transacciones, suponiendo que el sistema bancario obliga a que los tres procesos se ejecuten secuencialmente en algún orden.

b. ¿Qué otros valores podrían generarse si el sistema permite que los procesos se intercalen? Dibuje diagramas de tiempos como el de la figura  [3.29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.29) para explicar cómo pueden generarse estos valores.

**[3.4.2 Mecanismos para controlar la concurrencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.4.2)**

Hemos visto que la dificultad de manejar procesos concurrentes radica en la necesidad de considerar el intercalado del orden de los eventos en los diferentes procesos. Por ejemplo, supongamos que tenemos dos procesos, uno con tres eventos ordenados ( *a* , *b* , *c* ) y otro con tres eventos ordenados ( *x* , *y* , *z* ). Si los dos procesos se ejecutan simultáneamente, sin restricciones sobre cómo se intercala su ejecución, entonces hay 20 posibles ordenamientos diferentes para los eventos que son consistentes con los ordenamientos individuales para los dos procesos:

Forma

El contenido generado por IA puede ser incorrecto.

Como programadores que diseñamos este sistema, tendríamos que considerar los efectos de cada uno de estos 20 ordenamientos y comprobar que cada comportamiento es aceptable. Este enfoque se vuelve rápidamente difícil de manejar a medida que aumenta la cantidad de procesos y eventos.

Un enfoque más práctico para el diseño de sistemas concurrentes es idear mecanismos generales que nos permitan restringir el entrelazado de procesos concurrentes de modo que podamos estar seguros de que el comportamiento del programa es correcto. Se han desarrollado muchos mecanismos para este propósito. En esta sección, describimos uno de ellos, el *serializador* .

**[Serialización del acceso al estado compartido](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_413)**

La serialización implementa la siguiente idea: los procesos se ejecutarán simultáneamente, pero habrá ciertas colecciones de procedimientos que no se podrán ejecutar simultáneamente. Más precisamente, la serialización crea conjuntos distinguidos de procedimientos de modo que solo se permita una ejecución de un procedimiento en cada conjunto serializado a la vez. Si se está ejecutando algún procedimiento del conjunto, entonces un proceso que intente ejecutar cualquier procedimiento del conjunto se verá obligado a esperar hasta que haya finalizado la primera ejecución.

Podemos utilizar la serialización para controlar el acceso a las variables compartidas. Por ejemplo, si queremos actualizar una variable compartida en función del valor anterior de esa variable, ponemos el acceso al valor anterior de la variable y la asignación del nuevo valor a la variable en el mismo procedimiento. A continuación, nos aseguramos de que ningún otro procedimiento que asigne a la variable pueda ejecutarse simultáneamente con este procedimiento serializando todos estos procedimientos con el mismo serializador. Esto garantiza que el valor de la variable no se pueda cambiar entre un acceso y la asignación correspondiente.

**[Serializadores en Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_414)**

Para hacer más concreto el mecanismo anterior, supongamos que hemos ampliado Scheme para incluir un procedimiento llamadoejecución paralela :

(ejecución paralela < *p 1* > < *p 2* >  ... < *p k* >)

Cada < *p* > debe ser un procedimiento sin argumentos. La ejecución en paralelo crea un proceso independiente para cada < *p* >, que aplica < *p* > (a ningún argumento). Todos estos procesos se ejecutan simultáneamente. [40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_415)

Como ejemplo de cómo se utiliza esto, considere

(define x 10)   
  
(ejecuta en paralelo (lambda () (establece! x (\* x x)))   
                  (lambda () (establece! x (+ x 1))))

Esto crea dos procesos simultáneos: *P* 1 , que establece x en x multiplicado por x , y *P* 2 , que incrementa x . Una vez finalizada la ejecución, x quedará con uno de los cinco valores posibles, según el entrelazado de los eventos de *P* 1 y *P* 2 :

|  |  |
| --- | --- |
| 101: | *P* 1 establece x en 100 y luego *P* 2 incrementa x en 101. |
| 121: | *P* 2 incrementa x a 11 y luego *P* 1 establece x en x multiplicado por x . |
| 110: | *P* 2 cambia x de 10 a 11 entre las dos veces que *P* 1 accede al valor de x durante la evaluación de (\* xx) . |
| 11: | *P* 2 accede a x , luego *P* 1 establece x en 100, luego *P* 2 establece x . |
| 100: | *P* 1 accede a x (dos veces), luego *P* 2 establece x en 11, luego *P* 1 establece x . |
|  |  |

Podemos restringir la concurrencia mediante el uso de procedimientos serializados, que son creados por *serializers* . Los serializadores son construidos por make-serializer , cuya implementación se muestra a continuación. Un serializador toma un procedimiento como argumento y devuelve un procedimiento serializado que se comporta como el procedimiento original. Todas las llamadas a un serializador dado devuelven procedimientos serializados en el mismo conjunto.

Por lo tanto, a diferencia del ejemplo anterior, al ejecutar

(define x 10)   
  
(define s (make-serializer))   
  
(paralelo-ejecuta (s (lambda () (set! x (\* x x))))   
                  (s (lambda () (set! x (+ x 1)))))

sólo puede producir dos valores posibles para x , 101 o 121. Las demás posibilidades se eliminan, porque la ejecución de *P* 1 y *P* 2 no se puede intercalar.

A continuación se muestra una versión del procedimiento de creación de cuenta de la sección  [3.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.1) , donde se han serializado los depósitos y retiros:

(define (make-account balance)   
  (define (monto de retiro)   
    (if (>= monto de saldo)   
        (begin (set! balance (- monto de saldo))   
               saldo)   
        "Fondos insuficientes"))   
  (define (monto de depósito)   
    (set! balance (+ monto de saldo))   
    saldo)   
  (let ((protected (make-serializer)))   
    (define (dispatch m)   
      (cond ((eq? m 'retiro) (retiro protegido))   
            ((eq? m 'depósito) (depósito protegido))   
            ((eq? m 'saldo) saldo)   
            (else (error "Solicitud desconocida -- MAKE-ACCOUNT"   
                         m))))   
    dispatch))

Con esta implementación, dos procesos no pueden retirar o depositar en una sola cuenta simultáneamente. Esto elimina la fuente del error ilustrado en la figura  [3.29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.29) , donde Peter cambia el saldo de la cuenta entre los momentos en que Paul accede al saldo para calcular el nuevo valor y cuando Paul realmente realiza la asignación. Por otro lado, cada cuenta tiene su propio serializador, de modo que los depósitos y retiros para diferentes cuentas pueden realizarse simultáneamente.

**Ejercicio 3.39.**   ¿Cuál de las cinco posibilidades en la ejecución paralela mostrada arriba permanece si en cambio serializamos la ejecución de la siguiente manera?

(define x 10)   
  
(define s (make-serializer))   
  
(paralelo-ejecuta (lambda () (set! x ((s (lambda () (\* x x))))))   
                  (s (lambda () (set! x (+ x 1)))))

**Ejercicio 3.40.**   Dar todos los posibles valores de x que pueden resultar de la ejecución

(define x 10)   
  
(ejecuta en paralelo (lambda () (establece! x (\* x x)))   
                  (lambda () (establece! x (\* x x x))))

¿Cuál de estas posibilidades queda si en cambio utilizamos procedimientos serializados?

(define x 10)   
  
(define s (make-serializer))   
  
(paralelo-ejecuta (s (lambda () (set! x (\* x x))))   
                  (s (lambda () (set! x (\* x x x)))))

**Ejercicio 3.41.**   A Ben Bitdiddle le preocupa que sería mejor implementar la cuenta bancaria de la siguiente manera (donde se ha cambiado la línea comentada):

(define (make-account balance)   
  (define (monto de retiro)   
    (if (>= monto de saldo)   
        (begin (set! balance (- monto de saldo))   
               saldo)   
        "Fondos insuficientes"))   
  (define (monto de depósito)   
    (set! balance (+ monto de saldo))   
    saldo)   
  *;; continúa en la página siguiente*  
  
  (let ((protected (make-serializer)))   
    (define (dispatch m)   
      (cond ((eq? m 'withdraw) (protected retreat))   
            ((eq? m 'deposit) (protected deposit))   
            ((eq? m 'balance)   
             ((protected (lambda () balance))))  *; serialized*  
            (else (error "Solicitud desconocida -- MAKE-ACCOUNT"   
                         m))))   
    dispatch))

porque permitir el acceso no serializado al saldo bancario puede dar lugar a un comportamiento anómalo. ¿Estás de acuerdo? ¿Hay algún escenario que demuestre la preocupación de Ben?

**Ejercicio 3.42.**   Ben Bitdiddle sugiere que es una pérdida de tiempo crear un nuevo procedimiento serializado en respuesta a cada mensaje de retiro y depósito . Dice que make-account podría modificarse para que las llamadas a protected se realicen fuera del procedimiento de envío . Es decir, una cuenta devolvería el mismo procedimiento serializado (que se creó al mismo tiempo que la cuenta) cada vez que se le solicita un procedimiento de retiro.

(define (make-account balance)   
  (define (monto de retiro)   
    (if (>= monto de saldo)   
        (begin (set! balance (- monto de saldo))   
               saldo)   
        "Fondos insuficientes"))   
  (define (monto de depósito)   
    (set! balance (+ monto de saldo))   
    saldo)   
  (let ((protected (make-serializer)))   
    (let ((protected-withdraw (retiro protegido))   
          (protected-deposit (depósito protegido)))   
      (define (dispatch m)   
        (cond ((eq? m 'withdraw) retiro-protegido)   
              ((eq? m 'deposit) depósito-protegido)   
              ((eq? m 'balance) saldo)   
              (else (error "Solicitud desconocida -- MAKE-ACCOUNT"   
                           m))))   
      dispatch)))

¿Es seguro realizar este cambio? En particular, ¿existe alguna diferencia en la concurrencia permitida por estas dos versiones de make-account ?

**[Complejidad de utilizar múltiples recursos compartidos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_420)**

Los serializadores proporcionan una poderosa abstracción que ayuda a aislar las complejidades de los programas concurrentes para que se puedan manejar con cuidado y (con suerte) correctamente. Sin embargo, si bien el uso de serializadores es relativamente sencillo cuando solo hay un único recurso compartido (como una única cuenta bancaria), la programación concurrente puede resultar extremadamente difícil cuando hay múltiples recursos compartidos.

Para ilustrar una de las dificultades que pueden surgir, supongamos que queremos intercambiar los saldos de dos cuentas bancarias. Accedemos a cada cuenta para encontrar el saldo, calculamos la diferencia entre los saldos, retiramos esta diferencia de una cuenta y la depositamos en la otra. Podríamos implementar esto de la siguiente manera: [41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_421)

(define (intercambio cuenta1 cuenta2)   
  (let ((diferencia (- (cuenta1 'saldo)   
                       (cuenta2 'saldo)))   
    ((cuenta1 'retiro) diferencia)   
    ((cuenta2 'depósito) diferencia)))

Este procedimiento funciona bien cuando sólo un único proceso está intentando realizar el intercambio. Sin embargo, supongamos que Peter y Paul tienen acceso a las cuentas *a* 1, *a* 2 y *a* 3, y que Peter intercambia *a* 1 y *a* 2 mientras que Paul intercambia simultáneamente *a* 1 y *a* 3. Incluso con depósitos y retiros de cuentas serializados para cuentas individuales (como en el procedimiento de creación de cuenta que se muestra arriba en esta sección), el intercambio aún puede producir resultados incorrectos. Por ejemplo, Peter podría calcular la diferencia en los saldos de *a* 1 y *a* 2, pero luego Paul podría cambiar el saldo de *a* 1 antes de que Peter pueda completar el intercambio. [42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_422) Para un comportamiento correcto, debemos hacer que el procedimiento de intercambio bloquee cualquier otro acceso concurrente a las cuentas durante todo el tiempo del intercambio.

Una forma de lograr esto es usar los serializadores de ambas cuentas para serializar todo el procedimiento de intercambio . Para ello, organizaremos el acceso al serializador de una cuenta. Tenga en cuenta que estamos rompiendo deliberadamente la modularidad del objeto bank-account al exponer el serializador. La siguiente versión de make-account es idéntica a la versión original proporcionada en la sección  [3.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.1) , excepto que se proporciona un serializador para proteger la variable balance, y el serializador se exporta mediante el paso de mensajes:

(define (hacer-cuenta-y-serializador saldo)   
  (define (monto de retiro)   
    (if (>= monto de saldo)   
        (begin (set! saldo (- monto de saldo))   
               saldo)   
        "Fondos insuficientes"))   
  (define (monto de depósito)   
    (set! saldo (+ monto de saldo))   
    saldo)   
  (let ((serializador-saldo (hacer-serializador)))   
    (define (enviar m)   
      (cond ((eq? m 'retirar) retirar)   
            ((eq? m 'depositar) depositar)   
            ((eq? m 'saldo) saldo)   
            ((eq? m 'serializador) serializador-saldo)   
            (else (error "Solicitud desconocida -- HACER-CUENTA"   
                         m))))   
    enviar))

Podemos utilizar esto para realizar depósitos y retiros serializados. Sin embargo, a diferencia de nuestra cuenta serializada anterior, ahora es responsabilidad de cada usuario de los objetos de cuenta bancaria administrar explícitamente la serialización, por ejemplo, de la siguiente manera: [43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_423)

(define (cuenta de depósito importe)   
  (let ((s (cuenta 'serializador))   
        (d (cuenta 'depósito)))   
    ((s d) importe)))

Exportar el serializador de esta manera nos da la flexibilidad suficiente para implementar un programa de intercambio serializado. Simplemente serializamos el procedimiento de intercambio original con los serializadores de ambas cuentas:

(define (serializado-intercambio cuenta1 cuenta2)   
  (let ((serializador1 (cuenta1 'serializador))   
        (serializador2 (cuenta2 'serializador)))   
    ((serializador1 (serializador2 intercambio))   
     cuenta1   
     cuenta2)))

**Ejercicio 3.43.**   Suponga que los saldos de tres cuentas comienzan con $10, $20 y $30, y que se ejecutan múltiples procesos que intercambian los saldos de las cuentas. Argumente que si los procesos se ejecutan secuencialmente, después de cualquier número de intercambios simultáneos, los saldos de las cuentas deberían ser $10, $20 y $30 en algún orden. Dibuje un diagrama de tiempos como el de la figura  [3.29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.29) para mostrar cómo se puede violar esta condición si los intercambios se implementan utilizando la primera versión del programa de intercambio de cuentas de esta sección. Por otro lado, argumente que incluso con este programa de intercambio , se conservará la suma de los saldos de las cuentas. Dibuje un diagrama de tiempos para mostrar cómo se violaría incluso esta condición si no serializáramos las transacciones en cuentas individuales.

**Ejercicio 3.44.**  Consideremos el problema de transferir una cantidad de dinero de una cuenta a otra. Ben Bitdiddle afirma que esto se puede lograr con el siguiente procedimiento, incluso si hay varias personas transfiriendo dinero simultáneamente entre varias cuentas, utilizando cualquier mecanismo de cuenta que serialice las transacciones de depósito y retiro, por ejemplo, la versión de make-account en el texto anterior.

(define (transferencia de cuenta a cuenta importe)   
  ((de cuenta 'retiro) importe)   
  ((a cuenta 'depósito) importe))

Louis Reasoner sostiene que aquí hay un problema y que necesitamos utilizar un método más sofisticado, como el que se requiere para tratar el problema del intercambio. ¿Tiene razón Louis? Si no, ¿cuál es la diferencia esencial entre el problema de la transferencia y el problema del intercambio? (Debe suponer que el saldo en la cuenta de origen es al menos el monto .)

**Ejercicio 3.45.**   Louis Reasoner piensa que nuestro sistema de cuentas bancarias es innecesariamente complejo y propenso a errores ahora que los depósitos y retiros no se serializan automáticamente. Sugiere que make-account-and-serializer debería haber exportado el serializador (para su uso por parte de procedimientos como serialized-exchange ) además de (en lugar de) usarlo para serializar cuentas y depósitos como lo hizo make-account . Propone redefinir accounts de la siguiente manera:

(define (hacer-cuenta-y-serializador saldo)   
  (define (monto de retiro)   
    (if (>= monto de saldo)   
        (begin (set! saldo (- monto de saldo))   
               saldo)   
        "Fondos insuficientes"))   
  (define (monto de depósito)   
    (set! saldo (+ monto de saldo))   
    saldo)   
  (let ((serializador-saldo (hacer-serializador)))   
    (define (enviar m)   
      (cond ((eq? m 'retirar) (serializador-saldo retirar))   
            ((eq? m 'depositar) (serializador-saldo depositar))   
            ((eq? m 'saldo) saldo)   
            ((eq? m 'serializador) serializador-saldo)   
            (else (error "Solicitud desconocida -- HACER-CUENTA"   
                         m))))   
    enviar))

Luego, los depósitos se gestionan como en la cuenta original :

(define (monto de la cuenta de depósito)   
 ((monto de la cuenta 'depósito)))

Explique qué es lo que está mal en el razonamiento de Louis. En particular, considere qué sucede cuando se llama a serialized-exchange .

**[Implementando serializadores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_427)**

Implementamos serializadores en términos de un mecanismo de sincronización más primitivo llamado*mutex* . Un mutex es un objeto que admite dos operaciones: el mutex puede ser*adquirido* , y el mutex puede ser*liberado* . Una vez que se ha adquirido un mutex, no se pueden realizar otras operaciones de adquisición sobre ese mutex hasta que se libere el mutex. [44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_428) En nuestra implementación, cada serializador tiene un mutex asociado. Dado un procedimiento p , el serializador devuelve un procedimiento que adquiere el mutex, ejecuta p y luego libera el mutex. Esto garantiza que solo uno de los procedimientos producidos por el serializador pueda ejecutarse a la vez, que es precisamente la propiedad de serialización que necesitamos garantizar.

(define (make-serializer)   
  (let ((mutex (make-mutex)))   
    (lambda (p)   
      (define (serialized-p . args)   
        (mutex 'adquirir)   
        (let ((val (apply p args)))   
          (mutex 'liberar)   
          val))   
      serialized-p)))

El mutex es un objeto mutable (aquí usaremos una lista de un elemento, a la que nos referiremos como*celda* ) que puede contener el valor verdadero o falso. Cuando el valor es falso, el mutex está disponible para ser adquirido. Cuando el valor es verdadero, el mutex no está disponible y cualquier proceso que intente adquirir el mutex debe esperar.

Nuestro constructor de mutex make-mutex comienza inicializando el contenido de la celda en falso. Para adquirir el mutex, probamos la celda. Si el mutex está disponible, establecemos el contenido de la celda en verdadero y continuamos. De lo contrario, esperamos en un bucle, intentando adquirir una y otra vez, hasta que encontramos que el mutex está disponible. [45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_429) Para liberar el mutex, establecemos el contenido de la celda en falso.

(define (crear-mutex)   
  (let ((celda (lista falsa)))               
    (define (el-mutex m)   
      (cond ((eq? m 'adquirir)   
             (si (prueba-y-establece! celda)   
                 (el-mutex 'adquirir)))  *; reintentar*  
            ((eq? m 'liberar) (limpiar! celda))))   
    el-mutex))   
(define (limpiar! celda)   
  (establecer-auto! celda falsa))

¡Prueba y establece! prueba la celda y devuelve el resultado de la prueba. Además, si la prueba fue falsa, ¡prueba y establece! establece el contenido de la celda como verdadero antes de devolver falso. Podemos expresar este comportamiento como el siguiente procedimiento:

(define (prueba-y-establece! celda)   
  (si (celda del auto)   
      es verdadero   
      (comienza (establece-auto! celda es verdadero)   
             es falso)))

Sin embargo, esta implementación de test-and-set! no es suficiente tal como está. Hay una sutileza crucial aquí, que es el lugar esencial donde el control de concurrencia entra al sistema: la operación test-and-set! debe realizarse*atómicamente* . Es decir, debemos garantizar que, una vez que un proceso haya probado la celda y haya encontrado que es falsa, el contenido de la celda se establecerá como verdadero antes de que cualquier otro proceso pueda probar la celda. Si no ofrecemos esta garantía, el mutex puede fallar de una manera similar a la falla de la cuenta bancaria en la figura  [3.29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.29) . (Véase el ejercicio  [3.46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.46) .)

La implementación real de test-and-set! depende de los detalles de cómo nuestro sistema ejecuta procesos concurrentes. Por ejemplo, podríamos estar ejecutando procesos concurrentes en un procesador secuencial usando unmecanismo de división temporal que recorre los procesos en ciclos, permitiendo que cada proceso se ejecute durante un breve período de tiempo antes de interrumpirlo y pasar al siguiente proceso. En ese caso, test-and-set! puede funcionar desactivando la división temporal durante la prueba y la configuración. [46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_430) Alternativamente, las computadoras multiprocesadoras proporcionan instrucciones que admiten operaciones atómicas directamente en el hardware. [47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_431)

**Ejercicio 3.46.**   Supongamos que implementamos test-and-set! utilizando un procedimiento ordinario como el que se muestra en el texto, sin intentar que la operación sea atómica. Dibuje un diagrama de tiempos como el de la figura  [3.29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.29) para demostrar cómo la implementación del mutex puede fallar al permitir que dos procesos adquieran el mutex al mismo tiempo.

**Ejercicio 3.47.**  Un semáforo (de tamaño *n* ) es una generalización de un mutex. Al igual que un mutex, un semáforo admite operaciones de adquisición y liberación, pero es más general en el sentido de que hasta *n* procesos pueden adquirirlo simultáneamente. Los procesos adicionales que intenten adquirir el semáforo deben esperar a las operaciones de liberación. Dar implementaciones de semáforos

a. en términos de mutex

b. en términos de operaciones atómicas de prueba y ajuste .

**[Punto muerto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_434)**

Ahora que hemos visto cómo implementar serializadores, podemos ver que el intercambio de cuentas aún tiene un problema, incluso con el procedimiento de intercambio serializado anterior. Imagine que Peter intenta intercambiar *un* 1 con *un* 2 mientras que Paul intenta simultáneamente intercambiar *un* 2 con *un* 1. Suponga que el proceso de Peter llega al punto en el que ha ingresado a un procedimiento serializado que protege *un* 1 y, justo después de eso, el proceso de Paul ingresa a un procedimiento serializado que protege *un* 2. Ahora Peter no puede continuar (ingresar a un procedimiento serializado que protege *un* 2) hasta que Paul salga del procedimiento serializado que protege *un* 2. De manera similar, Paul no puede continuar hasta que Peter salga del procedimiento serializado que protege *un* 1. Cada proceso se detiene para siempre, esperando al otro. Esta situación se llama *interbloqueo* . El interbloqueo siempre es un peligro en los sistemas que brindan acceso simultáneo a múltiples recursos compartidos.

Una forma de evitar el bloqueo en esta situación es dar a cada cuenta un número de identificación único y reescribir el intercambio serializado de modo que un proceso siempre intente ingresar primero a un procedimiento que proteja la cuenta con el número más bajo. Aunque este método funciona bien para el problema del intercambio, existen otras situaciones que requieren técnicas más sofisticadas para evitar el bloqueo, o donde el bloqueo no se puede evitar en absoluto. (Véase los ejercicios  [3.48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.48) y  [3.49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.49) ). [48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_435)

**Ejercicio 3.48.**  Explique en detalle por qué el método de evitación de bloqueos descrito anteriormente (es decir, las cuentas están numeradas y cada proceso intenta adquirir primero la cuenta con el número más bajo) evita el bloqueo en el problema de intercambio. Reescriba serialized-exchange para incorporar esta idea. (También deberá modificar make-account para que cada cuenta se cree con un número, al que se pueda acceder enviando un mensaje apropiado).

**Ejercicio 3.49.**   Proponga un escenario en el que el mecanismo de prevención de bloqueos descrito anteriormente no funcione. (Pista: en el problema de intercambio, cada proceso sabe de antemano a qué cuentas necesitará acceder. Considere una situación en la que un proceso debe obtener acceso a algunos recursos compartidos antes de poder saber qué recursos compartidos adicionales necesitará).

**[Simultaneidad, tiempo y comunicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_438)**

Hemos visto cómo la programación de sistemas concurrentes requiere controlar el orden de los eventos cuando diferentes procesos acceden a un estado compartido, y hemos visto cómo lograr este control mediante un uso juicioso de serializadores. Pero los problemas de concurrencia son más profundos que esto, porque, desde un punto de vista fundamental, no siempre está claro qué se entiende por "estado compartido".

Mecanismos como test-and-set! requieren que los procesos examinen un indicador compartido global en momentos arbitrarios. Esto es problemático e ineficiente de implementar en procesadores modernos de alta velocidad, donde debido a técnicas de optimización como la segmentación y la memoria caché, el contenido de la memoria puede no estar en un estado consistente en todo momento. Por lo tanto, en los sistemas de multiprocesamiento contemporáneos, el paradigma del serializador está siendo reemplazado por nuevos enfoques para el control de la concurrencia. [49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_439)

Los aspectos problemáticos del estado compartido también surgen en sistemas distribuidos de gran tamaño. Por ejemplo, imaginemos un sistema bancario distribuido en el que las sucursales bancarias individuales mantienen valores locales para los saldos bancarios y los comparan periódicamente con los valores mantenidos por otras sucursales. En un sistema de este tipo, el valor del «saldo de la cuenta» sería indeterminado, excepto justo después de la sincronización. Si Peter deposita dinero en una cuenta que tiene en conjunto con Paul, ¿cuándo deberíamos decir que el saldo de la cuenta ha cambiado: cuando cambia el saldo de la sucursal local o no hasta después de la sincronización? Y si Paul accede a la cuenta desde una sucursal diferente, ¿cuáles son las restricciones razonables que se deben imponer al sistema bancario para que el comportamiento sea «correcto»? Lo único que podría importar para la corrección es el comportamiento observado por Peter y Paul individualmente y el «estado» de la cuenta inmediatamente después de la sincronización. Las preguntas sobre el saldo «real» de la cuenta o el orden de los eventos entre sincronizaciones pueden ser irrelevantes o carentes de sentido. [50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_440)

El fenómeno básico aquí es que sincronizar diferentes procesos, establecer un estado compartido o imponer un orden en los eventos requiere comunicación entre los procesos. En esencia, cualquier noción de tiempo en el control de concurrencia debe estar íntimamente ligada a la comunicación. [51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_441) Es intrigante que una conexión similar entre el tiempo y la comunicación también surja en elTeoría de la Relatividad, donde la velocidad de la luz (la señal más rápida que se puede utilizar para sincronizar eventos) es una constante fundamental que relaciona el tiempo y el espacio. Las complejidades que encontramos al tratar con el tiempo y el estado en nuestros modelos computacionales pueden, de hecho, reflejar una complejidad fundamental del universo físico.

[34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_405) La mayoría de los procesadores reales en realidad ejecutan unas pocas operaciones a la vez, siguiendo una estrategia llamada*canalización* . Aunque esta técnica mejora en gran medida la utilización eficaz del hardware, se utiliza únicamente para acelerar la ejecución de un flujo de instrucciones secuenciales, conservando al mismo tiempo el comportamiento del programa secuencial.

[35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_406) Para citar algunos grafitis vistos en Cambridgemuro de construcción: ``El tiempo es un dispositivo que se inventó para evitar que todo suceda a la vez.''

[36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_407) Un fallo aún peor para este sistema podría ocurrir si las dosde configuraciónintentan cambiar el equilibrio simultáneamente, en cuyo caso los datos reales que aparecen en la memoria podrían terminar siendo una combinación aleatoria de la información que escriben los dos procesos. La mayoría de las computadoras tienen interbloqueos en las operaciones primitivas de escritura en memoria, que protegen contra dicho acceso simultáneo. Sin embargo, incluso este tipo de protección aparentemente simple plantea desafíos de implementación en el diseño de computadoras multiprocesamiento, donde se requieren operaciones complejas de escritura en memoria.*Se requieren protocolos de coherencia de caché* para garantizar que los distintos procesadores mantendrán una visión consistente del contenido de la memoria, a pesar del hecho de que los datos pueden replicarse ("almacenarse en caché") entre los diferentes procesadores para aumentar la velocidad de acceso a la memoria.

[37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_409) El programa factorial de la sección [3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.3)ilustra esto para un solo proceso secuencial.

[38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_410) Las columnas muestran el contenido de la billetera de Peter, la cuenta conjunta (en el Banco 1), la billetera de Paul y la cuenta privada de Paul (en el Banco 2), antes y después de cada retiro (W) y depósito (D). Peter retira $10 del Banco 1; Paul deposita $5 en el Banco 2 y luego retira $25 del Banco 1.

[39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_411) Una forma más formal de expresar esta idea es decir que los programas concurrentes son inherentemente*no deterministas* . Es decir, no se describen mediante funciones univaluadas, sino mediante funciones cuyos resultados son conjuntos de valores posibles. En la sección  [4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3) estudiaremos un lenguaje para expresar cálculos no deterministas.

[40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_415) La ejecución paralelano forma parte del Scheme estándar, pero se puede implementar en el Scheme MIT. En nuestra implementación, los nuevos procesos simultáneos también se ejecutan simultáneamente con el proceso Scheme original. Además, en nuestra implementación, el valor devuelto porla ejecución paralelaes un objeto de control especial que se puede utilizar para detener los procesos recién creados.

Hemos simplificadoel intercambioaprovechando el hecho de que nuestrode depósito [acepta montos negativos. ( ¡](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_421) Esto es un grave error en nuestro sistema bancario!)

[42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_422) Si los saldos de las cuentas comienzan con $10, $20 y $30, luego de cualquier cantidad de intercambios simultáneos, los saldos deberían seguir siendo $10, $20 y $30 en algún orden. La serialización de los depósitos en cuentas individuales no es suficiente para garantizar esto. Vea el ejercicio [3.43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.43).

[El](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_423) ejercicio [3.45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.45)investiga por qué los depósitos y retiros ya no se serializan automáticamente por la cuenta.

[44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_428) El término "mutex" es una abreviatura de*Exclusión mutua* . El problema general de organizar un mecanismo que permita que los procesos concurrentes compartan recursos de forma segura se denomina problema de exclusión mutua. Nuestro mutex es una variante simple del*mecanismo de semáforo* (ver ejercicio  [3.47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.47) ), que se introdujo en el«EL» Sistema Multiprogramación desarrollado en laUniversidad Tecnológica de Eindhoven y nombrada así por las iniciales de la universidad en holandés (Dijkstra 1968a). Las operaciones de adquisición y liberación se denominaban originalmenteP y V, de las palabras holandesas *passeren* (pasar) y *vrijgeven* (liberar), en referencia a los semáforos utilizados en los sistemas ferroviarios. La exposición clásica de Dijkstra (1968b) fue una de las primeras en presentar claramente los problemas del control de concurrencia y mostró cómo utilizar semáforos para manejar una variedad de problemas de concurrencia.

[45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_429) En la mayoría de los sistemas operativos de tiempo compartido, los procesos que sonbloqueado por un mutexNo se pierde tiempo en espera, como se indica más arriba. En lugar de ello, el sistema programa otro proceso para que se ejecute mientras el primero espera, y el proceso bloqueado se activa cuando el mutex está disponible.

[En el esquema MIT para un solo procesador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_430) , que utiliza un modelo de segmentación temporal,¡test-and-set!se puede implementar de la siguiente manera:

(define (prueba-y-establece! celda)   
  (sin-interrupciones   
   (lambda ()   
     (si (auto celda)   
         verdadero   
         (comienza (establece-auto! celda verdadero)   
                falso)))))

Without-interrupts deshabilita las interrupciones de división de tiempo mientras se ejecuta su argumento de procedimiento.

Hay muchas variantes de [este](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_431)instrucciones, incluidas las de prueba y configuración, prueba y borrado, intercambio, comparación e intercambio, reserva de carga y almacenamiento condicional, cuyo diseño debe coincidir cuidadosamente con la interfaz de memoria del procesador de la máquina. Una cuestión que surge aquí es determinar qué sucede si dos procesos intentan adquirir el mismo recurso exactamente al mismo tiempo utilizando una instrucción de este tipo. Esto requiere algún mecanismo para tomar una decisión sobre qué proceso obtiene el control. Tal mecanismo se denomina*árbitro* . Los árbitros suelen reducirse a algún tipo de dispositivo de hardware. Desafortunadamente, es posible demostrar que no se puede construir físicamente un árbitro justo que funcione el 100% del tiempo a menos que se le permita al árbitro un tiempo arbitrariamente largo para tomar su decisión. El fenómeno fundamental aquí fue observado originalmente por el filósofo francés del siglo XIVJean Buridan en su comentario sobre*El De caelo* de Aristóteles . Buridan argumentó que una solución perfectamente racionalUn perro colocado entre dos fuentes de alimento igualmente atractivas morirá de hambre, porque es incapaz de decidir a cuál acudir primero.

[48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_435) La técnica general para evitar el bloqueo mediante la numeración de losLos recursos compartidos y su adquisición en orden se deben aHavender (1968). Las situaciones en las que no se puede evitar un bloqueo requieren métodos *de recuperación de bloqueos* , que implican que los procesos ``salgan'' del estado de bloqueo y vuelvan a intentarlo. Los mecanismos de recuperación de bloqueos se utilizan ampliamente en los sistemas de gestión de bases de datos, un tema que se trata en detalle en Gray y Reuter 1993.

[49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_439) Una de esas alternativas a la serialización se denomina*sincronización de barrera* . El programador permite que los procesos concurrentes se ejecuten como quieran, pero establece ciertos puntos de sincronización ("barreras") a través de los cuales ningún proceso puede avanzar hasta que todos los procesos hayan alcanzado la barrera. Los procesadores modernos proporcionan instrucciones de máquina que permiten a los programadores establecer puntos de sincronización en lugares donde se requiere consistencia.PowerPC *TM ,* por ejemplo, incluye para este propósito dos instrucciones llamadasSINCRONIZAR yEIEIO (Ejecución obligatoria en orden de entrada/salida).

[50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_440) Puede parecer un punto de vista extraño, pero hayLos sistemas que funcionan de esta manera. Por ejemplo, los cargos internacionales a cuentas de tarjetas de crédito normalmente se liquidan por país y los cargos realizados en diferentes países se concilian periódicamente. Por lo tanto, el saldo de la cuenta puede ser diferente en distintos países.

[51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_441) Para los sistemas distribuidos, esta perspectiva fue seguida porLamport (1978), quien mostró cómo utilizar la comunicación para establecer “relojes globales” que pueden emplearse para establecer ordenamientos de eventos en sistemas distribuidos.

[**3.5 Flujos**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_3.5)

Hemos adquirido una buena comprensión de la asignación como herramienta de modelado, así como una apreciación de los complejos problemas que plantea la asignación. Es hora de preguntarnos si podríamos haber hecho las cosas de otra manera, para evitar algunos de estos problemas. En esta sección, exploramos un enfoque alternativo para modelar el estado, basado en estructuras de datos llamadas *streams* . Como veremos, los streams pueden mitigar parte de la complejidad del modelado del estado.

Retrocedamos un poco y revisemos de dónde proviene esta complejidad. En un intento de modelar fenómenos del mundo real, tomamos algunas decisiones aparentemente razonables: modelamos objetos del mundo real con estado local mediante objetos computacionales con variables locales. Identificamos la variación temporal en el mundo real con la variación temporal en la computadora. Implementamos la variación temporal de los estados de los objetos del modelo en la computadora con asignaciones a las variables locales de los objetos del modelo.

¿Existe otro enfoque? ¿Podemos evitar identificar el tiempo en la computadora con el tiempo en el mundo modelado? ¿Debemos hacer que el modelo cambie con el tiempo para modelar fenómenos en un mundo cambiante? Pensemos en la cuestión en términos de funciones matemáticas. Podemos describir el comportamiento variable en el tiempo de una cantidad *x* como una función del tiempo *x* ( *t* ). Si nos concentramos en *x* instante a instante, pensamos en ella como una cantidad cambiante. Sin embargo, si nos concentramos en toda la historia temporal de los valores, no enfatizamos el cambio: la función en sí misma no cambia. [52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_442)

Si el tiempo se mide en pasos discretos, entonces podemos modelar una función de tiempo como una secuencia (posiblemente infinita). En esta sección, veremos cómo modelar el cambio en términos de secuencias que representan las historias temporales de los sistemas que se están modelando. Para lograr esto, presentamos nuevas estructuras de datos llamadas *streams* . Desde un punto de vista abstracto, un stream es simplemente una secuencia. Sin embargo, descubriremos que la implementación directa de streams como listas (como en la sección  [2.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.1) ) no revela completamente el poder del procesamiento de streams. Como alternativa, presentamos la técnica de*evaluación retardada* , que nos permite representar secuencias muy grandes (incluso infinitas) como flujos.

El procesamiento de flujos nos permite modelar sistemas que tienen estado sin utilizar nunca asignaciones ni datos mutables. Esto tiene importantes implicaciones, tanto teóricas como prácticas, porque podemos construir modelos que eviten los inconvenientes inherentes a la introducción de asignaciones. Por otro lado, el marco de flujos plantea sus propias dificultades, y la cuestión de qué técnica de modelado conduce a sistemas más modulares y de más fácil mantenimiento sigue abierta.

**[3.5.1 Las transmisiones son listas retrasadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.5.1)**

Como vimos en la sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) , las secuencias pueden servir como interfaces estándar para combinar módulos de programas. Formulamos abstracciones potentes para manipular secuencias, como map , filter y recover , que capturan una amplia variedad de operaciones de una manera que es a la vez sucinta y elegante.

Desafortunadamente, si representamos secuencias como listas, esta elegancia se consigue a costa de una grave ineficiencia en lo que respecta tanto al tiempo como al espacio que requieren nuestros cálculos. Cuando representamos manipulaciones de secuencias como transformaciones de listas, nuestros programas deben construir y copiar estructuras de datos (que pueden ser enormes) en cada paso de un proceso.

Para ver por qué esto es cierto, comparemos dos programas para calcular la suma de todos los números primos en un intervalo. El primer programa está escrito en estilo iterativo estándar: [53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_443)

(definir (suma-primos a b)   
  (definir (iter contar acumular)   
    (cond ((> contar b) acumular)   
          ((primo? contar) (iter (+ contar 1) (+ contar acumular)))   
          (de lo contrario (iter (+ contar 1) acumular))))   
  (iter a 0))

El segundo programa realiza el mismo cálculo utilizando las operaciones de secuencia de la sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) :

(definir (suma-primos a b)   
  (acumular +   
              0   
              (filtrar primo? (enumerar-intervalo a b))))

Para realizar el cálculo, el primer programa necesita almacenar únicamente la suma que se acumula. En cambio, el filtro del segundo programa no puede realizar ninguna prueba hasta que enumerate-interval haya construido una lista completa de los números del intervalo. El filtro genera otra lista, que a su vez se pasa a recover antes de ser colapsada para formar una suma. El primer programa no necesita un almacenamiento intermedio tan grande, ya que podemos pensar que enumera el intervalo de forma incremental, añadiendo cada número primo a la suma a medida que se genera.

La ineficiencia en el uso de listas se hace dolorosamente evidente si usamos el paradigma de secuencia para calcular el segundo primo en el intervalo de 10.000 a 1.000.000 evaluando la expresión

(coche (cdr (filtro primo?   
                  (enumerar-intervalo 10000 1000000))))

Esta expresión encuentra el segundo primo, pero la sobrecarga computacional es escandalosa. Construimos una lista de casi un millón de números enteros, filtramos esta lista probando cada elemento para determinar su primalidad y luego ignoramos casi todo el resultado. En un estilo de programación más tradicional, intercalaríamos la enumeración y el filtrado, y nos detendríamos cuando llegáramos al segundo primo.

Los streams son una idea inteligente que permite utilizar manipulaciones de secuencias sin incurrir en los costos de manipular secuencias como listas. Con los streams podemos lograr lo mejor de ambos mundos: podemos formular programas elegantemente como manipulaciones de secuencias, al mismo tiempo que logramos la eficiencia del cálculo incremental. La idea básica es organizar la construcción de un stream solo parcialmente y pasar la construcción parcial al programa que consume el stream. Si el consumidor intenta acceder a una parte del stream que aún no se ha construido, el stream construirá automáticamente solo lo suficiente de sí mismo para producir la parte requerida, preservando así la ilusión de que el stream completo existe. En otras palabras, aunque escribiremos programas como si estuviéramos procesando secuencias completas, diseñamos nuestra implementación de stream para intercalar de manera automática y transparente la construcción del stream con su uso.

A primera vista, los flujos son simplemente listas con distintos nombres para los procedimientos que los manipulan. Hay un constructor,cons-stream y dos selectores,carro de corriente ystream-cdr , que satisfacen las restricciones



Hay un objeto distinguible,the-empty-stream , que no puede ser el resultado de ninguna operación cons-stream , y que puede identificarse con el predicadostream-null? . [54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_444) Por lo tanto, podemos crear y utilizar streams, de la misma manera que podemos crear y utilizar listas, para representar datos agregados dispuestos en una secuencia. En particular, podemos crear análogos de stream de las operaciones de lista del capítulo 2, como list-ref , map y for-each : [55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_445)

(define (stream-ref s n)   
  (si (= n 0)   
      (stream-car s)   
      (stream-ref (stream-cdr s) (- n 1))))  
(define (stream-map proc s)   
  (si (stream-null? s)   
      el-flujo-vacío   
      (cons-stream (proc (stream-car s))   
                   (stream-map proc (stream-cdr s)))))  
(define (stream-for-each proc s)   
  (if (stream-null? s)   
      'done   
      (begin (proc (stream-car s))   
             (stream-for-each proc (stream-cdr s)))))

Stream-for-each es útil para visualizar transmisiones:

(define (flujo de visualización s)   
  (flujo para cada línea de visualización s))  
  
(define (mostrar-linea x)   
  (nueva línea)   
  (mostrar x))

Para que la implementación del stream intercale de manera automática y transparente la construcción de un stream con su uso, haremos que el cdr de un stream se evalúe cuando se accede a él mediante el procedimiento stream-cdr en lugar de cuando el stream se construye mediante cons-stream . Esta elección de implementación recuerda a nuestra discusión de números racionales en la sección  [2.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.2) , donde vimos que podemos elegir implementar números racionales de modo que la reducción del numerador y el denominador a los términos más bajos se realice en el momento de la construcción o en el momento de la selección. Las dos implementaciones de números racionales producen la misma abstracción de datos, pero la elección tiene un efecto en la eficiencia. Existe una relación similar entre streams y listas ordinarias. Como abstracción de datos, los streams son lo mismo que las listas. La diferencia es el momento en el que se evalúan los elementos. Con listas ordinarias, tanto el car como el cdr se evalúan en el momento de la construcción. Con streams, el cdr se evalúa en el momento de la selección.

Nuestra implementación de streams se basará en una forma especial llamada delay . Evaluar (delay < *exp* >) no evalúa la expresión < *exp* >, sino que devuelve un llamado*objeto retrasado* , que podemos considerar como una ``promesa'' de evaluar < *exp* > en algún momento futuro. Como complemento de delay , existe un procedimiento llamadofuerza que toma un objeto retrasado como argumento y realiza la evaluación; en efecto, obliga al retraso a cumplir su promesa. Veremos más adelante cómo se pueden implementar el retraso y la fuerza , pero primero usémoslos para construir secuencias.

Cons-stream es una forma especial definida de modo que

(cons-stream < *a* > < *b* >)

es equivalente a

(cons < *a* > (retraso < *b* >))

Esto significa que construiremos secuencias utilizando pares. Sin embargo, en lugar de colocar el valor del resto de la secuencia en el CDR del par, pondremos allí una promesa de calcular el resto si alguna vez se solicita. Stream-car y stream-cdr ahora se pueden definir como procedimientos:

(definir (flujo-flujo de carro) (flujo de carro))  
  
(definir (flujo-cdr flujo) (fuerza (flujo cdr)))

Stream-car selecciona el auto del par; stream-cdr selecciona el cdr del par y evalúa la expresión retrasada que se encuentra allí para obtener el resto del flujo. [56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_446)

[**La implementación del stream en acción**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_447)

Para ver cómo se comporta esta implementación, analicemos el cálculo primo "escandaloso" que vimos arriba, reformulado en términos de flujos:

(flujo-automóvil   
 (flujo-cdr   
  (flujo-filtro primo?   
                 (flujo-enumerar-intervalo 10000 1000000))))

Veremos que efectivamente funciona eficientemente.

Comenzamos llamando a stream-enumerate-interval con los argumentos 10,000 y 1,000,000. Stream-enumerate-interval es el análogo de stream de enumerate-interval (sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) ):

(define (flujo-enumerar-intervalo bajo alto)   
  (si (> bajo alto)   
      el-flujo-vacío   
      (cons-flujo   
       bajo   
       (flujo-enumerar-intervalo (+ bajo 1) alto))))

y por lo tanto el resultado devuelto por stream-enumerate-interval , formado por cons-stream , es [57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_448)

(cons 10000   
      (retraso (intervalo de enumeración de flujo 10001 1000000)))

Es decir, stream-enumerate-interval devuelve un flujo representado como un par cuyo car es 10.000 y cuyo cdr es una promesa de enumerar más del intervalo si así se solicita. Este flujo ahora se filtra para los primos, utilizando el análogo de flujo del procedimiento de filtro (sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) ):

(define (filtro-flujo pred flujo)   
  (cond ((flujo-nulo? flujo) el-flujo-vacío)   
        ((pred (flujo-car flujo))   
         (cons-flujo (flujo-car flujo)   
                      (filtro-flujo pred   
                                     (flujo-cdr flujo))))   
        (else (filtro-flujo pred (flujo-cdr flujo)))))

Stream-filter prueba el stream-car del flujo (el car del par, que es 10 000). Como no es primo, stream-filter examina el stream-cdr de su flujo de entrada. La llamada a stream-cdr fuerza la evaluación del stream-enumerate-interval retrasado , que ahora devuelve

(cons 10001   
      (retardo (intervalo de enumeración de flujo 10002 1000000)))

Stream-filter ahora observa el stream-car de este flujo, 10,001, ve que este tampoco es primo, fuerza otro stream-cdr , y así sucesivamente, hasta que stream-enumerate-interval produce el primo 10,007, con lo cual stream-filter , según su definición, retorna

(cons-stream (stream-car stream)   
             (stream-filter pred (stream-cdr stream)))

que en este caso es

(contras 10007   
      (retardo   
        (filtro de flujo   
         principal?   
         (contras 10008   
               (retardo   
                 (intervalo de enumeración de flujo 10009   
                                            1000000))))))

Este resultado ahora se pasa a stream-cdr en nuestra expresión original. Esto fuerza el retardado stream-filter , que a su vez sigue forzando el retardado stream-enumerate-interval hasta que encuentra el próximo primo, que es 10,009. Finalmente, el resultado pasado a stream-car en nuestra expresión original es

(cons 10009   
      (retardo   
        (filtro de flujo   
         principal?   
         (cons 10010   
               (retardo   
                 (intervalo de enumeración de flujo 10011   
                                            1000000))))))

Stream-car devuelve 10.009 y el cálculo está completo. Se probaron únicamente tantos números enteros para determinar su primalidad como fueron necesarios para encontrar el segundo primo, y el intervalo se enumeró solo hasta donde fue necesario para alimentar el filtro de primos.

En general, podemos pensar en la evaluación retrasada comoProgramación "basada en la demanda", en la que cada etapa del proceso de flujo se activa sólo lo suficiente para satisfacer la etapa siguiente. Lo que hemos hecho esDesacoplamos el orden real de los eventos en el cálculo de la estructura aparente de nuestros procedimientos. Escribimos procedimientos como si los flujos existieran ``todos a la vez'' cuando, en realidad, el cálculo se realiza de forma incremental, como en los estilos de programación tradicionales.

**[Implementación de retraso y fuerza](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_449)**

Aunque delay y force pueden parecer operaciones misteriosas, su implementación es bastante sencilla. Delay debe empaquetar una expresión para que pueda evaluarse más tarde cuando se lo solicite, y podemos lograr esto simplemente tratando la expresión como el cuerpo de un procedimiento. Delay puede ser una forma especial tal que

(retraso < *exp* >)

¿Es azúcar sintáctico para?

(lambda() *<exp>* )

Force simplemente llama al procedimiento (sin argumentos) producido por delay , por lo que podemos implementar force como un procedimiento:

(define (fuerza objeto-retardado)   
  (objeto-retardado))

Esta implementación es suficiente para que delay y force funcionen como se anuncia, pero hay una optimización importante que podemos incluir. En muchas aplicaciones, terminamos forzando el mismo objeto retrasado muchas veces. Esto puede llevar a una ineficiencia grave en programas recursivos que involucran flujos. (Vea el ejercicio  [3.57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.57) ). La solución es construir objetos retrasados ​​de modo que la primera vez que se los fuerce, almacenen el valor que se calcula. Los forzamientos posteriores simplemente devolverán el valor almacenado sin repetir el cálculo. En otras palabras, implementamos delay como un procedimiento memorizado de propósito especial similar al descrito en el ejercicio  [3.27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.27) . Una forma de lograr esto es usar el siguiente procedimiento, que toma como argumento un procedimiento (sin argumentos) y devuelve una versión memorizada del procedimiento. La primera vez que se ejecuta el procedimiento memorizado, guarda el resultado calculado. En evaluaciones posteriores, simplemente devuelve el resultado.

(define (memo-proc proc)   
  (let ((¿ya se ejecutó? falso) (resultado falso))   
    (lambda ()   
      (if (¿no ya se ejecutó?)   
          (begin (¡establecer! resultado (proc))   
                 (¡establecer! ¿ya se ejecutó? verdadero)   
                 resultado)   
          resultado))))

El retraso se define entonces de modo que (retraso < *exp* >) sea equivalente a

(memo-proc(lambda() <exp> *)* )

y la fuerza es como se definió anteriormente. [58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_450)

**Ejercicio 3.50.**   Complete la siguiente definición, que generaliza stream-map para permitir procedimientos que toman múltiples argumentos, de manera análoga a map en la sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) , nota al pie  [12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#footnote_Temp_166) .

(define (mapa-de-flujo proc . argstreams)   
  (si (< *??* > (auto argstreams))   
      el-flujo-vacío   
      (< *??* >   
       (aplicar proc (mapa < *??* > argstreams))   
       (aplicar mapa-de-flujo   
              (cons proc (mapa < *??* > argstreams))))))

**Ejercicio 3.51.**  Para analizar más de cerca la evaluación retrasada, utilizaremos el siguiente procedimiento, que simplemente devuelve su argumento después de imprimirlo:

(definir (mostrar x)   
  (mostrar-linea x)   
  x)

¿Qué imprime el intérprete en respuesta a la evaluación de cada expresión en la siguiente secuencia? [59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_453)

(define x (stream-map show (stream-enumerate-interval 0 10)))   
(stream-ref x 5)   
(stream-ref x 7)

**Ejercicio 3.52.**  Considere la secuencia de expresiones

(definir suma 0)   
(definir (acum x)   
  (¡establecer! suma (+ x suma))   
  suma)   
(definir seq (mapa-flujo acum (intervalo-enumeración-flujo 1 20)))   
(definir y (filtro-flujo par? seq))   
(definir z (filtro-flujo (lambda (x) (= (resto x 5) 0))   
                         seq))   
(ref-flujo y 7)   
(mostrar-flujo z)

¿Cuál es el valor de suma después de evaluar cada una de las expresiones anteriores? ¿Cuál es la respuesta impresa para evaluar las expresiones stream-ref y display-stream ? ¿Diferirían estas respuestas si hubiéramos implementado (delay < *exp* >) simplemente como (lambda () < *exp* >) sin usar la optimización proporcionada por memo-proc ? Explique.

**[3.5.2 Flujos infinitos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.5.2)**

Hemos visto cómo sustentar la ilusión de manipular secuencias como entidades completas, aunque en realidad solo calculamos la parte de la secuencia a la que necesitamos acceder. Podemos aprovechar esta técnica para representar secuencias de manera eficiente como secuencias, incluso si las secuencias son muy largas. Lo que es más sorprendente es que podemos usar secuencias para representar secuencias que son infinitamente largas. Por ejemplo, considere la siguiente definición de la secuencia de números enteros positivos:

(define (enteros que comienzan desde n)   
  (cons-stream n (enteros que comienzan desde (+ n 1))))  
  
(define números enteros (enteros que comienzan desde 1))

Esto tiene sentido porque los números enteros serán un par cuyo car es 1 y cuyo cdr es una promesa de producir los números enteros que comienzan con 2. Este es un flujo infinitamente largo, pero en cualquier momento dado podemos examinar solo una porción finita de él. Por lo tanto, nuestros programas nunca sabrán que el flujo infinito completo no está allí.

Usando números enteros podemos definir otros flujos infinitos, como el flujo de números enteros que no son divisibles por 7:

(define (divisible? x y) (= (resto x y) 0))   
(define no-sietes   
  (filtro-de-flujo (lambda (x) (no (divisible? x 7)))   
                 enteros))

Luego podemos encontrar números enteros no divisibles por 7 simplemente accediendo a los elementos de este flujo:

(stream-ref no-sietes 100)   
*117*

En analogía con los números enteros , podemos definir el flujo infinito de números de Fibonacci:

(define (fibgen a b)   
  (cons-stream a (fibgen b (+ a b))))  
(define mentiras (fibgen 0 1))

Fibs es un par cuyo auto es 0 y cuyo cdr es una promesa a evaluar (fibgen 1 1) . Cuando evaluamos este retraso (fibgen 1 1) , producirá un par cuyo auto es 1 y cuyo cdr es una promesa a evaluar (fibgen 1 2) , y así sucesivamente.

Para ver un flujo infinito más emocionante, podemos generalizar el ejemplo de los sietes para construir el flujo infinito de números primos, utilizando un método conocido como*criba de Eratóstenes* . [60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_455) Empezamos con los números enteros que empiezan por 2, que es el primer primo. Para obtener el resto de los primos, empezamos filtrando los múltiplos de 2 del resto de los números enteros. Esto deja un flujo que empieza por 3, que es el siguiente primo. Ahora filtramos los múltiplos de 3 del resto de este flujo. Esto deja un flujo que empieza por 5, que es el siguiente primo, y así sucesivamente. En otras palabras, construimos los primos mediante un proceso de cribado, descrito de la siguiente manera: Para cribar un flujo S , formamos un flujo cuyo primer elemento sea el primer elemento de S y cuyo resto se obtenga filtrando todos los múltiplos del primer elemento de S del resto de S y cribando el resultado. Este proceso se describe fácilmente en términos de operaciones de flujo:

(define (flujo de tamiz)   
  (flujo de cons-flujo   
   (flujo-carro)   
   (tamiz (filtro-flujo   
           (lambda (x)   
             (no (divisible? x (flujo-carro))))   
           (flujo-cdr flujo)))))  
  
(definir primos (criba (enteros-comenzando-desde-2)))

Ahora, para encontrar un primo particular, sólo necesitamos preguntarlo:

(ref. de corriente primos 50)   
*233*

Es interesante contemplar el sistema de procesamiento de señales establecido por sieve , que se muestra en la"Diagrama de Henderson" en la figura  [3.31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.31) . [61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_456) El flujo de entrada se introduce en un "cons er " que separa el primer elemento del flujo del resto del flujo. El primer elemento se utiliza para construir un filtro de divisibilidad, a través del cual pasa el resto, y la salida del filtro se introduce en otra caja de tamiz. Luego, el primer elemento original se convierte en la salida del tamiz interno para formar el flujo de salida. Por lo tanto, no solo el flujo es infinito, sino que el procesador de señales también es infinito, porque el tamiz contiene un tamiz dentro de él.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.31:**   El tamiz principal visto como un sistema de procesamiento de señales. |

**[Definición implícita de flujos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_457)**

Los flujos de números enteros y fibs anteriores se definieron especificando procedimientos de "generación" que calculan explícitamente los elementos del flujo uno por uno. Una forma alternativa de especificar flujos es aprovechar la evaluación retrasada para definir flujos de manera implícita. Por ejemplo, la siguiente expresión define el flujo ones como un flujo infinito de unos:

(define unos (cons-stream 1 unos))

Esto funciona de forma muy similar a la definición de un procedimiento recursivo: ones es un par cuyo car es 1 y cuyo cdr es una promesa de evaluar ones . Al evaluar el cdr volvemos a obtener un 1 y una promesa de evaluar ones , y así sucesivamente.

Podemos hacer cosas más interesantes manipulando flujos con operaciones como add-streams , que produce la suma elemento por elemento de dos flujos dados: [62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_458)

(define (add-streams s1 s2)   
  (mapa-de-flujos + s1 s2))

Ahora podemos definir los números enteros de la siguiente manera:

(define números enteros (cons-stream 1 (add-streams unos números enteros)))

Esto define a los números enteros como un flujo cuyo primer elemento es 1 y el resto es la suma de unos y números enteros . Por lo tanto, el segundo elemento de los números enteros es 1 más el primer elemento de los números enteros , o 2; el tercer elemento de los números enteros es 1 más el segundo elemento de los números enteros , o 3; y así sucesivamente. Esta definición funciona porque, en cualquier punto, se ha generado una cantidad suficiente del flujo de números enteros para que podamos volver a introducirlo en la definición para producir el siguiente número entero.

Podemos definir los números de Fibonacci en el mismo estilo:

(definir fibs   
  (cons-stream 0   
               (cons-stream 1   
                            (add-streams (stream-cdr fibs)   
                                         fibs))))

Esta definición dice que fibs es un flujo que comienza con 0 y 1, de modo que el resto del flujo se puede generar agregando fibs a sí mismo desplazado un lugar:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | ... = (fibras de stream-cdr) |
|  |  | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | ... = mentiras |
| 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | ... = mentiras |
|  |  |  |  |  |  |  |  |  |  |  |

El flujo de escala es otro procedimiento útil para formular dichas definiciones de flujo. Este multiplica cada elemento de un flujo por una constante dada:

(define (flujo de escala factor de flujo)   
  (mapa de flujo (lambda (x) (\* factor x)) flujo))

Por ejemplo,

(definir doble (cons-stream 1 (scale-stream doble 2)))

produce la corriente de potencias de 2: 1, 2, 4, 8, 16, 32, ... .

Se puede dar una definición alternativa de la secuencia de números primos comenzando con los números enteros y filtrándolos mediante la prueba de primalidad. Necesitaremos el primer número primo, 2, para comenzar:

(definir primos   
  (cons-stream   
   2   
   (filtro-stream primo? (enteros-empezando-desde-3))))

Esta definición no es tan sencilla como parece, porque probaremos si un número *n* es primo verificando si *n* es divisible por un primo (no por cualquier entero) menor o igual a *n* :

(define (prime? n)   
  (define (iter ps)   
    (cond ((> (square (stream-car ps)) n) true)   
          ((divisible? n (stream-car ps)) false)   
          (else (iter (stream-cdr ps)))))   
  (iter primos))

Esta es una definición recursiva, ya que primes se define en términos del predicado prime?, que a su vez utiliza el flujo primes . La razón por la que este procedimiento funciona es que, en cualquier punto, se ha generado suficiente del flujo primes para probar la primalidad de los números que necesitamos verificar a continuación. Es decir, para cada *n* que probamos para la primalidad, o bien *n* no es primo (en cuyo caso ya se ha generado un primo que lo divide) o bien *n* es primo (en cuyo caso ya se ha generado un primo, es decir, un primo menor que *n* , que es mayor que *n* ). [63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_459)

**Ejercicio 3.53.**   Sin ejecutar el programa, describa los elementos del flujo definido por

(define s (cons-stream 1 (add-streams s s)))

**Ejercicio 3.54.**   Definir un procedimientomul-streams , análogo a add-streams , que produce el producto elemento por elemento de sus dos flujos de entrada. Utilícelo junto con el flujo de números enteros para completar la siguiente definición del flujo cuyo elemento *n* º (contando desde 0) es factorial *n* + 1:

(define factoriales (cons-stream 1 (mul-streams < *??* > < *??* >)))

**Ejercicio 3.55.**   Definir un procedimientosumas parciales que toman como argumento un flujo *S* y devuelven el flujo cuyos elementos son *S* 0 , *S* 0 + *S* 1 , *S* 0 + *S* 1 + *S* 2 , ... . Por ejemplo, (sumas parciales de enteros) debe ser el flujo 1, 3, 6, 10, 15, ... .

**Ejercicio 3.56.**   Un problema famoso, planteado por primera vez porR. Hamming, consiste en enumerar, en orden ascendente y sin repeticiones, todos los números enteros positivos que no tengan otros factores primos que 2, 3 o 5. Una forma obvia de hacerlo es simplemente comprobar cada número entero por turno para ver si tiene otros factores que 2, 3 y 5. Pero esto es muy ineficiente, ya que, a medida que los números enteros se hacen más grandes, cada vez son menos los que cumplen el requisito. Como alternativa, llamemos S a la secuencia de números requerida y observemos los siguientes hechos al respecto.

* S comienza con 1.
* Los elementos de (flujo de escala S 2) también son elementos de S .
* Lo mismo es cierto para (flujo de escala S 3) y (flujo de escala 5 S) .
* Estos son todos los elementos de S.

Ahora solo nos queda combinar elementos de estas fuentes. Para ello definimos un procedimiento merge que combina dos flujos ordenados en un flujo de resultados ordenado, eliminando repeticiones:

(define (fusionar s1 s2)   
  (cond ((flujo-nulo? s1) s2)   
        ((flujo-nulo? s2) s1)   
        (de lo contrario   
         (let ((s1car (flujo-car s1))   
               (s2car (flujo-car s2)))   
           (cond ((< s1car s2car)   
                  (cons-flujo s1car (fusionar (flujo-cdr s1) s2)))   
                 ((> s1car s2car)   
                  (cons-flujo s2car (fusionar s1 (flujo-cdr s2))))   
                 (de lo contrario   
                  (cons-flujo s1car   
                               (fusionar (flujo-cdr s1)   
                                      (flujo-cdr s2))))))))

Luego, el flujo requerido se puede construir con merge , de la siguiente manera:

(define S (cons-stream 1 (merge < *??* > < *??* >)))

Complete las expresiones que faltan en los lugares marcados < *??* > arriba.

**Ejercicio 3.57.**  ¿Cuántas adiciones se realizan cuando calculamos el *n* -ésimo número de Fibonacci utilizando la definición de fibs basada en el procedimiento add-streams ? Demuestre que la cantidad de adiciones sería exponencialmente mayor si hubiéramos implementado (delay < *exp* >) simplemente como (lambda () < *exp* >) , sin utilizar la optimización proporcionada por el procedimiento  memo-proc descrito en la sección [3.5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.1) . [64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_465)

**Ejercicio 3.58.**   Dar una interpretación de la corriente calculada mediante el siguiente procedimiento:

(definir (expandir num den radix)   
  (cons-stream   
   (cociente (\* num radix) den)   
   (expandir (resto (\* num radix) den) den radix)))

( El cociente es un primitivo que devuelve el cociente entero de dos números enteros). ¿Cuáles son los elementos sucesivos producidos por (expandir 1 7 10) ? ¿Qué se produce por (expandir 3 8 10) ?

**Ejercicio 3.59.**  En la sección  [2.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.5.3) vimos cómo implementar un sistema aritmético polinomial que represente polinomios como listas de términos. De manera similar, podemos trabajar con *series de potencias* , como







representadas como corrientes infinitas. Representaremos la serie *a* 0 + *a* 1 *x* + *a* 2 *x* 2 + *a* 3 *x* 3 + Â·Â·Â· como la corriente cuyos elementos son los coeficientes *a* 0 , *a* 1 , *a* 2 , *a* 3 , ... .

a. La integral de la serie *a* 0 + *a* 1 *x* + *a* 2 *x* 2 + *a* 3 *x* 3 + Â·Â·Â· es la serie



donde *c* es cualquier constante. Definir un procedimientointegr-series que toma como entrada un flujo *a* 0 , *a* 1 , *a* 2 , ... que representa una serie de potencias y devuelve el flujo *a* 0 , (1/2) *a* 1 , (1/3) *a* 2 , ... de coeficientes de los términos no constantes de la integral de la serie. (Como el resultado no tiene un término constante, no representa una serie de potencias; cuando usemos integr-series , utilizaremos la constante apropiada).

b. La función *x* *e x* es su propia derivada. Esto implica que *e x* y la integral de *e x* son la misma serie, excepto por el término constante, que es *e* 0 = 1. En consecuencia, podemos generar la serie para *e x* como

(define exp-series   
  (cons-stream 1 (integrate-series exp-series)))

Muestra cómo generar la serie para seno y coseno, partiendo del hecho de que la derivada del seno es coseno y la derivada del coseno es el negativo del seno:

(define serie coseno   
  (cons-stream 1 < *??* >))   
(define serie seno   
  (cons-stream 0 < *??* >))

**Ejercicio 3.60.**  Con series de potencias representadas como flujos de coeficientes como en el ejercicio  [3.59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.59) , la suma de series se implementa mediante add-streams . Complete la definición del siguiente procedimiento para multiplicar series:

(define (serie múltiple s1 s2)   
  (cons-stream < *??* > (add-streams < *??* > < *??* >)))

Puedes probar tu procedimiento verificando que *s i n* 2 *x* + *c o s* 2 *x* = 1, usando la serie del ejercicio  [3.59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.59) .

**Ejercicio 3.61.**   Sea *S* una serie de potencias (ejercicio  [3.59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.59) ) cuyo término constante es 1. Supongamos que queremos hallar la serie de potencias 1/ *S* , es decir, la serie *X* tal que *S* · *X* = 1. Escribimos *S* = 1 + *S R* donde *S R* es la parte de *S* después del término constante. Entonces podemos despejar *X* de la siguiente manera:

Forma

El contenido generado por IA puede ser incorrecto.

En otras palabras, *X* es la serie de potencias cuyo término constante es 1 y cuyos términos de orden superior están dados por el negativo de *S R* por *X.* Utilice esta idea para escribir un procedimiento invert-unit-series que calcule 1/ *S* para una serie de potencias *S* con término constante 1. Necesitará utilizar mul-series del ejercicio  [3.60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.60) .

**Ejercicio 3.62.**  Utilice los resultados de los ejercicios  [3.60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.60) y  [3.61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.61) para definir un procedimiento div-series que divida dos series de potencias. Div-series debería funcionar para dos series cualesquiera, siempre que la serie del denominador comience con un término constante distinto de cero. (Si el denominador tiene un término constante cero, entonces div-series debería indicar un error). Muestre cómo utilizar div-series junto con el resultado del ejercicio  [3.59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.59) para generarLa serie de potencias para la tangente.

**[3.5.3 Explotación del paradigma de flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.5.3)**

Los flujos con evaluación retrasada pueden ser una herramienta de modelado poderosa, que brinda muchos de los beneficios del estado local y la asignación. Además, evitan algunos de los enredos teóricos que acompañan la introducción de la asignación en un lenguaje de programación.

El enfoque de flujo puede ser esclarecedor porque nos permite construir sistemas con límites de módulos diferentes a los de los sistemas organizados en torno a la asignación de variables de estado. Por ejemplo, podemos pensar en una serie temporal (o señal) completa como foco de interés, en lugar de los valores de las variables de estado en momentos individuales. Esto hace que sea conveniente combinar y comparar componentes de estado de diferentes momentos.

**[Formulación de iteraciones como procesos de flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_471)**

En la sección  [1.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.1) , presentamos procesos iterativos, que proceden mediante la actualización de las variables de estado. Ahora sabemos que podemos representar el estado como una secuencia de valores "atemporal" en lugar de como un conjunto de variables que se deben actualizar. Adoptemos esta perspectiva al revisar el procedimiento de raíz cuadrada de la sección  [1.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.7) . Recordemos que la idea es generar una secuencia de suposiciones cada vez mejores para la raíz cuadrada de *x* aplicando una y otra vez el procedimiento que mejora las suposiciones:

(define (sqrt-mejora suposición x)   
  (suposición promedio (/ x suposición)))

En nuestro procedimiento sqrt original , hicimos que estas suposiciones fueran los valores sucesivos de una variable de estado. En cambio, podemos generar una secuencia infinita de suposiciones, comenzando con una suposición inicial de [1:65](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_472)

(define (sqrt-stream x)   
  (define conjeturas   
    (cons-stream 1.0   
                 (stream-map (lambda (conjetura)   
                               (sqrt-improve conjetura x))   
                             conjeturas)))   
  conjeturas)   
(display-stream (sqrt-stream 2))   
*1.*   
*1.5*   
*1.4166666666666665*   
*1.4142156862745097*   
*1.4142135623746899*   
...

Podemos generar más y más términos de la secuencia para obtener conjeturas cada vez mejores. Si lo deseamos, podemos escribir un procedimiento que siga generando términos hasta que la respuesta sea lo suficientemente buena. (Véase el ejercicio  [3.64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.64) ).

Otra iteración que podemos tratar de la misma manera es generar una aproximación a , basada en la serie alternada que vimos en la sección  [1.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.3.1) :



Primero generamos el flujo de sumandos de la serie (los recíprocos de los números enteros impares, con signos alternos). Luego tomamos el flujo de sumas de más y más términos (usando el procedimiento  de sumas parciales del ejercicio [3.55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.55) ) y escalamos el resultado por 4:

(define (pi-sumandos n)   
  (cons-stream (/ 1.0 n)   
               (stream-map - (pi-sumandos (+ n 2)))))  
(define pi-stream   
  (scale-stream (partial-sumas (pi-sumandos 1)) 4))   
(display-stream pi-stream)   
*4.*   
*2.666666666666667*   
*3.466666666666667*   
*2.8952380952380956*   
*3.3396825396825403*   
*2.9760461760461765*   
*3.2837384837384844*   
*3.017071817071818*   
...

Esto nos da una serie de aproximaciones cada vez mejores a , aunque las aproximaciones convergen con bastante lentitud. Ocho términos de la secuencia limitan el valor de entre 3,284 y 3,017.

Hasta ahora, nuestro uso del enfoque de flujo de estados no es muy diferente de la actualización de variables de estado. Pero los flujos nos dan la oportunidad de hacer algunos trucos interesantes. Por ejemplo, podemos transformar un flujo con un*acelerador de secuencia* que convierte una secuencia de aproximaciones en una nueva secuencia que converge al mismo valor que el original, sólo que más rápido.

Uno de esos aceleradores, debido al matemático suizo del siglo XVIIILeonhard Euler, trabaja bien con sucesiones que son sumas parciales de series alternadas (series de términos con signos alternados). En la técnica de Euler, si *S n* es el término *n de la secuencia suma original, entonces la secuencia acelerada tiene términos*



Por lo tanto, si la secuencia original se representa como un flujo de valores, la secuencia transformada viene dada por

(define (transformada-de-euler s)   
  (sea ((s0 (ref-corriente s 0))            *;  S n -1*  
        (s1 (ref-corriente s 1))            *;  S n*  
        (s2 (ref-corriente s 2)))           *;  S n +1*  
    (cons-corriente (- s2 (/ (cuadrado (- s2 s1))   
                          (+ s0 (\* -2 s1) s2)))   
                 (transformada-de-euler (cdr-corriente s)))))

Podemos demostrar la aceleración de Euler con nuestra secuencia de aproximaciones a :

(flujo de visualización (flujo pi de la transformada de Euler))   
*3,166666666666667*   
*3,1333333333333337*   
*3,1452380952380956*   
*3,13968253968254*   
*3,1427128427128435*   
*3,1408813408813416*   
*3,142071817071818*   
*3,1412548236077655*   
...

Mejor aún, podemos acelerar la secuencia acelerada y acelerarla recursivamente, y así sucesivamente. Es decir, creamos un flujo de flujos (una estructura que llamaremos*tabla* ) en la que cada flujo es la transformación del anterior:

(define (make-tableau transforma s)   
  (cons-stream s   
               (make-tableau transforma   
                             (transforma s))))

El cuadro tiene la forma

Forma

El contenido generado por IA puede ser incorrecto.

Finalmente, formamos una secuencia tomando el primer término en cada fila de la tabla:

(define (transformada de secuencia acelerada s)   
  (mapa de flujo carro de flujo   
              (transformada de creación de tabla s)))

Podemos demostrar este tipo de “superaceleración” de la secuencia:

(flujo de visualización (   
                                      flujo pi de transformada de Euler de secuencia acelerada))   
*4.*   
*3,166666666666667*   
*3,142105263157895*   
*3,141599357319005*   
*3,1415927140337785*   
*3,1415926539752927*   
*3,1415926535911765*   
*3,141592653589778*   
...

El resultado es impresionante. Tomando ocho términos de la secuencia se obtiene el valor correcto de hasta 14 decimales. Si hubiéramos utilizado solo la secuencia original, necesitaríamos calcular en el orden de 10 13 términos (es decir, expandir la serie lo suficiente para que los términos individuales sean menores que 10 -13 ) para obtener tanta precisión. Podríamos haber implementado estas técnicas de aceleración sin usar flujos. Pero la formulación de flujo es particularmente elegante y conveniente porque toda la secuencia de estados está disponible para nosotros como una estructura de datos que se puede manipular con un conjunto uniforme de operaciones.

**Ejercicio 3.63.**   Louis Reasoner pregunta por qué el procedimiento sqrt-stream no se escribió de la siguiente manera más sencilla, sin las variables locales :

(define (sqrt-stream x)   
  (cons-stream 1.0   
               (stream-map (lambda (suposición)   
                             (sqrt-improve suposición x))   
                           (sqrt-stream x))))

Alyssa P. Hacker responde que esta versión del procedimiento es considerablemente menos eficiente porque realiza cálculos redundantes. Explique la respuesta de Alyssa. ¿Las dos versiones seguirían diferenciándose en eficiencia si nuestra implementación de delay utilizara solo (lambda() <exp> *)* sin utilizar la optimización proporcionada por memo-proc (sección  [3.5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.1) )?

**Ejercicio 3.64.**   Redactar un procedimientostream-limit que toma como argumentos un flujo y un número (la tolerancia). Debe examinar el flujo hasta encontrar dos elementos sucesivos que difieran en valor absoluto por menos que la tolerancia y devolver el segundo de los dos elementos. Con esto, podríamos calcular raíces cuadradas hasta una tolerancia dada

(definir (sqrt x tolerancia)   
  (límite de flujo (sqrt-flujo x) tolerancia))

**Ejercicio 3.65.**  Utilice la serie



Calcular tres secuencias de aproximaciones al logaritmo natural de 2, de la misma manera que hicimos anteriormente para . ¿Con qué rapidez convergen estas secuencias?

**[Corrientes infinitas de pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_476)**

En la sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) , vimos cómo el paradigma de secuencias maneja los bucles anidados tradicionales como procesos definidos en secuencias de pares. Si generalizamos esta técnica a secuencias infinitas, entonces podemos escribir programas que no se representan fácilmente como bucles, porque el "bucle" debe abarcar un conjunto infinito.

Por ejemplo, supongamos que queremos generalizar el procedimiento de pares de suma prima de la sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) para producir la secuencia de pares de *todos* los números enteros ( *i* , *j* ) con *i* < *j* tales que *i* + *j* es primo. Si int-pairs es la secuencia de todos los pares de números enteros ( *i* , *j* ) con *i* < *j* , entonces nuestra secuencia requerida es simplemente [66](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_477)

(filtro de flujo (lambda (par)   
                 (prime? (+ (par de autos) (par de cadr))))   
               pares int)

Nuestro problema, entonces, es producir los pares de flujos int . De manera más general, supongamos que tenemos dos flujos *S* = ( *S i* ) y *T* = ( *T j* ), e imaginemos la matriz rectangular infinita

Forma

El contenido generado por IA puede ser incorrecto.

Deseamos generar un flujo que contenga todos los pares en la matriz que se encuentran en la diagonal o por encima de ella, es decir, los pares

Forma

El contenido generado por IA puede ser incorrecto.

(Si tomamos tanto *S* como *T* como el flujo de números enteros, entonces este será nuestro flujo deseado de pares de números enteros ).

Llamemos a la corriente general de pares (pares ST) y consideremos que está compuesta de tres partes: el par ( *S* 0 , *T* 0 ), el resto de los pares de la primera fila y los pares restantes: [67](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_478)

Forma

El contenido generado por IA puede ser incorrecto.

Observe que la tercera pieza en esta descomposición (pares que no están en la primera fila) son (recursivamente) los pares formados a partir de (stream-cdr S) y (stream-cdr T) . Observe también que la segunda pieza (el resto de la primera fila) es

(mapa de flujo (lambda (x) (lista (flujo-car s) x))   
            (flujo-cdr t))

De esta manera podemos formar nuestro flujo de pares de la siguiente manera:

(define (pares s t)   
  (cons-stream   
   (lista (stream-car s) (stream-car t))   
   (< *combinar-de-alguna-manera* >   
       (stream-map (lambda (x) (lista (stream-car s) x))   
                   (stream-cdr t))   
       (pares (stream-cdr s) (stream-cdr t)))))

Para completar el procedimiento, debemos elegir alguna forma de combinar los dos flujos internos. Una idea es utilizar el flujo análogo del procedimiento de anexión de la sección  [2.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.1) :

(define (stream-append s1 s2)   
  (si (stream-null? s1)   
      s2   
      (cons-stream (stream-car s1)   
                   (stream-append (stream-cdr s1) s2))))

Sin embargo, esto no es adecuado para flujos infinitos, ya que toma todos los elementos del primer flujo antes de incorporar el segundo flujo. En particular, si tratamos de generar todos los pares de números enteros positivos utilizando

(pares de números enteros)

Nuestro flujo de resultados primero intentará recorrer todos los pares con el primer entero igual a 1 y, por lo tanto, nunca producirá pares con ningún otro valor del primer entero.

Para manejar flujos infinitos, necesitamos idear un orden de combinación que garantice que, si dejamos que nuestro programa se ejecute durante el tiempo suficiente, se llegará a cada elemento. Una forma elegante de lograr esto es con el siguiente procedimiento de intercalación : [68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_479)

(define (entrelazar s1 s2)   
  (si (flujo-nulo? s1)   
      s2   
      (cons-flujo (flujo-car s1)   
                   (entrelazar s2 (flujo-cdr s1)))))

Dado que el intercalado toma elementos alternativamente de los dos flujos, cada elemento del segundo flujo eventualmente encontrará su camino hacia el flujo intercalado, incluso si el primer flujo es infinito.

De este modo, podemos generar el flujo de pares requerido como

(define (pares s t)   
  (cons-stream   
   (lista (stream-car s) (stream-car t))   
   (entrelazado   
    (stream-map (lambda (x) (lista (stream-car s) x))   
                (stream-cdr t))   
    (pares (stream-cdr s) (stream-cdr t)))))

**Ejercicio 3.66.**   Examine la secuencia (pares enteros enteros) . ¿Puede hacer algún comentario general sobre el orden en el que se colocan los pares en la secuencia? Por ejemplo, ¿aproximadamente cuántos pares preceden al par (1,100)? ¿al par (99,100)? ¿al par (100,100)? (Si puede hacer afirmaciones matemáticas precisas aquí, mucho mejor. Pero siéntase libre de dar respuestas más cualitativas si se siente abrumado).

**Ejercicio 3.67.**   Modifique el procedimiento de pares de modo que (pares enteros enteros) produzca el flujo de *todos* los pares de enteros ( *i* , *j* ) (sin la condición *i* < *j* ). Sugerencia: deberá mezclar un flujo adicional.

**Exercise 3.68.**  Louis Reasoner thinks that building a stream of pairs from three parts is unnecessarily complicated. Instead of separating the pair (*S*0,*T*0) from the rest of the pairs in the first row, he proposes to work with the whole first row, as follows:

(define (pairs s t)  
  (interleave  
   (stream-map (lambda (x) (list (stream-car s) x))  
               t)  
   (pairs (stream-cdr s) (stream-cdr t))))

Does this work? Consider what happens if we evaluate (pairs integers integers) using Louis's definition of pairs.

**Exercise 3.69.**  Write a procedure triples that takes three infinite streams, *S*, *T*, and *U*, and produces the stream of triples (*Si*,*Tj*,*Uk*) such that *i* < *j* < *k*. Use triples to generate the stream of all Pythagorean triples of positive integers, i.e., the triples (*i*,*j*,*k*) such that *i* < *j* and *i*2 + *j*2 = *k*2.

**Exercise 3.70.**  It would be nice to be able to generate streams in which the pairs appear in some useful order, rather than in the order that results from an *ad hoc* interleaving process. We can use a technique similar to the merge procedure of exercise [3.56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.56), if we define a way to say that one pair of integers is ``less than'' another. One way to do this is to define a ``weighting function'' *W*(*i*,*j*) and stipulate that (*i*1,*j*1) is less than (*i*2,*j*2) if *W*(*i*1,*j*1) < *W*(*i*2,*j*2). Write a procedure merge-weighted that is like merge, except that merge-weighted takes an additional argument weight, which is a procedure that computes the weight of a pair, and is used to determine the order in which elements should appear in the resulting merged stream.[69](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_485) Using this, generalize pairs to a procedure weighted-pairs that takes two streams, together with a procedure that computes a weighting function, and generates the stream of pairs, ordered according to weight. Use your procedure to generate

a. the stream of all pairs of positive integers (*i*,*j*) with *i* < *j* ordered according to the sum *i* + *j*

b. the stream of all pairs of positive integers (*i*,*j*) with *i* < *j*, where neither *i* nor *j* is divisible by 2, 3, or 5, and the pairs are ordered according to the sum 2 *i* + 3 *j* + 5 *i* *j*.

**Exercise 3.71.**  Numbers that can be expressed as the sum of two cubes in more than one way are sometimes called *Ramanujan numbers*, in honor of the mathematician Srinivasa Ramanujan.[70](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_487) Ordered streams of pairs provide an elegant solution to the problem of computing these numbers. To find a number that can be written as the sum of two cubes in two different ways, we need only generate the stream of pairs of integers (*i*,*j*) weighted according to the sum *i*3 + *j*3 (see exercise [3.70](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.70)), then search the stream for two consecutive pairs with the same weight. Write a procedure to generate the Ramanujan numbers. The first such number is 1,729. What are the next five?

**Exercise 3.72.**  In a similar way to exercise [3.71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.71) generate a stream of all numbers that can be written as the sum of two squares in three different ways (showing how they can be so written).

**[Streams as signals](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_489)**

We began our discussion of streams by describing them as computational analogs of the ``signals'' in signal-processing systems. In fact, we can use streams to model signal-processing systems in a very direct way, representing the values of a signal at successive time intervals as consecutive elements of a stream. For instance, we can implement an *integrator* or *summer* that, for an input stream *x* = (*xi*), an initial value *C*, and a small increment *dt*, accumulates the sum



and returns the stream of values *S* = (*Si*). The following integral procedure is reminiscent of the ``implicit style'' definition of the stream of integers (section [3.5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.2)):

(define (integral integrand initial-value dt)  
  (define int  
    (cons-stream initial-value  
                 (add-streams (scale-stream integrand dt)  
                              int)))  
  int)

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figure 3.32:**  The integral procedure viewed as a signal-processing system. |

Figure [3.32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.32) is a picture of a signal-processing system that corresponds to the integral procedure. The input stream is scaled by *dt* and passed through an adder, whose output is passed back through the same adder. The self-reference in the definition of int is reflected in the figure by the feedback loop that connects the output of the adder to one of the inputs.

**Exercise 3.73.**  

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto.        *v* = *v*0 + (1/*C*)0*ti* *dt* + *R* *i*  Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figure 3.33:**  An RC circuit and the associated signal-flow diagram. |

We can model electrical circuits using streams to represent the values of currents or voltages at a sequence of times. For instance, suppose we have an *RC circuit* consisting of a resistor of resistance *R* and a capacitor of capacitance *C* in series. The voltage response *v* of the circuit to an injected current *i* is determined by the formula in figure [3.33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.33), whose structure is shown by the accompanying signal-flow diagram.

Write a procedure RC that models this circuit. RC should take as inputs the values of *R*, *C*, and *dt* and should return a procedure that takes as inputs a stream representing the current *i* and an initial value for the capacitor voltage *v*0 and produces as output the stream of voltages *v*. For example, you should be able to use RC to model an RC circuit with *R* = 5 ohms, *C* = 1 farad, and a 0.5-second time step by evaluating (define RC1 (RC 5 1 0.5)). This defines RC1 as a procedure that takes a stream representing the time sequence of currents and an initial capacitor voltage and produces the output stream of voltages.

**Exercise 3.74.**  Alyssa P. Hacker is designing a system to process signals coming from physical sensors. One important feature she wishes to produce is a signal that describes the *zero crossings* of the input signal. That is, the resulting signal should be + 1 whenever the input signal changes from negative to positive, - 1 whenever the input signal changes from positive to negative, and 0 otherwise. (Assume that the sign of a 0 input is positive.) For example, a typical input signal with its associated zero-crossing signal would be

...1  2  1.5  1  0.5  -0.1  -2  -3  -2  -0.5  0.2  3  4 ...... 0  0    0  0    0     -1  0   0   0     0    1  0  0 ...

En el sistema de Alyssa, la señal del sensor se representa como un flujo de datos de detección y el flujo de cruces por cero es el flujo correspondiente de cruces por cero. Alyssa primero escribe un procedimiento detector de cambio de signo que toma dos valores como argumentos y compara los signos de los valores para producir un 0, 1 o -1 apropiado. Luego construye su flujo de cruces por cero de la siguiente manera:

(define (hacer cruces por cero flujo de entrada último valor)   
  (cons-flujo   
   (detector-cambio-signo (flujo-auto flujo-entrada) último valor)   
   (hacer cruces por cero (flujo-cdr flujo-entrada)   
                        (flujo-auto flujo-entrada))))   
  
(define cruces por cero (hacer cruces por cero datos-de-sentido 0))

La jefa de Alyssa, Eva Lu Ator, pasa y sugiere que este programa es aproximadamente equivalente al siguiente, que utiliza la versión generalizada de stream-map del ejercicio  [3.50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.50) :

(define cruces por cero   
  (mapa de flujo detector de cambio de signo datos sensoriales < *expresión* >))

Complete el programa suministrando la < *expresión* > indicada.

**Ejercicio 3.75.**  Desafortunadamente, el detector de cruces por cero de Alyssa en el ejercicio  [3.74](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.74) resulta ser insuficiente, porque la señal ruidosa del sensor produce cruces por cero espurios. Lem E. Tweakit, un especialista en hardware, sugiere que Alyssa suavice la señal para filtrar el ruido antes de extraer los cruces por cero. Alyssa sigue su consejo y decide extraer los cruces por cero de la señal construida promediando cada valor de los datos de detección con el valor anterior. Le explica el problema a su asistente, Louis Reasoner, quien intenta implementar la idea, modificando el programa de Alyssa de la siguiente manera:

(define (hacer cruces por cero flujo de entrada último valor)   
  (let ((avpt (/ (+ (flujo-car flujo-de-entrada) último valor) 2)))   
    (cons-flujo (detector-de-cambio-de-signo avpt último valor)   
                 (hacer cruces por cero (flujo-cdr flujo-de-entrada)   
                                      avpt))))

Esto no implementa correctamente el plan de Alyssa. Busque el error que ha instalado Louis y corríjalo sin cambiar la estructura del programa. (Sugerencia: deberá aumentar la cantidad de argumentos para make-zero-crossings ).

**Ejercicio 3.76.**  Eva Lu Ator tiene una crítica al enfoque de Louis en el ejercicio  [3.75](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.75) . El programa que escribió no es modular, porque mezcla la operación de suavizado con la extracción de cruce por cero. Por ejemplo, no debería ser necesario cambiar el extractor si Alyssa encuentra una mejor manera de condicionar su señal de entrada. Ayude a Louis escribiendo un procedimiento smooth que tome un flujo como entrada y produzca un flujo en el que cada elemento sea el promedio de dos elementos de flujo de entrada sucesivos. Luego use smooth como un componente para implementar el detector de cruce por cero en un estilo más modular.

**[3.5.4 Flujos y evaluación diferida](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_3.5.4)**

El procedimiento integral al final de la sección anterior muestra cómo podemos usar flujos para modelar sistemas de procesamiento de señales que contienenbucles de retroalimentación. El bucle de retroalimentación para el sumador que se muestra en la figura  [3.32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.32) está modelado por el hecho de que la integralEl flujo interno int se define en términos de sí mismo:

(define int   
  (cons-stream valor-inicial   
               (add-streams (scale-stream integrando dt)   
                            int)))

La capacidad del intérprete para manejar una definición implícita de este tipo depende del retraso que se incorpora en cons-stream . Sin este retraso , el intérprete no podría construir int antes de evaluar ambos argumentos para cons-stream , lo que requeriría que int ya estuviera definido. En general, el retraso es crucial para usar streams para modelar sistemas de procesamiento de señales que contienen bucles. Sin retraso , nuestros modelos tendrían que formularse de modo que las entradas a cualquier componente de procesamiento de señales se evaluaran por completo antes de que se pudiera producir la salida. Esto prohibiría los bucles.

Desafortunadamente, los modelos de flujo de sistemas con bucles pueden requerir el uso de retardos más allá del retardo "oculto" proporcionado por cons-stream . Por ejemplo, la figura  [3.34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.34) muestra un sistema de procesamiento de señales para resolver el problemaecuación diferencial *d y* / *d t* = *f* ( *y* ) donde *f* es una función dada. La figura muestra un componente de mapeo, que aplica *f* a su señal de entrada, vinculado en un bucle de retroalimentación a un integrador de una manera muy similar a la de los circuitos de computadora analógicos que se utilizan realmente para resolver tales ecuaciones.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.34:**   Un "circuito de computadora analógico" que resuelve la ecuación *d y* / *d t* = *f* ( *y* ). |

Suponiendo que nos dan un valor inicial *y* 0 para *y* , podríamos intentar modelar este sistema utilizando el procedimiento

(definir (resolver f y0 dt)   
  (definir y (integral dy y0 dt))   
  (definir dy (mapa de flujo f y))   
  y)

Este procedimiento no funciona, porque en la primera línea de solve la llamada a integral requiere que se defina la entrada dy , lo que no sucede hasta la segunda línea de solve .

Por otra parte, la intención de nuestra definición tiene sentido, porque podemos, en principio, comenzar a generar el flujo y sin conocer dy . De hecho, integral y muchas otras operaciones de flujo tienen propiedades similares a las de cons-stream , en el sentido de que podemos generar parte de la respuesta dada solo información parcial sobre los argumentos. Para integral , el primer elemento del flujo de salida es el valor inicial especificado . Por lo tanto, podemos generar el primer elemento del flujo de salida sin evaluar el integrando dy . Una vez que conocemos el primer elemento de y , el mapa de flujo en la segunda línea de solve puede comenzar a trabajar para generar el primer elemento de dy , que producirá el siguiente elemento de y , y así sucesivamente.

Para aprovechar esta idea, redefiniremos la integral para esperar que el flujo del integrando sea un*argumento retrasado* . Integral obligará a que el integrando se evalúe solo cuando se requiera generar más que el primer elemento del flujo de salida:

(define (integrando retardado integral valor inicial dt)   
  (define int   
    (cons-stream valor inicial   
                 (let ((integrando (fuerza integrando retardado)))   
                   (add-streams (flujo de escala integrando dt)   
                                int))))   
  int)

Ahora podemos implementar nuestro procedimiento de resolución retrasando la evaluación de dy en la definición de y : [71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_494)

(definir (resolver f y0 dt)   
  (definir y (integral (retardo dy) y0 dt))   
  (definir dy (mapa de flujo f y))   
  y)

En general, cada invocador de integrales ahora debe retrasar el argumento del integrando. Podemos demostrar que el procedimiento de resolución funciona aproximando*e* 2.718 calculando el valor en *y* = 1 de la solución de la ecuación diferencial *d y* / *d t* = *y* con condición inicial *y* (0) = 1:

(stream-ref (solve (lambda (y) y) 1 0,001) 1000)   
*2,716924*

**Ejercicio 3.77.**   El procedimiento integral utilizado anteriormente era análogo a la definición "implícita" de la secuencia infinita de números enteros de la sección  [3.5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.2) . Alternativamente, podemos dar una definición de integral que se parezca más a la de los números enteros a partir de (también en la sección  [3.5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.2) ):

(define (integrando integral valor-inicial dt)   
  (cons-flujo valor-inicial   
               (si (flujo-nulo? integrando)   
                   el-flujo-vacío   
                   (integrando (flujo-cdr integrando)   
                             (+ (\* dt (flujo-car integrando))   
                                valor-inicial)   
                             dt))))

Cuando se utiliza en sistemas con bucles, este procedimiento tiene el mismo problema que nuestra versión original de integral . Modifique el procedimiento para que espere el integrando como argumento retrasado y, por lo tanto, se pueda utilizar en el procedimiento de resolución que se muestra arriba.

**Ejercicio 3.78.**  

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.35:**   Diagrama de flujo de señales para la solución de una ecuación diferencial lineal de segundo orden. |

Consideremos el problema de diseñar un sistema de procesamiento de señales para estudiar la ecuación diferencial lineal homogénea de segundo orden.



El flujo de salida, que modela *y* , se genera mediante una red que contiene un bucle. Esto se debe a que el valor de *d* 2 *y* / *d t* 2 depende de los valores de *y* y *d y* / *d t* y ambos se determinan integrando *d* 2 *y* / *d t* 2 . El diagrama que nos gustaría codificar se muestra en la figura  [3.35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.35) . Escriba un procedimiento solve-2nd que tome como argumentos las constantes *a* , *b* y *d t* y los valores iniciales *y* 0 y *d y* 0 para *y* y *d y* / *d t* y genere el flujo de valores sucesivos de *y* .

**Ejercicio 3.79.**  Generalice el procedimiento  solve-2nd del ejercicio [3.78](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.78) para que pueda usarse para resolver ecuaciones diferenciales generales de segundo orden *d* 2 *y* / *d t* 2 = *f* ( *d y* / *d t* , *y* ).

**Ejercicio 3.80.**  Un *circuito RLC en serie* consta de una resistencia, un condensador y un inductor conectados en serie, como se muestra en la figura  [3.36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.36) . Si *R* , *L* y *C* son la resistencia, la inductancia y la capacitancia, entonces las relaciones entre el voltaje ( *v* ) y la corriente ( *i* ) para los tres componentes se describen mediante las ecuaciones

Forma

El contenido generado por IA puede ser incorrecto.

y las conexiones del circuito dictan las relaciones



La combinación de estas ecuaciones muestra que el estado del circuito (resumido por *v C* , el voltaje a través del capacitor, y *i L* , la corriente en el inductor) se describe mediante el par de ecuaciones diferenciales

Forma

El contenido generado por IA puede ser incorrecto.

El diagrama de flujo de señales que representa este sistema de ecuaciones diferenciales se muestra en la figura  [3.37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.37) .

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.36:**   Un circuito RLC en serie. |

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.37:**   Diagrama de flujo de señales para la solución de un circuito RLC en serie. |

Escriba un procedimiento RLC que tome como argumentos los parámetros *R* , *L* y *C* del circuito y el incremento de tiempo *d t* . De manera similar a la del procedimiento  RC del ejercicio [3.73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.73) , RLC debe producir un procedimiento que tome los valores iniciales de las variables de estado, *v C* 0 e *i L* 0 , y produzca un par (usando cons ) de los flujos de estados *v C* e *i L* . Usando RLC , genere el par de flujos que modela el comportamiento de un circuito RLC en serie con *R* = 1 ohmio, *C* = 0,2 faradios, *L* = 1 henrio, *d t* = 0,1 segundo y valores iniciales *i L* 0 = 0 amperios y *v C* 0 = 10 voltios.

**[Evaluación de orden normal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_499)**

Los ejemplos de esta sección ilustran cómo el uso explícito de delay y force proporciona una gran flexibilidad de programación, pero los mismos ejemplos también muestran cómo esto puede hacer que nuestros programas sean más complejos. Nuestro nuevo procedimiento integral , por ejemplo, nos da el poder de modelar sistemas con bucles, pero ahora debemos recordar que integral debe ser llamado con un integrando retrasado, y cada procedimiento que use integral debe ser consciente de esto. En efecto, hemos creado dos clases de procedimientos: procedimientos ordinarios y procedimientos que toman argumentos retrasados. En general, crear clases separadas de procedimientos nos obliga a crear también clases separadas de procedimientos de orden superior. [72](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_500)

Una forma de evitar la necesidad de dos clases diferentes de procedimientos es hacer que todos los procedimientos tomen argumentos retrasados. Podríamos adoptar un modelo de evaluación en el que todos los argumentos de los procedimientos se retrasaran automáticamente y los argumentos se obligaran solo cuando realmente se necesitan (por ejemplo, cuando los requiere una operación primitiva). Esto transformaría nuestro lenguaje para usar la evaluación de orden normal, que describimos por primera vez cuando presentamos el modelo de sustitución para la evaluación en la sección  [1.1.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.5) . La conversión a la evaluación de orden normal proporciona una forma uniforme y elegante de simplificar el uso de la evaluación retrasada, y esta sería una estrategia natural para adoptar si solo nos interesara el procesamiento de flujo. En la sección  [4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2) , después de haber estudiado el evaluador, veremos cómo transformar nuestro lenguaje de esta manera. Desafortunadamente, incluir retrasos en las llamadas a procedimientos causa estragos en nuestra capacidad para diseñar programas que dependen del orden de los eventos, como programas que usan asignación, mutan datos o realizan entrada o salida. Incluso el único retraso en cons-stream puede causar gran confusión, como lo ilustran los ejercicios  [3.51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.51) y  [3.52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.52) . Hasta donde se sabe, la mutabilidad y la evaluación retrasada no se llevan bien en los lenguajes de programación, y diseñar formas de lidiar con ambas a la vez es un área activa de investigación.

[**3.5.5 Modularidad de programas funcionales y modularidad de objetos**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_3.5.5)

Como vimos en la sección  [3.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.2) , uno de los principales beneficios de introducir la asignación es que podemos aumentar la modularidad de nuestros sistemas encapsulando u "ocultando" partes del estado de un sistema grande dentro de variables locales. Los modelos de flujo pueden proporcionar una modularidad equivalente sin el uso de la asignación.A modo de ilustración, podemos reimplementar la estimación de Monte Carlo de , que examinamos en la sección  [3.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.2) , desde un punto de vista de procesamiento de flujo.

La cuestión clave de la modularidad era que queríamos ocultar el estado interno de un generador de números aleatorios a los programas que utilizaban números aleatorios. Comenzamos con un procedimiento rand-update , cuyos valores sucesivos proporcionaban nuestro suministro de números aleatorios, y lo utilizamos para producir un generador de números aleatorios:

(define rand   
  (let ((x random-init))   
    (lambda ()   
      (set! x (rand-update x))   
      x)))

En la formulación del flujo no hay un generador de números aleatorios *per se* , solo un flujo de números aleatorios producidos por llamadas sucesivas a rand-update :

(definir números aleatorios   
  (cons-stream random-init   
               (stream-map rand-update números aleatorios)))

Usamos esto para construir el flujo de resultados del experimento Cesíro realizado en pares consecutivos en el flujo de números aleatorios :

(define cesaro-stream   
  (map-successive-pairs (lambda (r1 r2) (= (mcd r1 r2) 1))   
                        números aleatorios))  
  
(define (map-pares-sucesivos f s)   
  (cons-stream   
   (f (stream-car s) (stream-car (stream-cdr s)))   
   (map-pares-sucesivos f (stream-cdr (stream-cdr s)))))

Ahora, el flujo de cesaro se alimenta a un procedimiento de Montecarlo , que produce un flujo de estimaciones de probabilidades. Luego, los resultados se convierten en un flujo de estimaciones de . Esta versión del programa no necesita un parámetro que indique cuántos ensayos realizar. Se obtienen mejores estimaciones de (a partir de la realización de más experimentos) si se observa más a fondo el flujo de pi :

(define (monte-carlo experiment-stream pasó falló)   
  (define (siguiente pasó falló)   
    (cons-stream   
     (/ pasó (+ pasó falló))   
     (monte-carlo   
      (stream-cdr experiment-stream) pasó falló)))   
  (if (stream-car experiment-stream)   
      (siguiente (+ pasó 1) falló)   
      (siguiente pasó (+ falló 1))))   
  
(define pi   
  (stream-map (lambda (p) (sqrt (/ 6 p)))   
              (monte-carlo cesaro-stream 0 0)))

Este enfoque tiene una modularidad considerable, porque todavía podemos formular un procedimiento general de Montecarlo que pueda manejar experimentos arbitrarios, pero no hay asignación ni estado local.

**Ejercicio 3.81.**  En el ejercicio  [3.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.6) se analizó la generalización del generador de números aleatorios para permitir restablecer la secuencia de números aleatorios de modo de producir secuencias repetibles de números "aleatorios". Genere una formulación de flujo de este mismo generador que funcione sobre un flujo de entrada de solicitudes para generar un nuevo número aleatorio o para restablecer la secuencia a un valor especificado y que produzca el flujo deseado de números aleatorios. No utilice la asignación en su solución.

**Ejercicio 3.82.**  Repita el ejercicio  [3.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.5) sobre la integración de Monte Carlo en términos de flujos. La versión de flujo de la estimación integral no tendrá un argumento que indique cuántos ensayos se deben realizar. En cambio, producirá un flujo de estimaciones basado en sucesivos ensayos.

**[Una visión del tiempo desde la programación funcional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_503)**

Ahora volvamos a las cuestiones de los objetos y el estado que se plantearon al principio de este capítulo y examinémoslas desde una nueva perspectiva. Introdujimos la asignación y los objetos mutables para proporcionar un mecanismo para la construcción modular de programas que modelan sistemas con estado. Construimos objetos computacionales con variables de estado locales y utilizamos la asignación para modificar estas variables. Modelamos el comportamiento temporal de los objetos en el mundo mediante el comportamiento temporal de los objetos computacionales correspondientes.

Ahora hemos visto que los flujos proporcionan una forma alternativa de modelar objetos con estado local. Podemos modelar una cantidad cambiante, como el estado local de algún objeto, utilizando un flujo que representa el historial temporal de estados sucesivos. En esencia, representamos el tiempo explícitamente, utilizando flujos, de modo que desacoplamos el tiempo en nuestro mundo simulado de la secuencia de eventos que tienen lugar durante la evaluación. De hecho, debido a la presencia de retrasos, puede haber poca relación entre el tiempo simulado en el modelo y el orden de los eventos durante la evaluación.

Para contrastar estos dos enfoques de modelado, reconsideremos la implementación de un "procesador de retiro" queMonitorea el saldo de una cuenta bancaria. En la sección  [3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.3) implementamos una versión simplificada de dicho procesador:

(define (make-simplified-withdraw balance)   
  (lambda (amount)   
    (set! balance (- balance amount))   
    balance))

Las llamadas a make-simplified-withdraw producen objetos computacionales, cada uno con una variable de estado local, balance , que se reduce con llamadas sucesivas al objeto. El objeto toma un monto como argumento y devuelve el nuevo balance. Podemos imaginar al usuario de una cuenta bancaria ingresando una secuencia de entradas a dicho objeto y observando la secuencia de valores devueltos que se muestran en una pantalla.

Alternativamente, podemos modelar un procesador de retiros como un procedimiento que toma como entrada un saldo y un flujo de montos a retirar y produce el flujo de saldos sucesivos en la cuenta:

(define (flujo-retirar saldo monto-flujo)   
  (cons-flujo   
   saldo   
   (flujo-retirar (- saldo (flujo-car monto-flujo))   
                    (flujo-cdr monto-flujo))))

Stream-withdraw implementa una función matemática bien definida cuya salida está completamente determinada por su entrada. Sin embargo, supongamos que la entrada amount-stream es la secuencia de valores sucesivos escritos por el usuario y que se muestra la secuencia de saldos resultante. Entonces, desde la perspectiva del usuario que está escribiendo valores y viendo los resultados, el proceso de secuencia tiene el mismo comportamiento que el objeto creado por make-simplified-withdraw . Sin embargo, con la versión de secuencia, no hay asignación, ni variable de estado local y, en consecuencia, no hay ninguna de las dificultades teóricas que encontramos.en la sección  [3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.3) . ¡Aún así el sistema tiene estado!

Esto es realmente notable. Aunque stream-withdraw implementa una función matemática bien definida cuyo comportamiento no cambia, la percepción del usuario aquí es la de interactuar con un sistema que tiene un estado cambiante. Una forma de resolver esta paradoja es darse cuenta de que es la existencia temporal del usuario la que impone el estado en el sistema. Si el usuario pudiera dar un paso atrás en la interacción y pensar en términos de flujos de saldos en lugar de transacciones individuales, el sistema parecería no tener estado. [73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_504)

Desde el punto de vista de una parte de un proceso complejo, las otras partes parecen cambiar con el tiempo. Tienen estados locales ocultos que varían con el tiempo. Si deseamos escribir programas que modelen este tipo de descomposición natural en nuestro mundo (tal como lo vemos desde nuestro punto de vista como parte de ese mundo) con estructuras en nuestra computadora, creamos objetos computacionales que no son funcionales: deben cambiar con el tiempo. Modelamos el estado con variables de estado locales y modelamos los cambios de estado con asignaciones a esas variables. Al hacer esto, hacemos que el tiempo de ejecución de un modelo computacional sea el tiempo en el mundo del que formamos parte y, por lo tanto, obtenemos "objetos" en nuestra computadora.

El modelado con objetos es poderoso e intuitivo, en gran medida porque coincide con la percepción de interactuar con un mundo del que formamos parte. Sin embargo, como hemos visto repetidamente a lo largo de este capítulo, estos modelos plantean problemas espinosos de restricción del orden de los eventos y de sincronización de múltiples procesos. La posibilidad de evitar estos problemas ha estimulado el desarrollo de*Lenguajes de programación funcional* , que no incluyen ninguna disposición para la asignación o datos mutables. En un lenguaje de este tipo, todos los procedimientos implementan funciones matemáticas bien definidas de sus argumentos, cuyo comportamiento no cambia. El enfoque funcional es extremadamenteatractivo para tratar con sistemas concurrentes. [74](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_505)

Por otra parte, si observamos con atención, podemos ver que los problemas relacionados con el tiempo también se infiltran en los modelos funcionales. Un área particularmente problemática surge cuando deseamos diseñar sistemas interactivos, especialmente aquellos que modelan interacciones entre entidades independientes. Por ejemplo, considere una vez más la implementación de un sistema bancario que permite cuentas bancarias conjuntas. En un sistema convencional que utiliza asignación y objetos, modelaríamos el hecho de que Peter y Paul comparten una cuenta haciendo que tanto Peter como Paul envíen sus solicitudes de transacción al mismo objeto de cuenta bancaria, como vimos en la sección  [3.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.1.3) . Desde el punto de vista del flujo, donde no hay "objetos" *per se* , ya hemos indicado que una cuenta bancaria puede modelarse como un proceso que opera en un flujo de solicitudes de transacción para producir un flujo de respuestas. En consecuencia, podríamos modelar el hecho de que Peter y Paul tienen una cuenta bancaria conjunta fusionando el flujo de solicitudes de transacción de Peter con el flujo de solicitudes de Paul y alimentando el resultado al proceso del flujo de cuenta bancaria, como se muestra en la figura  [3.38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_3.38) .

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 3.38:**   Una cuenta bancaria conjunta, modelada mediante la fusión de dos flujos de solicitudes de transacciones. |

El problema con esta formulación está en la noción de *fusión* . No se pueden fusionar los dos flujos simplemente tomando alternativamente una solicitud de Peter y una solicitud de Paul. Supongamos que Paul accede a la cuenta sólo muy raramente. Difícilmente podríamos obligar a Peter a esperar a que Paul acceda a la cuenta antes de poder emitir una segunda transacción. Independientemente de cómo se implemente dicha fusión, debe intercalar los dos flujos de transacciones de alguna manera que esté restringida por el "tiempo real" tal como lo perciben Peter y Paul, en el sentido de que, si Peter y Paul se encuentran, pueden acordar que ciertas transacciones se procesaron antes de la reunión y otras transacciones se procesaron después de la reunión. [75](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_506) Esta es precisamente la misma restricción con la que tuvimos que lidiar en la sección  [3.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.4.1) , donde encontramos la necesidad de introducir una sincronización explícita para asegurar un orden "correcto" de eventos en el procesamiento concurrente de objetos con estado. Por lo tanto, en un intento de apoyar el estilo funcional, la necesidad de fusionar las entradas de diferentes agentes reintroduce los mismos problemas que el estilo funcional pretendía eliminar.

Comenzamos este capítulo con el objetivo de construir modelos computacionales cuya estructura coincida con nuestra percepción del mundo real que estamos tratando de modelar. Podemos modelar el mundo como una colección de objetos separados, limitados en el tiempo e interactuando con un estado, o podemos modelar el mundo como una unidad única, atemporal y sin estado. Cada perspectiva tiene ventajas poderosas, pero ninguna de ellas por sí sola es completamente satisfactoria. Aún no ha surgido una gran unificación. [76](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_507)

[52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_442) Los físicos a veces adoptan este punto de vista introduciendo laLas "líneas de universo" de partículas se utilizan como un mecanismo para razonar sobre el movimiento. También hemos mencionado (sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) ) que esta es la forma natural de pensar sobre los sistemas de procesamiento de señales. Exploraremos las aplicaciones de los flujos al procesamiento de señales en la sección  [3.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.3) .

[53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_443) Supongamos que tenemos un predicadoprimo(por ejemplo, como en la sección [1.2.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.6)) que prueba la primalidad.

[54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_444) En la implementación del MIT,the-empty-stream es lo mismo que la lista vacía '() , y stream-null? es lo mismo que null? .

[55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_445) Esto debería preocuparle. El hecho de que estemos definiendo procedimientos tan similares para flujos y listas indica que nos falta alguna abstracción subyacente. Desafortunadamente, para explotar esta abstracción, necesitaremos ejercer un control más preciso sobre el proceso de evaluación del que podemos ejercer actualmente. Analizaremos este punto más a fondo al final de la sección [3.5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.4). En la sección [4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2), desarrollaremos un marco que unifique listas y flujos.

[56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_446) Aunqueel carro de flujoystream-cdr puede definirse como procedimientos, cons-stream debe ser una forma especial. Si cons-stream fuera un procedimiento, entonces, según nuestro modelo de evaluación, evaluar (cons-stream < *a* > < *b* >) causaría automáticamente que se evaluara < *b* >, que es precisamente lo que no queremos que suceda. Por la misma razón, delay debe ser una forma especial, aunque force puede ser un procedimiento ordinario.

[57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_448) Los números que se muestran aquí no aparecen realmente en la expresión retrasada. Lo que aparece en realidad es la expresión original, en un entorno en el que las variables están vinculadas a los números adecuados. Por ejemplo,(+ low 1)coninferiora 10 000 aparece en realidad dondese muestra10 001

[58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_450) Existen muchas posibles implementaciones de flujos además de la descrita en esta sección. La evaluación retrasada, que es la clave para que los flujos sean prácticos, era inherente a*Método de paso de parámetros por nombre* de Algol 60. El uso de este mecanismo para implementar flujos fue descrito por primera vez porLandin (1965). La evaluación retrasada de los flujos de datos se introdujo en Lisp porFriedman y Wise (1976). En su implementación, cons siempre retrasa la evaluación de sus argumentos, de modo que las listas se comportan automáticamente como secuencias. La optimización de memorización también se conoce como*llamada por necesidad* . La comunidad de Algol se referiría a nuestros objetos retrasados ​​originales como *thunks de llamada por nombre* y a las versiones optimizadas como *thunks de llamada por necesidad* .

[59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_453) Ejercicios como [los 3.51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.51)y [3.52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.52)son valiosos para poner a prueba nuestra comprensión de cómoel retraso. Por otra parte, mezclar la evaluación diferida con la impresión -y, peor aún, con la tarea- es extremadamente confuso, y los instructores de cursos sobre lenguajes informáticos tradicionalmente han atormentado a sus estudiantes con preguntas de examen como las de esta sección. No hace falta decir que escribir programas que dependen de tales sutilezas esEstilo de programación odioso. Parte del poder del procesamiento de flujos es que nos permite ignorar el orden en el que realmente suceden los eventos en nuestros programas. Desafortunadamente, esto es precisamente lo que no podemos permitirnos hacer en presencia de la asignación, que nos obliga a preocuparnos por el tiempo y el cambio.

[60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_455) Eratóstenes, sigloIIIa.C.C.​El filósofo griego alejandrino es famoso por haber dado la primera estimación precisa de la circunferencia de la Tierra, que calculó observando las sombras proyectadas al mediodía el día del solsticio de verano. El método de la criba de Eratóstenes, aunque antiguo, ha formado la base de los "cribos" de hardware de propósito especial que, hasta hace poco, eran las herramientas más poderosas que existían para localizar números primos grandes. Sin embargo, desde los años 70, estos métodos han sido reemplazados por desarrollos deltécnicas probabilísticas discutidas en la sección  [1.2.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.6) .

[61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_456) Hemos bautizado estas figuras con el nombre dePeter Henderson, quien fue la primera persona que nos mostró diagramas de este tipo como una forma de pensar en el procesamiento de flujos. Cada línea sólida representa un flujo de valores que se transmiten. La línea discontinua desde el automóvil hasta los contras y el filtro indica que se trata de un valor único en lugar de un flujo.

[62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_458) Esto utiliza la versión generalizada destream-mapdel ejercicio [3.50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.50).

[63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_459) Este último punto es muy sutil y se basa en el hecho de que*p n* + 1 < *p n* 2. (Aquí,*p k* denota el*k*-ésimo). Estimaciones como estas son muy difíciles de establecer. La antigua prueba deEuclides demostró que hay un número infinito de primos, lo que demuestra que *p n* + 1 < *p* 1 *p* 2 Â·Â·Â· *p n* + 1, y no se demostró ningún resultado sustancialmente mejor hasta 1851, cuando el matemático ruso PL Chebyshev establecióque *p n* +1 < 2 *p n* para todo *n* . Este resultado, conjeturado originalmente en 1845, se conoce como *la hipótesis de Bertrand* . Se puede encontrar una prueba en la sección 22.3 de Hardy y Wright 1960.

[64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_465) Este ejercicio muestra cómo la llamada por necesidad está estrechamente relacionada conmemorización ordinaria como se describe en el ejercicio  [3.27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.27) . En ese ejercicio, usamos la asignación para construir explícitamente una tabla local. Nuestra optimización de flujo por llamada según necesidad construye efectivamente dicha tabla automáticamente, almacenando valores en las partes previamente forzadas del flujo.

No podemos usarletpara vincular la variable localguesses, porque el valor deguessesdepende deguessesen sí. El ejercicio [3.63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.63) [explica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_472) por qué queremos una variable local aquí.

[66](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_477) Como en la sección [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3), representamos un par de números enteros como una lista en lugar de un par Lisp.

Consulte el ejercicio [3.68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.68) [para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_478) obtener una idea de por qué elegimos esta descomposición.

[68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_479) La declaración precisa de la propiedad requerida sobre el orden de combinación es la siguiente: debe haber una función*f*de dos argumentos tal que el par correspondiente al elemento *i*del primer flujo y el elemento *j*del segundo flujo aparezca como el elemento número*f*(*i*,*j*) del flujo de salida. El truco de usarintercalarpara lograr esto nos lo mostróDavid Turner, quien lo empleó en el lenguajeRevista de Ciencias de la Computación (Turner 1981).

Requeriremos que la función de ponderación sea tal que el peso de un par aumente a medida que nos movemos a lo largo de una fila o hacia abajo a lo largo de una columna de la matriz de pares [.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_485)

[70](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_487) Para citar el obituario de GH Hardy deRamanujan (Hardy 1921): “Creo que fue el señor Littlewood quien comentó que “todo entero positivo era uno de sus amigos”. Recuerdo que una vez fui a verlo cuando estaba enfermo en Putney. Yo había viajado en el taxi número 1729 y comenté que el número me parecía bastante aburrido y que esperaba que no fuera un mal presagio. “No”, respondió, “es un número muy interesante; es el número más pequeño que se puede expresar como la suma de dos cubos de dos maneras diferentes”. Charles Leiserson nos mostró el truco de usar pares ponderados para generar los números de Ramanujan.

[No se garantiza](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_494) que este procedimiento funcione en todas las implementaciones de Scheme, aunque existe una variación sencilla que funcionará en cualquier implementación. El problema tiene que ver con diferencias sutiles en las formas en que las implementaciones de Scheme manejan las definiciones internas. (Consulte la sección [4.1.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.6)).

[72](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_500) Esta es una pequeña reflexión, en Lisp, de las dificultades que presentan los lenguajes fuertemente tipados convencionales comoPascal tiene que lidiar con procedimientos de orden superior. En tales lenguajes, el programador debe especificar los tipos de datos de los argumentos y el resultado de cada procedimiento: número, valor lógico, secuencia, etc. En consecuencia, no podríamos expresar una abstracción como ``mapear un procedimiento dado proc sobre todos los elementos de una secuencia'' mediante un único procedimiento de orden superior como stream-map . En lugar de ello, necesitaríamos un procedimiento de mapeo diferente para cada combinación diferente de tipos de datos de argumentos y resultados que pudieran especificarse para un proc . Mantener una noción práctica de ``tipo de datos'' en presencia de procedimientos de orden superior plantea muchas cuestiones difíciles. Una forma de abordar este problema se ilustra con el lenguaje ML.(Gordon, Milner y Wadsworth 1979), cuyos "tipos de datos polimórficos" incluyen plantillas para transformaciones de orden superior entre tipos de datos. Además, los tipos de datos para la mayoría de los procedimientos en ML nunca son declarados explícitamente por el programador. En cambio, ML incluye un*mecanismo de inferencia de tipos* que utiliza información del entorno para deducir los tipos de datos para procedimientos recién definidos.

[73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_504) De manera similar, en física, cuando observamos una partícula en movimiento, decimos que la posición (estado) de la partícula está cambiando. Sin embargo, desde la perspectiva de la partículalínea del mundo en el espacio-tiempo no hay ningún cambio involucrado.

[74](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_505) John Backus, el inventor de Fortran, dio gran importanciaLa programación funcional se hizo visible cuando recibió el premio ACM Turing en 1978. En su discurso de aceptación (Backus 1978) defendió firmemente el enfoque funcional. En Henderson 1980 y en Darlington, Henderson y Turner 1982 se ofrece una buena descripción general de la programación funcional.

[75](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_506) Obsérvese que, para dos corrientes cualesquiera, en general hay más de unaorden aceptable de intercalación. Por lo tanto, técnicamente, "merge" es una relación en lugar de una función: la respuesta no es una función determinista de las entradas. Ya mencionamos (nota al pie  [39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#footnote_Temp_411) ) que el no determinismo es esencial cuando se trata de concurrencia. La relación de fusión ilustra el mismo no determinismo esencial, desde la perspectiva funcional. En la sección  [4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3) , analizaremos el no determinismo desde otro punto de vista.

[76](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_507) El modelo de objetos se aproxima al mundo dividiéndolo en partes separadas. El modelo funcional no se modulariza a lo largo de los límites de los objetos. El modelo de objetos es útil cuandoEl estado no compartido de los "objetos" es mucho mayor que el estado que comparten. Un ejemplo de un lugar donde el punto de vista del objeto falla esLa mecánica cuántica, en la que pensar en las cosas como partículas individuales conduce a paradojas y confusiones. Unificar la visión del objeto con la visión funcional puede tener poco que ver con la programación, sino más bien con cuestiones epistemológicas fundamentales.

[**Capítulo 4**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_4)

[**Abstracción metalingüística**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_4)

|  |
| --- |
| ... La magia está en las palabras: Abracadabra, Ábrete Sésamo y las demás, pero las palabras mágicas de una historia no lo son en la siguiente. La verdadera magia consiste en entender qué palabras funcionan, cuándo y para qué; el truco consiste en aprender el truco.  ... Y esas palabras están formadas por las letras de nuestro alfabeto: un par de docenas de garabatos que podemos dibujar con el bolígrafo. ¡Ésa es la clave! ¡Y el tesoro también, si tan solo pudiéramos conseguirlo! ¡Es como si... como si la clave del tesoro *fuera* el tesoro!  John Barth, *Quimera* |

En nuestro estudio del diseño de programas, hemos visto que los programadores expertos controlan la complejidad de sus diseños con las mismas técnicas generales que utilizan los diseñadores de todos los sistemas complejos. Combinan elementos primitivos para formar objetos compuestos, abstraen objetos compuestos para formar bloques de construcción de nivel superior y preservan la modularidad adoptando vistas apropiadas a gran escala de la estructura del sistema. Para ilustrar estas técnicas, hemos utilizado Lisp como lenguaje para describir procesos y para construir objetos y procesos de datos computacionales para modelar fenómenos complejos en el mundo real. Sin embargo, a medida que nos enfrentamos a problemas cada vez más complejos, descubriremos que Lisp, o de hecho cualquier lenguaje de programación fijo, no es suficiente para nuestras necesidades. Debemos recurrir constantemente a nuevos lenguajes para expresar nuestras ideas de manera más efectiva. Establecer nuevos lenguajes es una estrategia poderosa para controlar la complejidad en el diseño de ingeniería; a menudo podemos mejorar nuestra capacidad para abordar un problema complejo adoptando un nuevo lenguaje que nos permita describir (y, por lo tanto, pensar en) el problema de una manera diferente, utilizando primitivos, medios de combinación y medios de abstracción que sean particularmente adecuados para el problema en cuestión. [1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_508)

La programación está dotada de una multitud de lenguajes. Existen lenguajes físicos, como los lenguajes de máquina para computadoras particulares. Estos lenguajes se ocupan de la representación de datos y el control en términos de bits individuales de almacenamiento e instrucciones primitivas de máquina. El programador de lenguaje de máquina se ocupa de utilizar el hardware dado para construir sistemas y utilidades para la implementación eficiente de cálculos con recursos limitados. Los lenguajes de alto nivel, erigidos sobre un sustrato de lenguaje de máquina, esconden preocupaciones sobre la representación de datos como colecciones de bits y la representación de programas como secuencias de instrucciones primitivas. Estos lenguajes tienen medios de combinación y abstracción, como la definición de procedimientos, que son apropiados para la organización de sistemas a gran escala.

*La abstracción metalingüística* (establecer nuevos lenguajes) desempeña un papel importante en todas las ramas del diseño de ingeniería. Es particularmente importante para la programación informática, porque en la programación no sólo podemos formular nuevos lenguajes, sino que también podemos implementar estos lenguajes mediante la construcción de evaluadores.*Un evaluador* (o *intérprete* ) de un lenguaje de programación es un procedimiento que, cuando se aplica a una expresión del lenguaje, realiza las acciones necesarias para evaluar esa expresión.

No es exagerado considerar que ésta es la idea más fundamental en programación:

El evaluador, que determina el significado de las expresiones en un lenguaje de programación, es simplemente otro programa.

Para entender este punto es necesario cambiar la imagen que tenemos de nosotros mismos como programadores. Empezamos a vernos como diseñadores de lenguajes, en lugar de ser meros usuarios de lenguajes diseñados por otros.

De hecho, podemos considerar casi cualquier programa como el evaluador de algún lenguaje. Por ejemplo, el sistema de manipulación de polinomios de la sección  [2.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.5.3) incorpora las reglas de la aritmética polinómica y las implementa en términos de operaciones sobre datos estructurados en listas. Si ampliamos este sistema con procedimientos para leer e imprimir expresiones polinómicas, tenemos el núcleo de un lenguaje de propósito especial para tratar problemas de matemáticas simbólicas. El simulador de lógica digital de la sección  [3.3.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.4) y el propagador de restricciones de la sección  [3.3.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.5) son lenguajes legítimos por derecho propio, cada uno con sus propios primitivos, medios de combinación y medios de abstracción. Visto desde esta perspectiva, la tecnología para hacer frente a los sistemas informáticos a gran escala se fusiona con la tecnología para construir nuevos lenguajes informáticos, yLa informática en sí misma se convierte en nada más (ni menos) que la disciplina de construir lenguajes descriptivos apropiados.

A continuación, nos embarcaremos en un recorrido por la tecnología mediante la cual se establecen los lenguajes en función de otros lenguajes. En este capítulo utilizaremos Lisp como base, implementando los evaluadores como procedimientos Lisp.Lisp es particularmente adecuado para esta tarea, debido a su capacidad para representar y manipular expresiones simbólicas. Daremos el primer paso para entender cómo se implementan los lenguajes construyendo un evaluador para el propio Lisp. El lenguaje implementado por nuestro evaluador será un subconjunto del dialecto Scheme de Lisp que utilizamos en este libro. Aunque el evaluador descrito en este capítulo está escrito para un dialecto particular de Lisp, contiene la estructura esencial de un evaluador para cualquier lenguaje orientado a expresiones diseñado para escribir programas para una máquina secuencial. (De hecho, la mayoría de los procesadores de lenguaje contienen, en su interior, un pequeño evaluador «Lisp».) El evaluador se ha simplificado con fines ilustrativos y de discusión, y se han omitido algunas características que serían importantes incluir en un sistema Lisp de calidad de producción. No obstante, este evaluador simple es adecuado para ejecutar la mayoría de los programas de este libro. [2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_509)

Una ventaja importante de hacer accesible el evaluador como un programa Lisp es que podemos implementar reglas de evaluación alternativas describiéndolas como modificaciones al programa evaluador. Un lugar donde podemos usar este poder con buenos resultados es para obtener un control adicional sobre las formas en que los modelos computacionales incorporan la noción de tiempo, que fue tan central para la discusión en el capítulo 3. Allí, mitigamos algunas de las complejidades del estado y la asignación al usar flujos para desacoplar la representación del tiempo en el mundo del tiempo en la computadora. Nuestros programas de flujo, sin embargo, a veces eran engorrosos, porque estaban restringidos por la evaluación de orden aplicativo de Scheme. En la sección  [4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2) , cambiaremos el lenguaje subyacente para proporcionar un enfoque más elegante, modificando el evaluador para proporcionar una *evaluación de orden normal* .

La sección  [4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3) implementa un cambio lingüístico más ambicioso, por el cual las expresiones tienen muchos valores, en lugar de un solo valor. En este lenguaje de *computación no determinista* , es natural expresar procesos que generan todos los valores posibles para las expresiones y luego buscar aquellos valores que satisfacen ciertas restricciones. En términos de modelos de computación y tiempo, esto es como hacer que el tiempo se ramifique en un conjunto de "futuros posibles" y luego buscar líneas de tiempo apropiadas. Con nuestro evaluador no determinista, el seguimiento de múltiples valores y la realización de búsquedas se manejan automáticamente por el mecanismo subyacente del lenguaje.

En la sección  [4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4) implementamos un lenguaje *de programación lógica* en el que el conocimiento se expresa en términos de relaciones, en lugar de en términos de cálculos con entradas y salidas. Aunque esto hace que el lenguaje sea drásticamente diferente de Lisp, o de cualquier lenguaje convencional, veremos que el evaluador de programación lógica comparte la estructura esencial del evaluador de Lisp.

[1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_508) La misma idea está presente en toda la ingeniería. Por ejemplo, los ingenieros eléctricos utilizan muchos lenguajes diferentes para describir circuitos. Dos de ellos son el lenguaje de*redes*y el lenguaje de*sistemas*. El lenguaje de redes enfatiza el modelado físico de dispositivos en términos de elementos eléctricos discretos. Los objetos primitivos del lenguaje de redes son componentes eléctricos primitivos como resistencias, capacitores, inductores y transistores, que se caracterizan en términos de variables físicas llamadas voltaje y corriente. Al describir circuitos en el lenguaje de redes, el ingeniero se preocupa por las características físicas de un diseño. En contraste, los objetos primitivos del lenguaje de sistemas son módulos de procesamiento de señales como filtros y amplificadores. Solo es relevante el comportamiento funcional de los módulos, y las señales se manipulan sin preocuparse por su realización física como voltajes y corrientes. El lenguaje de sistemas se erige sobre el lenguaje de redes, en el sentido de que los elementos de los sistemas de procesamiento de señales se construyen a partir de redes eléctricas. Aquí, sin embargo, las preocupaciones son con la organización a gran escala de dispositivos eléctricos para resolver un problema de aplicación dado; Se supone la viabilidad física de las partes. Esta colección de lenguajes en capas es otro ejemplo de la técnica de diseño estratificado ilustrada por el lenguaje de imágenes de la sección [2.2.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.4).

[2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_509) Las características más importantes que nuestro evaluador omite son los mecanismos para gestionar errores y respaldar la depuración. Para una discusión más extensa de los evaluadores, consulteFriedman, Wand y Haynes 1992, que ofrece una exposición de los lenguajes de programación que procede a través de una secuencia de evaluadores escritos en Scheme.

[**4.1 El evaluador metacircular**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_4.1)

Nuestro evaluador para Lisp se implementará como un programa Lisp. Puede parecer circular pensar en evaluar programas Lisp utilizando un evaluador que esté implementado en Lisp. Sin embargo, la evaluación es un proceso, por lo que es apropiado describir el proceso de evaluación utilizando Lisp, que, después de todo, es nuestra herramienta para describir procesos. [3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_510) Un evaluador que esté escrito en el mismo lenguajeque evalúa se dice que es *metacircular* .

El evaluador metacircular es esencialmente una formulación de Scheme del modelo de evaluación del entorno descrito en la sección  [3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.2) . Recordemos que el modelo tiene dos partes básicas:

1. Para evaluar una combinación (una expresión compuesta que no sea una forma especial), evalúe las subexpresiones y luego aplique el valor de la subexpresión del operador a los valores de las subexpresiones del operando.

2. Para aplicar un procedimiento compuesto a un conjunto de argumentos, evalúe el cuerpo del procedimiento en un nuevo entorno. Para construir este entorno, extienda la parte del entorno del objeto de procedimiento mediante un marco en el que los parámetros formales del procedimiento estén vinculados a los argumentos a los que se aplica el procedimiento.

Estas dos reglas describen la esencia del proceso de evaluación, un ciclo básico en el que las expresiones a evaluar en los entornos se reducen a procedimientos a aplicar a los argumentos, que a su vez se reducen a nuevas expresiones a evaluar en nuevos entornos, y así sucesivamente, hasta llegar a los símbolos, cuyos valores se buscan en el entorno, y a los procedimientos primitivos, que se aplican directamente (véase la figura  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.1) ). [4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_511) Este ciclo de evaluación se materializará en la interacción entre los dos procedimientos críticos del evaluador, eval y apply , que se describen en la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) (véase la figura  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.1) ).

La implementación del evaluador dependerá de procedimientos que definan la *sintaxis* de las expresiones a evaluar. Usaremosabstracción de datos para hacer que el evaluador sea independiente de la representación del lenguaje. Por ejemplo, en lugar de comprometernos con una elección de que una asignación se represente mediante una lista que comience con el conjunto de símbolos! usamos un predicado abstracto assignment? para probar una asignación, y usamos los selectores abstractos assignment-variable y assignment-value para acceder a las partes de una asignación. La implementación de expresiones se describirá en detalle en la sección  [4.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.2) . También hay operaciones, descritas en la sección  [4.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.3) , que especifican la representación de procedimientos y entornos. Por ejemplo, make-procedure construye procedimientos compuestos, lookup-variable-value accede a los valores de las variables y apply-primitive-procedure aplica un procedimiento primitivo a una lista dada de argumentos.

**[4.1.1 El núcleo del evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.1.1)**

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 4.1:**   El ciclo eval - apply expone la esencia de un lenguaje de computadora. |

El proceso de evaluación puede describirse como la interacción entre dos procedimientos: evaluar y aplicar .

**[Evaluar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_512)**

Eval toma como argumentos una expresión y un entorno. Clasifica la expresión y dirige su evaluación. Eval está estructurado como un análisis de caso del tipo sintáctico de la expresión a evaluar. Para mantener el procedimiento general, expresamos la determinación del tipo de una expresión de forma abstracta, sin comprometernos con ningún parámetro en particular.Representación de los distintos tipos de expresiones. Cada tipo de expresión tiene un predicado que lo prueba y un medio abstracto para seleccionar sus partes.*La sintaxis abstracta* permite ver fácilmente cómo podemos cambiar la sintaxis del lenguaje utilizando el mismo evaluador, pero con una colección diferente de procedimientos de sintaxis.

**[Expresiones primitivas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_513)**

* Para expresiones de autoevaluación, como números, eval devuelve la expresión en sí.
* Eval debe buscar variables en el entorno para encontrar sus valores.

**[Formularios especiales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_514)**

* Para las expresiones entre comillas, eval devuelve la expresión que fue entre comillas.
* Una asignación a (o una definición de) una variable debe llamar recursivamente a eval para calcular el nuevo valor que se asociará con la variable. El entorno debe modificarse para cambiar (o crear) la vinculación de la variable.
* Una expresión if requiere un procesamiento especial de sus partes, para evaluar el consecuente si el predicado es verdadero, y en caso contrario para evaluar la alternativa.
* Una expresión lambda debe transformarse en un procedimiento aplicable empaquetando juntos los parámetros y el cuerpo especificados por la expresión lambda con el entorno de la evaluación.
* Una expresión de inicio requiere evaluar su secuencia de expresiones en el orden en que aparecen.
* Un análisis de caso ( cond ) se transforma en un nido de expresiones if y luego se evalúa.

**[Combinaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_515)**

* Para una aplicación de procedimiento, eval debe evaluar de forma recursiva la parte del operador y los operandos de la combinación. El procedimiento y los argumentos resultantes se pasan a apply , que maneja la aplicación del procedimiento real.

Aquí está la definición de eval :

(define (eval exp env)   
  (cond ((¿autoevaluación? exp) exp)   
        ((variable? exp) (valor-variable-búsqueda exp env))   
        ((¿entrecomillado? exp) (texto-de-entrecomillado exp))   
        ((asignación? exp) (asignación-eval exp env))   
        ((definición? exp) (definición-eval exp env))   
        ((si? exp) (eval-if exp env))   
        ((lambda? exp)   
         (procedimiento-make (parámetros-lambda exp)   
                         (cuerpo-lambda exp)   
                         env))   
        ((comienzo? exp)    
         (secuencia-eval (acciones-comienzo exp) env))   
        ((cond? exp) (eval (cond->if exp) env))   
        ((aplicación? exp)   
         (aplicar (eval (operador exp) env)   
                (lista-de-valores (operandos exp) env)))   
        (else   
         (error "Tipo de expresión desconocido -- EVAL" exp))))

Para mayor claridad, eval se ha implementado como un análisis de caso utilizando cond . La desventaja de esto es que nuestro procedimiento maneja solo unos pocos tipos distinguibles de expresiones, y no se pueden definir nuevos tipos sin editar la definición de eval . En la mayoría de las implementaciones de Lisp, el envío del tipo de una expresión se realiza en un estilo dirigido por datos. Esto permite que un usuario agregue nuevos tipos de expresiones que eval puede distinguir, sin modificar la definición de eval en sí. (Ver ejercicio  [4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.3) .)

**[Aplicar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_516)**

Apply toma dos argumentos, un procedimiento y una lista de argumentos a los que se debe aplicar el procedimiento. Apply clasifica los procedimientos en dos tipos: Llamaapply-primitive-procedure para aplicar primitivos; aplica procedimientos compuestos evaluando secuencialmente las expresiones que forman el cuerpo del procedimiento. El entorno para la evaluación del cuerpo de un procedimiento compuesto se construye extendiendo el entorno base que lleva el procedimiento para incluir un marco que vincula los parámetros del procedimiento a los argumentos a los que se aplicará el procedimiento. Aquí está la definición de apply :

(define (procedimiento aplicar argumentos)   
  (cond ((procedimiento-primitivo? procedimiento)   
         (procedimiento-aplicar-procedimiento-primitivo argumentos))   
        ((procedimiento-compuesto? procedimiento)   
         (secuencia-eval   
           (cuerpo-procedimiento procedimiento)   
           (entorno-extendido   
             (parámetros-procedimiento procedimiento)   
             argumentos   
             (entorno-procedimiento procedimiento))))   
        (de lo contrario   
         (error   
          "Tipo de procedimiento desconocido -- APLICAR" procedimiento))))

**[Argumentos de procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_517)**

Cuando eval procesa una aplicación de procedimiento, utiliza una lista de valores para generar la lista de argumentos a los que se aplicará el procedimiento. La lista de valores toma como argumento los operandos de la combinación. Evalúa cada operando y devuelve una lista de los valores correspondientes: [5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_518)

(define (lista-de-valores exps env)   
  (if (¿sin-operandos? exps)   
      '()   
      (cons (eval (primer-operando exps) env)   
            (lista-de-valores (resto-operandos exps) env))))

**[Condicionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_519)**

Eval-if evalúa la parte predicativa de una expresión if en el entorno dado. Si el resultado es verdadero, eval-if evalúa el consecuente; de ​​lo contrario, evalúa la alternativa:

(define (eval-if exp env)   
  (if (¿verdadero? (eval (if-predicado exp) env))   
      (eval (if-consecuente exp) env)   
      (eval (if-alternativa exp) env)))

El uso de true? en eval-if resalta el problema de la conexión entre un lenguaje implementado y un lenguaje de implementación. El predicado if se evalúa en el lenguaje que se está implementando y, por lo tanto, produce un valor en ese lenguaje. El predicado del intérprete true? traduce ese valor en un valor que puede probarse mediante el if en el lenguaje de implementación: La representación metacircular de la verdad podría no ser la misma que la del Scheme subyacente. [6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_520)

**[Secuencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_521)**

Eval-sequence se utiliza en apply para evaluar la secuencia de expresiones en el cuerpo de un procedimiento y en eval para evaluar la secuencia de expresiones en una expresión de inicio . Toma como argumentos una secuencia de expresiones y un entorno, y evalúa las expresiones en el orden en que aparecen. El valor devuelto es el valor de la expresión final.

(define (eval-secuencia exps env)   
  (cond ((ultima exp? exps) (eval (primera exp exps) env))   
        (else (eval (primera exp exps) env)   
              (eval-secuencia (resto exps exps) env))))

**[Tareas y definiciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_522)**

El siguiente procedimiento maneja las asignaciones a variables. Llama a eval para encontrar el valor que se asignará y transmite la variable y el valor resultante a set-variable-value! para que se instale en el entorno designado.

(define (eval-asignación exp env)   
  (establecer-valor-variable! (asignación-variable exp)   
                       (eval (asignación-valor exp) env)   
                       env)   
  'ok)

Las definiciones de variables se manejan de manera similar. [7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_523)

(define (definición-eval exp env)   
  (variable-definición! (variable-definición exp)   
                    (eval (valor-definición exp) env)   
                    env)   
  'ok)

Hemos elegido aquí devolver el símbolo ok como valor de una asignación o una definición [.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_524)

**Ejercicio 4.1.**  Tenga en cuenta que no podemos determinar si el evaluador metacircular evalúa los operandos de izquierda a derecha o de derecha a izquierda. Su orden de evaluación se hereda del Lisp subyacente: si los argumentos de cons en list-of-values ​​se evalúan de izquierda a derecha, list-of-values ​​evaluará los operandos de izquierda a derecha; y si los argumentos de cons se evalúan de derecha a izquierda, list-of-values ​​evaluará los operandos de derecha a izquierda.

Escriba una versión de lista de valores que evalúe los operandos de izquierda a derecha independientemente del orden de evaluación en el Lisp subyacente. Escriba también una versión de lista de valores que evalúe los operandos de derecha a izquierda.

**[4.1.2 Representación de expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.1.2)**

El evaluador recuerda al programa de diferenciación simbólica analizado en la sección  [2.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.2) . Ambos programas operan con expresiones simbólicas. En ambos programas, el resultado de operar con una expresión compuesta se determina operando recursivamente sobre las partes de la expresión y combinando los resultados de una manera que depende del tipo de expresión. En ambos programas usamosabstracción de datos para disociar las reglas generales de operación de los detalles de cómo se representan las expresiones. En el programa de diferenciación, esto significaba que el mismo procedimiento de diferenciación podía tratar expresiones algebraicas en forma de prefijo, en forma de infijo o en alguna otra forma. Para el evaluador, esto significa que la sintaxis del lenguaje que se evalúa está determinada únicamente por los procedimientos que clasifican y extraen fragmentos de expresiones.

Aquí está la especificación de la sintaxis de nuestro lenguaje:

¤ Los únicos elementos que se autoevaluan son los números y las cadenas:

(define (¿autoevaluable? exp)   
  (cond ((¿número? exp) verdadero)   
        ((¿cadena? exp) verdadero)   
        (de lo contrario falso)))

Â¤ Las variables se representan mediante símbolos:

(definir (variable? exp) (símbolo? exp))

Â¤ Las citas tienen la forma (cita < *texto-de-la-cita* >) : [9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_526)

(define (¿citado? exp)   
  (¿lista etiquetada? exp 'cita'))  
  
(define (texto de cita exp) (cadr exp))

¿Citado? se define en términos del procedimiento tagged-list?, que identifica listas que comienzan con un símbolo designado:

(define (lista-etiquetada? exp etiqueta)   
  (si (par? exp)   
      (ecuación? (automóvil exp) etiqueta)   
      falso))

Â¤ Las asignaciones tienen la forma (set! < *var* > < *valor* >) :

(define (asignación? exp)   
  (lista-etiquetada? exp 'conjunto!))  
(define (variable de asignación exp) (cadr exp))  
(define (exp valor-asignación) (exp caddr))

Â¤ Las definiciones tienen la forma

*(* define *<var>* <valor> )

o la forma

(definir (< *var* > < *parámetro 1* > ... < *parámetro n* > )   
  < *cuerpo* >)

La última forma (definición de procedimiento estándar) es azúcar sintáctico para

(definir <var> *(*  
  lambda (< *parámetro 1 >* ... < parámetro *n >* )   
    <cuerpo> ) )

Los procedimientos de sintaxis correspondientes son los siguientes:

(define (definición? exp)   
  (lista-etiquetada? exp 'define))  
(define (variable-definición exp)   
  (si (símbolo? (cadr exp))   
      (cadr exp)   
      (caadr exp)))  
(define (valor-definición exp)   
  (si (símbolo? (cadr exp))   
      (caddr exp)   
      (make-lambda (cdadr exp)    *; parámetros formales*  
                   (cddr exp))))  *; cuerpo*

Â¤ Las expresiones lambda son listas que comienzan con el símbolo lambda :

(define (lambda? exp) (lista-etiquetada? exp 'lambda))  
(define (parámetros lambda exp) (cadr exp))  
(define (lambda-body exp) (cddr exp))

También proporcionamos un constructor para expresiones lambda , que se utiliza mediante definition-value , arriba:

(define (make-lambda parámetros cuerpo)   
  (cons 'lambda (cons parámetros cuerpo)))

Â¤ Los condicionales comienzan con if y tienen un predicado, un consecuente y una alternativa (opcional). Si la expresión no tiene una parte alternativa, proporcionamos false como la alternativa. [10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_527)

(define (si? exp) (lista-etiquetada? exp 'si))  
(define (si-predicado exp) (cadr exp))  
(define (si es consecuente exp) (caddr exp))  
(define (if-alternative exp)   
  (if (no (nulo? (cdddr exp)))   
      (cadddr exp)   
      'falso))

También proporcionamos un constructor para expresiones if , que será utilizado por cond->if para transformar expresiones cond en expresiones if :

(define (make-if predicado consecuente alternativa)   
  (lista 'if predicado consecuente alternativa))

Â¤ Begin empaqueta una secuencia de expresiones en una sola expresión. Incluimos operaciones de sintaxis en las expresiones de inicio para extraer la secuencia real de la expresión de inicio , así como selectores que devuelven la primera expresión y el resto de las expresiones en la secuencia. [11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_528)

(define (begin? exp) (tagged-list? exp 'begin))  
(define (begin-actions exp) (cdr exp))  
(define (last-exp? seq) (nulo? (cdr seq)))  
(define (primera secuencia de experiencias) (secuencia de autos))  
(define (resto-exp secuencia) (cdr secuencia))

También incluimos un constructor sequence->exp (para uso de cond->if ) que transforma una secuencia en una sola expresión, usando begin si es necesario:

(define (secuencia->exp seq)   
  (cond ((¿nulo? seq) seq)   
        ((¿última-exp? seq) (primera-exp seq))   
        (sino (make-begin seq))))  
(define (make-begin seq) (cons 'begin seq))

Â¤ Una aplicación de procedimiento es cualquier expresión compuesta que no sea uno de los tipos de expresión anteriores. El car de la expresión es el operador y el cdr es la lista de operandos:

(define (aplicación? exp) (par? exp))  
(define (operador exp) (coche exp))  
(define (operandos exp) (cdr exp))  
(define (¿sin operandos? operaciones) (¿nulos? operaciones))  
(define (operaciones del primer operando) (operaciones del carro))  
(define (operaciones de resto) (operaciones de cdr))

**[Expresiones derivadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_529)**

Algunas formas especiales de nuestro lenguaje se pueden definir en términos de expresiones que involucran otras formas especiales, en lugar de implementarse directamente. Un ejemplo es cond , que se puede implementar como un anidamiento de expresiones if . Por ejemplo, podemos reducir el problema de evaluar la expresión

(cond ((> x 0) x)   
      ((= x 0) (mostrar 'cero) 0)   
      (de lo contrario (- x)))

al problema de evaluar la siguiente expresión que involucra las expresiones if y begin :

(si (> x 0)   
    x   
    (si (= x 0)   
        (comienza (muestra 'cero)   
               0)   
        (- x)))

Implementar la evaluación de cond de esta manera simplifica al evaluador porque reduce el número de formularios especiales para los cuales el proceso de evaluación debe especificarse explícitamente.

Incluimos procedimientos de sintaxis que extraen las partes de una expresión cond y un procedimiento cond->if que transforma expresiones cond en expresiones if . Un análisis de caso comienza con cond y tiene una lista de cláusulas predicado-acción. Una cláusula es una cláusula else si su predicado es el símbolo else . [12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_530)

(define (cond? exp) (lista-etiquetada? exp 'cond))  
(define (cláusulas cond. exp) (cdr exp))  
(define (cláusula cond-else? cláusula)   
  (eq? (cláusula cond-predicado) 'else))  
(define (cláusula de predicado condicional) (cláusula de automóvil))  
(define (cláusula de cond-acciones) (cláusula cdr))  
(define (cond->if exp)   
  (cláusulas-expandidas (cláusulas-cond exp)))  
  
(define (cláusulas-expandidas)   
  (if (cláusulas-nulas)   
      'false                           *; no hay   cláusula else*  
      (let ((first (cláusulas-car))   
            (rest (cláusulas-cdr)))   
        (if (cláusula-cond-else? first)   
            (if (cláusula-nulas? rest)   
                (secuencia->exp (acciones-cond primero))   
                (error "la cláusula ELSE no es la última --   
                       cláusulas COND->IF"))   
            (make-if (predicado-cond primero)   
                     (secuencia->exp (acciones-cond primero))   
                     (cláusulas-expandidas resto))))))

Las expresiones (como cond ) que elegimos implementar como transformaciones sintácticas se denominan *expresiones derivadas* . Las expresiones let también son expresiones derivadas (ver ejercicio  [4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.6) ). [13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_531)

**Ejercicio 4.2.**  Louis Reasoner planea reordenar las cláusulas cond en eval de modo que la cláusula para aplicaciones de procedimientos aparezca antes de la cláusula para asignaciones. Sostiene que esto hará que el intérprete sea más eficiente: dado que los programas suelen contener más aplicaciones que asignaciones, definiciones, etc., su eval modificado generalmente verificará menos cláusulas que el eval original antes de identificar el tipo de una expresión.

a. ¿Qué hay de malo en el plan de Louis? (Pista: ¿Qué hará el evaluador de Louis con la expresión (define x 3) ?)

b. Louis está molesto porque su plan no funcionó. Está dispuesto a hacer lo que sea para que su evaluador reconozca las aplicaciones de procedimiento antes de que verifique la mayoría de los otros tipos de expresiones. Ayúdelo cambiando la sintaxis del lenguaje evaluado de modo que las aplicaciones de procedimiento comiencen con call . Por ejemplo, en lugar de (factorial 3) ahora tendremos que escribir (call factorial 3) y en lugar de (+ 1 2) tendremos que escribir (call + 1 2) .

**Ejercicio 4.3.**  Reescriba eval de modo que el envío se realice en un estilo dirigido por datos. Compare esto con el procedimiento de diferenciación dirigido por datos del ejercicio  [2.73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.73) . (Puede utilizar el carro de una expresión compuesta como el tipo de la expresión, según corresponda a la sintaxis implementada en esta sección).

**Ejercicio 4.4.**  Recuerde las definiciones de las formas especiales y y o del capítulo 1:

* y : Las expresiones se evalúan de izquierda a derecha. Si alguna expresión se evalúa como falsa, se devuelve falso; las expresiones restantes no se evalúan. Si todas las expresiones se evalúan como valores verdaderos, se devuelve el valor de la última expresión. Si no hay expresiones, se devuelve verdadero.
* o : Las expresiones se evalúan de izquierda a derecha. Si alguna expresión se evalúa como un valor verdadero, se devuelve ese valor; las expresiones restantes no se evalúan. Si todas las expresiones se evalúan como falsas, o si no hay expresiones, se devuelve falso.

Instalar y y o como nuevas formas especiales para el evaluador definiendo procedimientos de sintaxis y procedimientos de evaluación apropiados eval-and y eval-or . Alternativamente, mostrar cómo implementar y y o como expresiones derivadas.

**Ejercicio 4.5.**  Scheme permite una sintaxis adicional para las cláusulas cond , (< *test* > => < *receiver* >) . Si < *test* > se evalúa como un valor verdadero, se evalúa < *receiver* >. Su valor debe ser un procedimiento de un argumento; este procedimiento se invoca entonces sobre el valor de < *test* > y el resultado se devuelve como el valor de la expresión cond . Por ejemplo

(cond ((assoc 'b '((a 1) (b 2))) => cadr)   
      (de lo contrario falso))

devuelve 2. Modifique el manejo de cond para que admita esta sintaxis extendida.

**Ejercicio 4.6.**  Las expresiones sean expresiones derivadas, porque

(let ((< *var 1* > < *exp 1* >)  ... (< *var n* > < *exp n* >))   
  < *cuerpo* >)

es equivalente a

((lambda (< *var 1* >  ... < *var n* >)   
   < *cuerpo* >)   
 < *exp 1* >   
   
 < *exp n* >)

Implemente una transformación sintáctica let->combination que reduzca la evaluación de expresiones let a la evaluación de combinaciones del tipo que se muestra arriba, y agregue la cláusula apropiada a eval para manejar expresiones let .

**Ejercicio 4.7.**  Let\* es similar a let , excepto que las vinculaciones de las variables let se realizan secuencialmente de izquierda a derecha, y cada vinculación se realiza en un entorno en el que todas las vinculaciones anteriores son visibles. Por ejemplo

(sea\* ((x 3)   
       (y (+ x 2))   
       (z (+ x y 5)))   
  (\* x z))

devuelve 39. Explique cómo una expresión let\* puede reescribirse como un conjunto de expresiones let anidadas y escriba un procedimiento let\*->nested-lets que realice esta transformación. Si ya hemos implementado let (ejercicio  [4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.6) ) y queremos extender el evaluador para que maneje let\* , ¿es suficiente agregar una cláusula a eval cuya acción sea

(eval (let\*->lets-anidados exp) env)

¿O debemos expandir explícitamente let\* en términos de expresiones no derivadas?

**Ejercicio 4.8.**  ``Named let '' es una variante de let que tiene la forma

(dejemos < *var* > < *enlaces* > < *cuerpo* >)

Los < *bindings* > y < *body* > son iguales que en un let común , excepto que < *var* > está enlazado dentro de < *body* > a un procedimiento cuyo cuerpo es < *body* > y cuyos parámetros son las variables en los < *bindings* >. Por lo tanto, uno puede ejecutar repetidamente el < *body* > invocando el procedimiento llamado < *var* >. Por ejemplo, el procedimiento iterativo Fibonacci (sección  [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) ) puede reescribirse usando let nombrado de la siguiente manera:

(define (fib n)   
  (sea fib-iter ((a 1)   
                 (b 0)   
                 (cuenta n))   
    (si (= cuenta 0)   
        b   
        (fib-iter (+ a b) a (- cuenta 1)))))

Modificar let->combination del ejercicio  [4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.6) para que también admita let con nombre .

**Ejercicio 4.9.**  Muchos lenguajes admiten una variedad de construcciones de iteración, como do , for , while y till . En Scheme, los procesos iterativos se pueden expresar en términos de llamadas a procedimientos ordinarios, por lo que las construcciones de iteración especiales no proporcionan una ganancia esencial en potencia computacional. Por otro lado, dichas construcciones suelen ser convenientes. Diseñe algunas construcciones de iteración, proporcione ejemplos de su uso y muestre cómo implementarlas como expresiones derivadas.

**Ejercicio 4.10.**  Al utilizar la abstracción de datos, pudimos escribir un procedimiento eval que es independiente de la sintaxis particular del lenguaje que se va a evaluar. Para ilustrar esto, diseñe e implemente una nueva sintaxis para Scheme modificando los procedimientos de esta sección, sin cambiar eval o apply .

**[4.1.3 Estructuras de datos del evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.1.3)**

Además de definir la sintaxis externa de las expresiones, la implementación del evaluador también debe definir las estructuras de datos que el evaluador manipula internamente, como parte de la ejecución de un programa, como la representación de procedimientos y entornos y la representación de verdadero y falso.

**[Prueba de predicados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_541)**

Para los condicionales, aceptamos como verdadero cualquier cosa que no sea el objeto falso explícito .

(define (¿verdadero? x)   
  (no (¿eq? x falso)))  
(definir (falso? x)   
  (eq? x falso))

**[Representando procedimientos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_542)**

Para manejar primitivas, asumimos que tenemos disponibles los siguientes procedimientos:

* (apply-primitive-procedure < *proc* > < *args* >)  
  aplica el procedimiento primitivo dado a los valores de los argumentos en la lista < *args* > y devuelve el resultado de la aplicación.
* (procedimiento-primitivo? <proc> *)* prueba  
  si <proc> *es* un procedimiento primitivo.

Estos mecanismos para manejar primitivos se describen con más detalle en la sección  [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4) .

Los procedimientos compuestos se construyen a partir de parámetros, cuerpos de procedimientos y entornos utilizando el constructor make-procedure :

(define (make-procedure parámetros cuerpo env)   
  (list 'procedimiento parámetros cuerpo env))  
(define (procedimiento-compuesto? p)   
  (lista-etiquetada? p 'procedimiento))  
(define (parámetros de procedimiento p) (cadr p))  
(define (cuerpo de procedimiento p) (caddr p))  
(define (procedimiento-entorno p) (cadddr p))

**[Operaciones en entornos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_543)**

El evaluador necesita operaciones para manipular entornos. Como se explica en la sección  [3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.2) , un entorno es una secuencia de marcos, donde cada marco es una tabla de enlaces que asocian variables con sus valores correspondientes. Usamos las siguientes operaciones para manipular entornos:

* (lookup-variable-value < *var* > < *env* >)  
  devuelve el valor que está enlazado al símbolo < *var* > en el entorno < *env* >, o señala un error si la variable no está enlazada.
* (extend-environment < *variables* > < *valores* > < *base-env* >)  
  devuelve un nuevo entorno, que consiste en un nuevo marco en el que los símbolos en la lista < *variables* > están enlazados a los elementos correspondientes en la lista < *valores* >, donde el entorno circundante es el entorno < *base-env* >.
* (define-variable! < *var* > < *value* > < *env* >)  
  agrega al primer marco en el entorno < *env* > un nuevo enlace que asocia la variable < *var* > con el valor < *value* >.
* (set-variable-value! < *var* > < *value* > < *env* >)  
  cambia la vinculación de la variable < *var* > en el entorno < *env* > de modo que la variable ahora esté vinculada al valor < *value* >, o señala un error si la variable no está vinculada.

Para implementar estas operaciones, representamos un entorno como una lista de marcos. El entorno que encierra un entorno es el cdr de la lista. El entorno vacío es simplemente la lista vacía.

(define (entorno envolvente env) (cdr env))  
(define (entorno del primer cuadro) (entorno del automóvil))   
(define el entorno vacío '())

Cada marco de un entorno se representa como un par de listas: una lista de las variables enlazadas en ese marco y una lista de los valores asociados. [14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_544)

(define (valores de las variables make-frame)   
  (valores de las variables cons))  
(define (marco-variables marco) (marco del coche))  
(define (marco-valores marco) (cdr marco))  
(define (agrega-enlace-a-marco! var val marco)   
  (establece-auto! marco (cons var (auto marco)))   
  (establece-cdr! marco (cons val (cdr marco))))

Para ampliar un entorno con un nuevo marco que asocie variables con valores, creamos un marco que consta de la lista de variables y la lista de valores y lo adjuntamos al entorno. Señalamos un error si el número de variables no coincide con el número de valores.

(define (extend-environment vars vals base-env)   
  (if (= (length vars) (length vals))   
      (cons (make-frame vars vals) base-env)   
      (if (< (length vars) (length vals))   
          (error "Demasiados argumentos suministrados" vars vals)   
          (error "Muy pocos argumentos suministrados" vars vals))))

Para buscar una variable en un entorno, escaneamos la lista de variables en el primer marco. Si encontramos la variable deseada, devolvemos el elemento correspondiente en la lista de valores. Si no encontramos la variable en el marco actual, buscamos en el entorno que la contiene, y así sucesivamente. Si llegamos al entorno vacío, indicamos un error de "variable no vinculada".

(define (variable-de-búsqueda-valor var env)   
  (define (bucle-entorno env)   
    (define (escanear vars vals)   
      (cond ((vars? nulas)   
             (bucle-entorno (entorno-de-entorno)))   
            ((var? eq (vars auto))   
             (vals auto))   
            (else (escanear (vars cdr) (vals cdr)))))   
    (if (eq? env el-entorno-vacío)   
        (error var "Variable no enlazada")   
        (let ((marco (primer-marco env)))   
          (escanear (variables-marco marco)   
                (valores-marco marco)))))   
  (bucle-entorno env))

Para establecer una variable con un nuevo valor en un entorno específico, buscamos la variable, tal como en lookup-variable-value , y cambiamos el valor correspondiente cuando lo encontramos.

(define (establecer-valor-variable! var val env)   
  (define (bucle-entorno env)   
    (define (escanear vars vals)   
      (cond ((null? vars)   
             (bucle-entorno (entorno-de-enlace)))   
            ((eq? var (vars coche))   
             (establecer-coche! vals val))   
            (de lo contrario (escanear (vars cdr) (vals cdr)))))   
    (si (eq? env el-entorno-vacío)   
        (error "Variable no vinculada -- ¡ESTABLECER!" var)   
        (let ((marco (primer-marco env)))   
          (escanear (variables-marco marco)   
                (valores-marco marco)))))   
  (bucle-entorno env))

Para definir una variable, buscamos en el primer cuadro un enlace para la variable y lo modificamos si existe (como en set-variable-value! ). Si no existe tal enlace, agregamos uno al primer cuadro.

(define (define-variable! var val env)   
  (let ((frame (first-frame env)))   
    (define (scan vars vals)   
      (cond ((null? vars)   
             (add-binding-to-frame! var val frame))   
            ((eq? var (car vars))   
             (set-car! vals val))   
            (else (scan (cdr vars) (cdr vals)))))   
    (scan (frame-variables frame)   
          (frame-values ​​frame))))

El método descrito aquí es sólo una de las muchas formas plausibles de representar entornos. Dado que utilizamos la abstracción de datos para aislar el resto del evaluador de la elección detallada de la representación, podríamos cambiar la representación del entorno si quisiéramos. (Véase el ejercicio  [4.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.11) .) En un sistema Lisp de calidad de producción, la velocidad de las operaciones del entorno del evaluador (especialmente la de búsqueda de variables) tiene un gran impacto en el rendimiento del sistema. La representación descrita aquí, aunque conceptualmente sencilla, no es eficiente y no se utilizaría normalmente en un sistema de producción. [15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_545)

**Ejercicio 4.11.**   En lugar de representar un marco como un par de listas, podemos representarlo como una lista de enlaces, donde cada enlace es un par nombre-valor. Reescriba las operaciones del entorno para utilizar esta representación alternativa.

**Ejercicio 4.12.**   Los procedimientos set-variable-value!, define -variable! y lookup-variable-value se pueden expresar en términos de procedimientos más abstractos para recorrer la estructura del entorno. Defina abstracciones que capturen los patrones comunes y redefina los tres procedimientos en términos de estas abstracciones.

**Ejercicio 4.13.**   Scheme nos permite crear nuevos enlaces para variables por medio de define , pero no proporciona ninguna manera de deshacernos de los enlaces. Implemente para el evaluador una forma especial make-unbound! que elimine el enlace de un símbolo dado del entorno en el que se evalúa la expresión make-unbound! . Este problema no está completamente especificado. Por ejemplo, ¿deberíamos eliminar solo el enlace en el primer marco del entorno? Complete la especificación y justifique cualquier elección que haga.

**[4.1.4 Ejecución del evaluador como programa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.1.4)**

Dado el evaluador, tenemos en nuestras manos una descripción (expresada en Lisp) del proceso mediante el cual se evalúan las expresiones de Lisp. Una ventaja de expresar el evaluador como un programa es que podemos ejecutar el programa. Esto nos proporciona, ejecutándose dentro de Lisp, un modelo funcional de cómo Lisp evalúa las expresiones. Esto puede servir como marco para experimentar con reglas de evaluación, como haremos más adelante en este capítulo.

Nuestro programa evaluador reduce las expresiones en última instancia a la aplicación de procedimientos primitivos. Por lo tanto, todo lo que necesitamos para ejecutar el evaluador es crear un mecanismo que invoque al sistema Lisp subyacente para modelar la aplicación de procedimientos primitivos.

Debe haber un enlace para cada nombre de procedimiento primitivo, de modo que cuando eval evalúe el operador de una aplicación de un primitivo, encuentre un objeto para pasar a apply . Por lo tanto, configuramos unentorno global que asocia objetos únicos con los nombres de los procedimientos primitivos que pueden aparecer en las expresiones que vamos a evaluar. El entorno global también incluye enlaces para los símbolosverdadero y falso , para que puedan usarse como variables en expresiones a evaluar.

(define (entorno-de-configuración)   
  (let ((entorno-inicial   
         (entorno-extendido (nombres-de-procedimiento-primitivo)   
                             (objetos-de-procedimiento-primitivo)   
                             el-entorno-vacío)))   
    (define-variable! 'true true entorno-inicial)   
    (define-variable! 'false false entorno-inicial)   
    entorno-inicial))  
(definir el entorno global (configurar el entorno))

No importa cómo representemos los objetos de procedimiento primitivo, siempre que apply pueda identificarlos y aplicarlos mediante los procedimientos primitive-procedure? y apply-primitive-procedure . Hemos optado por representar un procedimiento primitivo como una lista que comienza con el símbolo primitive y que contiene un procedimiento en el Lisp subyacente que implementa ese primitivo.

(define (procedimiento-primitivo? proc)   
  (lista-etiquetada? proc 'primitivo))  
  
(define (procedimiento de implementación primitiva) (procedimiento cadr))

El entorno de configuración obtendrá los nombres primitivos y los procedimientos de implementación de una lista: [16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_549)

(define procedimientos-primitivos   
  (lista (lista 'coche coche)   
        (lista 'cdr cdr)   
        (lista 'cons cons)   
        (lista 'null? null?)   
        < *más primitivos* >   
        ))  
(define (nombres de procedimientos primitivos)   
  (map car   
       procedimientos primitivos))   
  
(define (objetos de procedimientos primitivos)  
  (mapa (lambda (proc) (lista 'primitivo (cadr proc)))   
       procedimientos-primitivos))

Para aplicar un procedimiento primitivo, simplemente aplicamos el procedimiento de implementación a los argumentos, utilizando el sistema Lisp subyacente: [17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_550)

(define (procedimiento-primitivo-aplicar argumentos)   
  (procedimiento-primitivo-aplicar-en-esquema-subyacente   
   (procedimiento-primitivo) argumentos))

Para facilitar la ejecución del evaluador metacircular, proporcionamos un *bucle de controlador* que modela el bucle de lectura-evaluación-impresión del sistema Lisp subyacente. Imprime un*prompt* , lee una expresión de entrada, evalúa esta expresión en el entorno global e imprime el resultado. Precedemos cada resultado impreso con un *mensaje de salida* para distinguir el valor de la expresión de otra salida que pueda imprimirse. [18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_551)

(define el mensaje de entrada ";;; entrada M-Eval:")   
(define el mensaje de salida ";;; valor M-Eval:")  
(define (bucle-del-controlador)   
  (solicitud-de-entrada solicitud-de-entrada)   
  (let ((entrada (leer)))   
    (let ((salida (eval entrada el-entorno-global)))   
      (salida-del-anunciador solicitud-de-salida)   
      (impresión-del-usuario salida)))   
  (bucle-del-controlador))  
(define (cadena de solicitud de entrada)   
  (nueva línea) (nueva línea) (cadena de visualización) (nueva línea))  
  
(define (cadena de salida de anuncio)   
  (nueva línea) (cadena de visualización) (nueva línea))

Utilizamos un procedimiento de impresión especial, user-print , para evitar imprimir la parte del entorno de un procedimiento compuesto, que puede ser una lista muy larga (o incluso puede contener ciclos).

(define (objeto de impresión de usuario)   
  (si (objeto de procedimiento compuesto?)   
      (mostrar (lista 'procedimiento compuesto   
                     (objeto de parámetros de procedimiento)   
                     (objeto de cuerpo de procedimiento)   
                     '<entorno de procedimiento>))   
      (objeto de visualización)))

Ahora, todo lo que tenemos que hacer para ejecutar el evaluador es inicializar el entorno global e iniciar el bucle del controlador. A continuación, se muestra una interacción de ejemplo:

(define el entorno global (entorno de configuración))   
(bucle del controlador)   
*;;; Entrada de evaluación M:*  
(define (añade x y)   
  (si (¿nulo? x)   
      y   
      (cons (auto x)   
            (añade (cdr x) y))))   
*;;; Valor de evaluación M:*   
*ok*   
*;;; Entrada de evaluación M:*  
(añade '(a b c) '(d e f))   
*;;; Valor de evaluación M:*   
*(a b c d e f)*

**Ejercicio 4.14.**   Eva Lu Ator y Louis Reasoner están experimentando con el evaluador metacircular. Eva escribe la definición de map y ejecuta algunos programas de prueba que lo utilizan. Funcionan bien. Louis, en cambio, ha instalado la versión del sistema de map como primitiva para el evaluador metacircular. Cuando lo prueba, las cosas salen terriblemente mal. Explique por qué el map de Louis falla aunque el de Eva funciona.

**[4.1.5 Datos como programas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.1.5)**

Al pensar en un programa Lisp que evalúa expresiones Lisp, puede resultar útil una analogía. Una visión operativa del significado de un programa es que unUn programa es una descripción de una máquina abstracta (quizás infinitamente grande). Por ejemplo, considere el conocido programa para calcular factoriales:

(define (factorial n)   
  (si (= n 1)   
      1   
      (\* (factorial (- n 1)) n)))

Podemos considerar este programa como la descripción de una máquina que contiene partes que decrementan, multiplican y prueban la igualdad, junto con un interruptor de dos posiciones y otra máquina factorial. (La máquina factorial es infinita porque contiene otra máquina factorial dentro de ella). La Figura  [4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.2) es un diagrama de flujo para la máquina factorial, que muestra cómo se conectan las partes entre sí.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 4.2:**   El programa factorial, visto como una máquina abstracta. |

De manera similar, podemos considerar al evaluador como una máquina muy especial que toma como entrada una descripción de una máquina. Dada esta entrada, el evaluador se configura a sí mismo para emular la máquina descrita. Por ejemplo, si le damos a nuestro evaluador la definición de factorial , como se muestra en la figura  [4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.3) , el evaluador podrá calcular factoriales.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 4.3:**   El evaluador emulando una máquina factorial. |

Desde esta perspectiva, nuestro evaluador se ve como una *máquina universal* . Imita a otras máquinas cuando se las describe como programas Lisp. [19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_553) Esto es sorprendente. Trate de imaginar un evaluador análogo para circuitos eléctricos. Este sería un circuito que toma como entrada una señal que codifica los planes para algún otro circuito, como un filtro. Dada esta entrada, el evaluador de circuitos se comportaría entonces como un filtro con la misma descripción. Un circuito eléctrico universal de este tipo es casi inimaginablemente complejo. Es notable que el evaluador de programas sea un programa bastante simple. [20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_554)

Otro aspecto sorprendente del evaluador es que actúa como un puente entre los objetos de datos que son manipulados por nuestro lenguaje de programación y el lenguaje de programación en sí. Imagine que el programa evaluador (implementado en Lisp) está ejecutándose y que un usuario está escribiendo expresiones en el evaluador y observando los resultados. Desde la perspectiva del usuario, una expresión de entrada como (\* xx) es una expresión en el lenguaje de programación, que el evaluador debe ejecutar. Sin embargo, desde la perspectiva del evaluador, la expresión es simplemente una lista (en este caso, una lista de tres símbolos: \* , x y x ) que se debe manipular de acuerdo con un conjunto de reglas bien definidas.

El hecho de que los programas del usuario sean los datos del evaluador no tiene por qué ser una fuente de confusión. De hecho, a veces es conveniente ignorar esta distinción y dar al usuario la capacidad de evaluar explícitamente un objeto de datos como una expresión Lisp, haciendo que eval esté disponible para su uso en programas. Muchos dialectos Lisp proporcionan unaprocedimiento de evaluación primitivo que toma como argumentos una expresión y un entorno y evalúa la expresión en relación con el entorno. [21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_555) Por lo tanto,

(eval '(\* 5 5) entorno inicial del usuario)

y

(eval (cons '\* (list 5 5)) entorno inicial del usuario)

Ambos regresarán el [25.22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_556)

**Ejercicio 4.15.**  Dado un procedimiento de un argumento p y un objeto a , se dice que p ``se detiene'' en a si la evaluación de la expresión (pa) devuelve un valor (en lugar de terminar con un mensaje de error o ejecutarse indefinidamente). Demuestre que es imposible escribir un procedimiento halts? que determine correctamente si p se detiene en a para cualquier procedimiento p y objeto a . Utilice el siguiente razonamiento: Si tuviera un procedimiento halts? de este tipo , podría implementar el siguiente programa:

(define (ejecutar-por-siempre) (ejecutar-por-siempre))   
  
(define (intentar p)   
  (si (¿detiene? p p)   
      (ejecutar-por-siempre)   
      'detenido))

Ahora considere evaluar la expresión (try try) y demostrar que cualquier resultado posible (ya sea detenerse o correr para siempre) viola el comportamiento previsto de las paradas. . [23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_558)

**[4.1.6 Definiciones internas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.1.6)**

Nuestro modelo de entorno de evaluación y nuestro evaluador metacircular ejecutan definiciones en secuencia, extendiendo el marco del entorno una definición a la vez. Esto es particularmente conveniente para el desarrollo de programas interactivos, en los que el programador necesita mezclar libremente la aplicación de procedimientos con la definición de nuevos procedimientos. Sin embargo, si pensamos detenidamente en las definiciones internas utilizadas para implementar la estructura de bloques (introducidas en la sección  [1.1.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.8) ), descubriremos que la extensión nombre por nombre del entorno puede no ser la mejor manera de definir variables locales.

Considere un procedimiento con definiciones internas, como

(define (f x)   
  (define (¿par? n)   
    (si (= n 0)   
        es verdadero   
        (¿impar? (- n 1))))   
  (define (¿impar? n)   
    (si (= n 0)   
        es falso   
        (¿par? (- n 1))))   
  < *resto del cuerpo de  f* >)

Nuestra intención aquí es que el nombre odd? en el cuerpo del procedimiento even? se refiera al procedimiento odd? que se define después de even?. El alcance del nombre odd? es todo el cuerpo de f , no solo la parte del cuerpo de f que comienza en el punto donde aparece la definición para odd?. De hecho, cuando consideramos que odd? se define en términos de even? (de modo que even? y odd? son procedimientos recursivos entre sí), vemos que la única interpretación satisfactoria de las dos definiciones es considerarlas como si los nombres even? y odd? se estuvieran agregando al entorno simultáneamente. De manera más general, en la estructura de bloques, el alcance de un nombre local es todo el cuerpo del procedimiento en el que se evalúa la definición .

En realidad, nuestro intérprete evaluará las llamadas a f correctamente, pero por una razón "accidental": dado que las definiciones de los procedimientos internos vienen primero, no se evaluarán las llamadas a estos procedimientos hasta que se hayan definido todos. Por lo tanto, se habrá definido odd? para cuando se ejecute even ?. De hecho, nuestro mecanismo de evaluación secuencial dará el mismo resultado que un mecanismo que implemente directamente la definición simultánea para cualquier procedimiento en el que laLas definiciones internas aparecen primero en un cuerpo y la evaluación de las expresiones de valor para las variables definidas en realidad no utiliza ninguna de las variables definidas. (Para ver un ejemplo de un procedimiento que no obedece estas restricciones, de modo que la definición secuencial no es equivalente a la definición simultánea, consulte el ejercicio  [4.19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.19) ). [24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_559)

Sin embargo, existe una forma sencilla de tratar las definiciones de modo que los nombres definidos internamente tengan un alcance verdaderamente simultáneo: basta con crear todas las variables locales que estarán en el entorno actual antes de evaluar cualquiera de las expresiones de valor. Una forma de hacerlo es mediante una transformación de sintaxis en expresiones lambda . Antes de evaluar el cuerpo de una expresión lambda ,``scan out'' y eliminar todas las definiciones internas en el cuerpo. Las variables definidas internamente se crearán con un let y luego se establecerán en sus valores mediante asignación. Por ejemplo, el procedimiento

(lambda < *vars* >   
  (define u < *e1* >)   
  (define v < *e2* >)   
  < *e3* >)

se transformaría en

(lambda < *vars* >   
  (let ((u '\*sin asignar\*)   
        (v '\*sin asignar\*))   
    (set! u < *e1* >)   
    (set! v < *e2* >)   
    < *e3* >))

donde \*unassigned\* es un símbolo especial que hace que al buscar una variable se indique un error si se intenta utilizar el valor de la variable aún no asignada.

[En el ejercicio 4.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.18) se muestra una estrategia alternativa para escanear definiciones internas  . A diferencia de la transformación mostrada anteriormente, esta aplica la restricción de que los valores de las variables definidas se pueden evaluar sin utilizar ninguno de los valores de las variables. [25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_560)

**Ejercicio 4.16.**   En este ejercicio implementamos el método que acabamos de describir para interpretar definiciones internas. Suponemos que el evaluador admite let (ver ejercicio  [4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.6) ).

a. Cambie lookup-variable-value (sección  [4.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.3) ) para señalar un error si el valor que encuentra es el símbolo \*unassigned\* .

b. Escriba un procedimiento scan-out-defines que tome un cuerpo de procedimiento y devuelva uno equivalente que no tenga definiciones internas, realizando la transformación descrita anteriormente.

c. Instale scan-out-defines en el intérprete, ya sea en make-procedure o en procedure-body (consulte la sección  [4.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.3) ). ¿Qué lugar es mejor? ¿Por qué?

**Ejercicio 4.17.**   Dibuje diagramas del entorno en efecto cuando se evalúa la expresión < *e3* > en el procedimiento del texto, comparando cómo se estructurará cuando las definiciones se interpreten secuencialmente con cómo se estructurará si las definiciones se escanean como se describe. ¿Por qué hay un marco adicional en el programa transformado? Explique por qué esta diferencia en la estructura del entorno nunca puede marcar una diferencia en el comportamiento de un programa correcto. Diseñe una forma de hacer que el intérprete implemente la regla de alcance "simultáneo" para las definiciones internas sin construir el marco adicional.

**Ejercicio 4.18.**   Considere una estrategia alternativa para escanear definiciones que traduzca el ejemplo en el texto a

(lambda < *vars* >   
  (let ((u '\*sin asignar\*)   
        (v '\*sin asignar\*))   
    (let ((a < *e1* >)   
          (b < *e2* >))   
      (set! u a)   
      (set! v b))   
    < *e3* >))

Aquí a y b representan nuevos nombres de variables, creados por el intérprete, que no aparecen en el programa del usuario. Considere el procedimiento  de resolución de la sección [3.5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.4) :

(definir (resolver f y0 dt)   
  (definir y (integral (retardo dy) y0 dt))   
  (definir dy (mapa de flujo f y))   
  y)

¿Funcionará este procedimiento si se escanean las definiciones internas como se muestra en este ejercicio? ¿Qué sucede si se escanean como se muestra en el texto? Explique.

**Ejercicio 4.19.**   Ben Bitdiddle, Alyssa P. Hacker y Eva Lu Ator están discutiendo sobre el resultado deseado de evaluar la expresión

(sea ((a 1))   
  (defina (f x)   
    (defina b (+ a x))   
    (defina a 5)   
    (+ a b))   
  (f 10))

Ben afirma que el resultado debería obtenerse utilizando la regla secuencial para define : b se define como 11, luego a se define como 5, por lo que el resultado es 16. Alyssa objeta que la recursión mutua requiere la regla de alcance simultáneo para las definiciones de procedimientos internos, y que no es razonable tratar los nombres de los procedimientos de manera diferente a otros nombres. Por lo tanto, aboga por el mecanismo implementado en el ejercicio  [4.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.16) . Esto llevaría a que a no se asignara en el momento en que se debe calcular el valor de b . Por lo tanto, en opinión de Alyssa, el procedimiento debería producir un error. Eva tiene una tercera opinión. Dice que si las definiciones de a y b realmente están destinadas a ser simultáneas, entonces el valor 5 para a debería usarse para evaluar b . Por lo tanto, en opinión de Eva a debería ser 5, b debería ser 15 y el resultado debería ser 20. ¿Cuál de estos puntos de vista (si es que hay alguno) apoya? ¿Puede idear una forma de implementar definiciones internas para que se comporten como prefiere Eva? [26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_565)

**Ejercicio 4.20.**  Debido a que las definiciones internas parecen secuenciales pero en realidad son simultáneas, algunas personas prefieren evitarlas por completo y usan la forma especial letrec en su lugar. Letrec se parece a let , por lo que no es sorprendente que las variables que vincula estén vinculadas simultáneamente y tengan el mismo alcance entre sí. El procedimiento de ejemplo f anterior se puede escribir sin definiciones internas, pero con exactamente el mismo significado, como

(define (f x)   
  (letrec ((¿par?   
            (lambda (n)   
              (si (= n 0)   
                  verdadero   
                  (¿impar? (- n 1)))))   
           (¿impar?   
            (lambda (n)   
              (si (= n 0)   
                  falso   
                  (¿par? (- n 1))))))   
    < *resto del cuerpo de  f* >))

Expresiones Letrec , que tienen la forma

(letrec ((< *var 1* > < *exp 1* >)  ... (< *var n* > < *exp n* >))   
  < *cuerpo* >)

son una variación de let en la que las expresiones < *exp k* > que proporcionan los valores iniciales para las variables < *var k* > se evalúan en un entorno que incluye todos los enlaces letrec . Esto permite la recursión en los enlaces, como la recursión mutua de even? y odd? en el ejemplo anterior, ola evaluación del factorial 10 con

(letrec ((hecho   
          (lambda (n)   
            (si (= n 1)   
                1   
                (\* n (hecho (- n 1)))))))   
  (hecho 10))

a. Implemente letrec como una expresión derivada, transformando una expresión letrec en una expresión let como se muestra en el texto anterior o en el ejercicio  [4.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.18) . Es decir, las variables letrec deben crearse con let y luego asignarse sus valores con set !.

b. Louis Reasoner está confundido por todo este alboroto sobre las definiciones internas. Según su punto de vista, si no te gusta usar define dentro de un procedimiento, puedes usar let . Ilustre lo impreciso de su razonamiento dibujando un diagrama de entorno que muestre el entorno en el que se evalúa el < *resto del cuerpo de f* > durante la evaluación de la expresión (f 5) , con f definida como en este ejercicio. Dibuje un diagrama de entorno para la misma evaluación, pero con let en lugar de letrec en la definición de f .

**Ejercicio 4.21.**  Sorprendentemente, la intuición de Louis en el ejercicio  [4.20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.20) es correcta. De hecho, es posible especificar procedimientos recursivos sin utilizar letrec (o incluso define ), aunque el método para lograrlo es mucho más sutil de lo que Louis imaginaba. La siguiente expresión calcula 10 factorial aplicando un procedimiento recursivo.procedimiento factorial: [27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_568)

((lambda (n)   
   ((lambda (hecho)   
      (hecho hecho n))   
    (lambda (ft k)   
      (si (= k 1)   
          1   
          (\* k (ft ft (- k 1)))))))   
 10)

a. Compruebe (evaluando la expresión) que esto realmente permite calcular factoriales. Piense en una expresión análoga para calcular números de Fibonacci.

b. Considere el siguiente procedimiento, que incluye definiciones internas recursivas entre sí:

(define (f x)   
  (define (¿par? n)   
    (si (= n 0)   
        es verdadero   
        (¿impar? (- n 1))))   
  (define (¿impar? n)   
    (si (= n 0)   
        es falso   
        (¿par? (- n 1))))   
  (¿par? x))

Complete las expresiones faltantes para completar una definición alternativa de f , que no utiliza ni definiciones internas ni letrec :

(define (f x)   
  ((lambda (¿par? ¿impar?)   
     (¿par? ¿par? ¿impar? x))   
   (lambda (ev? od? n)   
     (si (= n 0) verdadero (od? <??> <??> <??>)))   
   (lambda (ev? od? n)   
     (si (= n 0) falso (ev? <??> <??> <??>)))))

**[4.1.7 Separación del análisis sintáctico de la ejecución](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.1.7)**

El evaluador implementado anteriormente es simple, pero es muy ineficiente, porque el análisis sintáctico de las expresiones se intercala con su ejecución. Por lo tanto, si un programa se ejecuta muchas veces, su sintaxis se analiza muchas veces. Considere, por ejemplo, evaluar (factorial 4) utilizando la siguiente definición de factorial :

(define (factorial n)   
  (si (= n 1)   
      1   
      (\* (factorial (- n 1)) n)))

Cada vez que se llama a factorial , el evaluador debe determinar que el cuerpo es una expresión if y extraer el predicado. Solo entonces puede evaluar el predicado y despachar según su valor. Cada vez que evalúa la expresión (\* (factorial (- n 1)) n) , o las subexpresiones (factorial (- n 1)) y (- n 1) , el evaluador debe realizar el análisis de caso en eval para determinar que la expresión es una aplicación, y debe extraer su operador y operandos. Este análisis es costoso. Realizarlo repetidamente es un desperdicio.

Podemos transformar el evaluador para que sea significativamente más eficiente si organizamos las cosas de modo que el análisis sintáctico se realice solo una vez. [28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_569) Dividimos eval , que toma una expresión y un entorno, en dos partes. El procedimiento analyze toma solo la expresión. Realiza el análisis sintáctico y devuelve un nuevo procedimiento, el*procedimiento de ejecución* , que encapsula el trabajo que se debe realizar al ejecutar la expresión analizada. El procedimiento de ejecución toma un entorno como argumento y completa la evaluación. Esto ahorra trabajo porque el análisis se llamará solo una vez en una expresión, mientras que el procedimiento de ejecución puede llamarse muchas veces.

Con la separación entre análisis y ejecución, eval ahora se convierte en

(define (eval exp env)   
  ((analiza exp) env))

El resultado de llamar a analyze es el procedimiento de ejecución que se aplicará al entorno. El procedimiento analyze es el mismo análisis de caso que el realizado por el eval original de la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) , excepto que los procedimientos a los que enviamos realizan solo análisis, no evaluación completa:

(define (analiza exp)   
  (cond ((¿autoevaluación? exp)    
         (analiza-autoevaluación exp))   
        ((¿citado? exp) (analiza-citado exp))   
        ((¿variable? exp) (analiza-variable exp))   
        ((¿asignación? exp) (analiza-asignación exp))   
        ((¿definición? exp) (analiza-definición exp))   
        ((¿si? exp) (analiza-si exp))   
        ((¿lambda? exp) (analiza-lambda exp))   
        ((¿inicio? exp) (analiza-secuencia (inicio-acciones exp)))   
        ((cond? exp) (analiza (cond->si exp)))   
        ((¿aplicación? exp) (analiza-aplicación exp))   
        (else   
         (error "Tipo de expresión desconocido -- ANALIZAR" exp))))

A continuación se muestra el procedimiento de análisis sintáctico más simple, que maneja expresiones autoevaluadas. Devuelve un procedimiento de ejecución que ignora su argumento de entorno y solo devuelve la expresión:

(define (analiza-autoevaluación exp)   
  (lambda (env) exp))

Para una expresión citada, podemos ganar un poco de eficiencia extrayendo el texto de la cita solo una vez, en la fase de análisis, en lugar de en la fase de ejecución.

(define (analiza-la-expresión-citada)   
  (let ((qval (expresión-del-texto-de-la-cita)))   
    (lambda (env) qval)))

La búsqueda del valor de una variable todavía debe realizarse en la fase de ejecución, ya que esto depende del conocimiento del entorno. [29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_570)

(define (variable-analizador exp)   
  (lambda (env) (valor-variable-de-búsqueda exp env)))

El análisis de asignación también debe posponer la configuración real de la variable hasta la ejecución, cuando se haya proporcionado el entorno. Sin embargo, el hecho de que la expresión del valor de asignación se pueda analizar (de forma recursiva) durante el análisis es una importante ganancia en eficiencia, porque la expresión del valor de asignación ahora se analizará solo una vez. Lo mismo se aplica a las definiciones.

(define (analiza-asignación exp)   
  (let ((var (asignación-variable exp))   
        (vproc (analiza (asignación-valor exp))))   
    (lambda (entorno)   
      (establecer-variable-valor! var (vproc entorno) entorno)   
      'ok)))   
(define (analiza-definición exp)   
  (let ((var (definición-variable exp))   
        (vproc (analiza (definición-valor exp))))   
    (lambda (entorno)   
      (definición-variable! var (vproc entorno) entorno)   
      'ok)))

Para las expresiones if , extraemos y analizamos el predicado, el consecuente y la alternativa en el momento del análisis.

(define (analiza-si exp)   
  (let ((pproc (analiza (si-predicado exp)))   
        (cproc (analiza (si-consecuente exp)))   
        (aproc (analiza (si-alternativa exp))))   
    (lambda (entorno)   
      (si (¿verdadero? (entorno pproc))   
          (entorno cproc)   
          (entorno aproc)))))

El análisis de una expresión lambda también logra una importante ganancia en eficiencia: analizamos el cuerpo lambda solo una vez, aunque los procedimientos resultantes de la evaluación de lambda se pueden aplicar muchas veces.

(define (analiza-lambda exp)   
  (let ((vars (lambda-parameters exp))   
        (bproc (analiza-secuencia (lambda-body exp))))   
    (lambda (env) (make-procedure vars bproc env))))

El análisis de una secuencia de expresiones (como en un inicio o el cuerpo de una expresión lambda ) es más complejo. [30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_571) Se analiza cada expresión de la secuencia, lo que genera un procedimiento de ejecución. Estos procedimientos de ejecución se combinan para producir un procedimiento de ejecución que toma un entorno como argumento y llama secuencialmente a cada procedimiento de ejecución individual con el entorno como argumento.

(define (analiza-secuencia exps)   
  (define (secuencialmente proc1 proc2)   
    (lambda (env) (proc1 env) (proc2 env)))   
  (define (bucle primer-proc rest-procs)   
    (if (null? rest-procs)   
        primer-proc   
        (bucle (secuencialmente primer-proc (car rest-procs))   
              (cdr rest-procs))))   
  (let ((procs (map analyze exps)))   
    (if (null? procs)   
        (error "Secuencia vacía -- ANALIZAR"))   
    (bucle (car procs) (cdr procs))))

Para analizar una aplicación, analizamos el operador y los operandos y construimos un procedimiento de ejecución que llama al procedimiento de ejecución del operador (para obtener el procedimiento real que se aplicará) y a los procedimientos de ejecución del operando (para obtener los argumentos reales). Luego los pasamos aexecute-application , que es el análogo de apply en la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) . Execute-application se diferencia de apply en que el cuerpo del procedimiento para un procedimiento compuesto ya se ha analizado, por lo que no es necesario realizar un análisis adicional. En su lugar, simplemente llamamos al procedimiento de ejecución para el cuerpo en el entorno extendido.

(define (analiza-aplicación exp)   
  (let ((fproc (analiza (operador exp)))   
        (aprocs (map analiza (operandos exp))))   
    (lambda (entorno)   
      (ejecuta-aplicación (entorno fproc)   
                           (map (lambda (aproc) (entorno aproc))   
                                aprocs)))))  
(define (proc. ejecutar-aplicación args)   
  (cond ((proc. procedimiento-primitivo?)   
         (proc. aplicar-proc. procedimiento-primitivo args))   
        ((proc. procedimiento-compuesto?)   
         ((proc. cuerpo-proc)   
          (proc. extender-entorno (proc. parámetros-proc)   
                              args   
                              (proc. entorno-proc))))   
        (else   
         (error   
          "Tipo de procedimiento desconocido -- proc. EJECUTAR-APLICACIÓN"   
          ))))

Nuestro nuevo evaluador utiliza las mismas estructuras de datos, procedimientos de sintaxis y procedimientos de soporte en tiempo de ejecución que en las secciones  [4.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.2) ,   [4.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.3) y  [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4) .

**Ejercicio 4.22.**  Amplíe el evaluador en esta sección para admitir la forma especial let . (Véase el ejercicio  [4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.6) .)

**Ejercicio 4.23.**  Alyssa P. Hacker no entiende por qué el análisis de secuencias debe ser tan complicado. Todos los demás procedimientos de análisis son transformaciones sencillas de los procedimientos de evaluación correspondientes (o cláusulas eval ) de la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) . Ella esperaba que el análisis de secuencias se viera así:

(define (analiza-secuencia exps)   
  (define (ejecuta-secuencia procs env)   
    (cond ((null? (cdr procs)) ((car procs) env))   
          (else ((car procs) env)   
                (ejecuta-secuencia (cdr procs) env))))   
  (let ((procs (map analizar exps)))   
    (if (null? procs)   
        (error "Secuencia vacía -- ANALIZAR"))   
    (lambda (env) (ejecuta-secuencia procs env))))

Eva Lu Ator le explica a Alyssa que la versión del texto hace más del trabajo de evaluar una secuencia en el momento del análisis. El procedimiento de ejecución de secuencia de Alyssa, en lugar de tener integradas las llamadas a los procedimientos de ejecución individuales, recorre los procedimientos para llamarlos: En efecto, aunque se han analizado las expresiones individuales de la secuencia, no se ha analizado la secuencia en sí.

Compare las dos versiones de analyze-sequence . Por ejemplo, considere el caso común (típico de los cuerpos de procedimientos) donde la secuencia tiene solo una expresión. ¿Qué trabajo realizará el procedimiento de ejecución producido por el programa de Alyssa? ¿Qué sucede con el procedimiento de ejecución producido por el programa en el texto anterior? ¿Cómo se comparan las dos versiones para una secuencia con dos expresiones?

**Ejercicio 4.24.**   Diseñe y lleve a cabo algunos experimentos para comparar la velocidad del evaluador metacircular original con la versión de esta sección. Utilice sus resultados para estimar la fracción de tiempo que se emplea en el análisis en comparación con la ejecución de varios procedimientos.

[3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_510) Aun así, quedan aspectos importantes del proceso de evaluación que no son explicados por nuestro evaluador. Los más importantes son los mecanismos detallados por los cuales los procedimientos llaman a otros procedimientos y devuelven valores a quienes los llaman. Abordaremos estas cuestiones en el capítulo 5, donde analizaremos más de cerca el proceso de evaluación implementando el evaluador como una simple máquina de registros.

[4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_511) Si nos concedemos la capacidad de aplicar primitivos,¿Qué nos queda entonces por implementar en el evaluador? El trabajo del evaluador no consiste en especificar los primitivos del lenguaje, sino más bien en proporcionar el tejido conectivo (los medios de combinación y los medios de abstracción) que une una colección de primitivos para formar un lenguaje. En concreto:

* El evaluador nos permite trabajar con expresiones anidadas. Por ejemplo, aunque la simple aplicación de primitivas sería suficiente para evaluar la expresión (+ 1 6) , no es adecuada para manejar (+ 1 (\* 2 3)) . En lo que respecta al procedimiento primitivo + , sus argumentos deben ser números, y se bloquearía si le pasáramos la expresión (\* 2 3) como argumento. Una función importante del evaluador es la de coreografiar la composición del procedimiento de modo que (\* 2 3) se reduzca a 6 antes de pasarlo como argumento a + .
* El evaluador nos permite utilizar variables. Por ejemplo, el procedimiento primitivo para la suma no tiene forma de manejar expresiones como (+ x 1) . Necesitamos un evaluador que lleve un registro de las variables y obtenga sus valores antes de invocar los procedimientos primitivos.
* El evaluador nos permite definir procedimientos compuestos. Esto implica realizar un seguimiento de las definiciones de procedimientos, saber cómo utilizar estas definiciones para evaluar expresiones y proporcionar un mecanismo que permita que los procedimientos acepten argumentos.
* El evaluador proporciona los formularios especiales, que deben evaluarse de forma diferente a las llamadas a procedimientos.

[5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_518) Podríamos haber simplificado lacláusula deaplicaciónevalutilizandomap(y estipulando queoperandsdevuelve una lista) en lugar de escribir un procedimiento explícitode lista de valores. Elegimos no utilizarmapaquí para enfatizar el hecho de que laEl evaluador se puede implementar sin ningún uso de procedimientos de orden superior (y, por lo tanto, se puede escribir en un lenguaje que no tenga procedimientos de orden superior), aunque el lenguaje que soporta incluirá procedimientos de orden superior.

[6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_520) En este caso, el lenguaje que se está implementando y el lenguaje de implementación son el mismo. Contemplación del significado de¿cierto? aquí se produce una expansión de la conciencia sin el abuso de sustancias.

Esta implementación dedefineignora un problema sutil en el manejo de definiciones internas, aunque funciona correctamente en la mayoría de los casos. Veremos cuál es el problema y cómo solucionarlo en la [sección](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.6) [4.1.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_523) .

[Como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_524) dijimos cuando presentamosdefineyset!,estos valores dependen de la implementación en Scheme, es decir, el implementador puede elegir qué valor devolver.

[9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_526) Como se mencionó en la sección [2.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.1), el evaluador ve una expresión citada como una lista que comienza conquote, incluso si la expresión está escrita con comillas. Por ejemplo,el evaluador vería'a(quote a). Consulte el ejercicio [2.55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.55).

[10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_527) El valor de unaifcuando el predicado es falso y no hay alternativa no está especificado en Scheme; aquí hemos elegido que sea falso. Admitiremos el uso de las variablestrueyfalseen expresiones que se evaluarán vinculándolas en el entorno global. Consulte la sección [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4).

[11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_528) Estos selectores para una lista de expresiones (y los correspondientes para una lista de operandos) no están pensados ​​como una abstracción de datos. Se presentan como nombres mnemotécnicos para las operaciones básicas de lista con el fin de facilitar la comprensión del evaluador de control explícito en la sección [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4).

[12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_530) El valor de unacondcuando todos los predicados son falsos y no hay ningunaelseno está especificado en Scheme; aquí hemos elegido hacerlo falso.

[13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_531) Los sistemas Practical Lisp proporcionan un mecanismo que permite al usuario agregar nuevas expresiones derivadas y especificar su implementación como transformaciones sintácticas sin modificar el evaluador. Una transformación definida por el usuario de este tipo se denomina*macro* . Aunque es fácil añadir un mecanismo elemental para definir macros, el lenguaje resultante tiene problemas sutiles de conflicto de nombres. Se han realizado muchas investigaciones sobre mecanismos para la definición de macros que no causan estas dificultades. Véase,por ejemplo, Kohlbecker 1986, Clinger y Rees 1991 y Hanson 1991.

[14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_544) Los marcos no son realmente una abstracción de datos en el código siguiente:¡Set-variable-value!y¡define-variable!usanset-car!para modificar directamente los valores en un marco. El propósito de los procedimientos de marco es hacer que los procedimientos de manipulación del entorno sean fáciles de leer.

[El inconveniente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_545) de esta representación (así como de la variante del ejercicio [4.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.11)) es que el evaluador puede tener que buscar en muchos marcos para encontrar el enlace para una variable dada.(Este enfoque se conoce como *enlace profundo* ). Una forma de evitar esta ineficiencia es utilizar una estrategia llamada *direccionamiento léxico* , que se analizará en la sección  [5.5.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.6) .

[16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_549) Cualquier procedimiento definido en el Lisp subyacente puede utilizarse como primitivo para el evaluador metacircular. El nombre de un primitivo instalado en el evaluador no tiene por qué ser el mismo que el nombre de su implementación en el Lisp subyacente; los nombres son los mismos aquí porque el evaluador metacircular implementa el propio Scheme. Así, por ejemplo, podríamos poner(list 'first car)o(list 'square (lambda (x) (\* xx)))en la lista deprocedimientos-primitivos.

[17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_550) Apply-in-underlying-schemees elapplyque hemos usado en capítulos anteriores. Elapply(sección [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1)) modela el funcionamiento de este primitivo. Tener dos cosas diferentes llamadasapplyapplydel evaluador metacircularenmascarará la definición del primitivo. Una forma de evitar esto es cambiar el nombre delprocedimiento applypara evitar conflictos con el nombre del procedimiento primitivo. En cambio, hemos asumido que hemos guardado una referencia alapplyhaciendo

(definir aplicar-en-el-esquema-subyacente aplicar)

antes de definir la metacircular apply . Esto nos permite acceder a la versión original de apply con un nombre diferente.

[18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_551) El procedimiento primitivoread espera la entrada del usuario y devuelve la siguiente expresión completa que se escribe. Por ejemplo, si el usuario escribe (+ 23 x) , read devuelve una lista de tres elementos que contiene el símbolo + , el número 23 y el símbolo x .Si el usuario escribe 'x' , read devuelve una lista de dos elementos que contiene el símbolo quote y el símbolo x .

[19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_553) El hecho de que las máquinas se describan en Lisp no es esencial. Si le damos a nuestro evaluador un programa Lisp que se comporta como un evaluador para algún otro lenguaje, digamos C, el evaluador Lisp emulará al evaluador C, que a su vez puede emular cualquier máquina descrita como un programa C. De manera similar, escribir un evaluador Lisp en C produce un programa C que puede ejecutar cualquier programa Lisp. La idea profunda aquí es que cualquier evaluador puede emular a cualquier otro. Por lo tanto, la noción de "lo que en principio se puede calcular" (ignorando los aspectos prácticos de tiempo y memoria requeridos) es independiente del lenguaje o la computadora, y en cambio refleja una noción subyacente de*computabilidad* . Esto fue demostrado por primera vez de manera clara porAlan M. Turing (1912-1954), cuyo artículo de 1936 sentó las bases de la teoríaEn el artículo, Turing presentó un modelo computacional simple, ahora conocido como*Máquina de Turing* - y argumentó que cualquier "proceso efectivo" puede formularse como un programa para dicha máquina. (Este argumento se conoce como el*Tesis de Church-Turing* .) Turing implementó entonces una máquina universal, es decir, una máquina de Turing que se comporta como un evaluador de programas de máquinas de Turing. Utilizó este marco para demostrar que hay problemas bien planteados que no pueden ser calculados por máquinas de Turing (véase el ejercicio  [4.15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.15) ), y por lo tanto, por implicación, no pueden formularse como "procesos efectivos". Turing también realizó contribuciones fundamentales a la ciencia informática práctica. Por ejemplo, inventó la idea deestructurar programas utilizando subrutinas de propósito general. VerHodges 1983 para una biografía de Turing.

[20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_554) A algunas personas les parece contradictorio que un evaluador, que se implementa mediante un procedimiento relativamente simple, pueda emular programas que son más complejos que el propio evaluador. La existencia de una máquina evaluadora universal es una propiedad profunda y maravillosa de la computación.*La teoría de la recursión* , una rama de la lógica matemática, se ocupa de los límites lógicos del cálculo.El hermoso libro de Douglas Hofstadter *Gödel, Escher, Bach* (1979) explora algunas de estas ideas.

[21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_555) Advertencia:Esta primitiva eval no es idéntica al procedimiento eval que implementamos en la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) , porque utiliza entornos de Scheme *reales* en lugar de las estructuras de entorno de muestra que creamos en la sección  [4.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.3) . El usuario no puede manipular estos entornos reales como listas comunes; se debe acceder a ellos mediante eval u otras operaciones especiales.De manera similar, el primitivo de aplicación que vimos anteriormente no es idéntico al metacircular apply , porque utiliza procedimientos de Scheme reales en lugar de los objetos de procedimiento que construimos en las secciones  [4.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.3) y  [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4) .

[22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_556) El MITLa implementación de Scheme incluye eval , así como un símbolo user-initial-environment que está vinculado al entorno inicial en el que se evalúan las expresiones de entrada del usuario.

[23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_558) Aunque estipulamos que ahalts?se le da un objeto de procedimiento, observe que este razonamiento todavía se aplica incluso sihalts?puede obtener acceso al texto del procedimiento y su entorno.*Este es el célebre Teorema de Halting* de Turing , que dio el primer ejemplo claro de un problema *no computable* , es decir, una tarea bien planteada que no puede llevarse a cabo como un procedimiento computacional.

[24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_559) El deseo de que los programas no dependan de este mecanismo de evaluación es la razón de la observación de que "la administración no es responsable" en la nota al pie [28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#footnote_Temp_45)del capítulo 1. Al insistir en que las definiciones internas vienen primero y no se utilizan entre sí mientras se evalúan las definiciones, el estándar IEEE para Scheme deja a los implementadores cierta elección en el mecanismo utilizado para evaluar estas definiciones. La elección de una regla de evaluación en lugar de otra aquí puede parecer un tema menor, que afecta sólo a la interpretación de programas "mal formados". Sin embargo, veremos en la sección [5.5.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.6)que pasar a un modelo de alcance simultáneo para las definiciones internas evita algunas dificultades desagradables que de otro modo surgirían al implementar un compilador.

[25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_560) El estándar IEEE para Scheme permite diferentes estrategias de implementación al especificar que es responsabilidad del programador obedecer esta restricción, no de la implementación hacerla cumplir. Algunas implementaciones de Scheme, incluidasEsquema MIT, utilice la transformación que se muestra arriba. Por lo tanto, algunos programas que no cumplan con esta restricción se ejecutarán en dichas implementaciones.

[Los implementadores de Scheme del MIT apoyan a Alyssa por los siguientes motivos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_565) : Eva tiene razón en principio: las definiciones deben considerarse simultáneas, pero parece difícil implementar un mecanismo general y eficiente que haga lo que Eva requiere. En ausencia de un mecanismo de este tipo, es mejor generar un error en los casos difíciles de definiciones simultáneas (la idea de Alyssa) que producir una respuesta incorrecta (como diría Ben).

[27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_568) Este ejemplo ilustra un truco de programación para formular procedimientos recursivos sin utilizardefine.El truco más general de este tipo es el *operador Y* , que se puede utilizar para dar una implementación de " cálculo puro" derecursión. (Véase Stoy 1977 para obtener detalles sobre el cálculo lambda, y Gabriel 1988 para una exposición del operador *Y* en Scheme.)

[28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_569) Esta técnica es una parte integral del proceso de compilación, que analizaremos en el capítulo 5. Jonathan Rees escribió un esquemaUn intérprete como éste lo utilizó en 1982 para el proyecto T (Rees y Adams, 1982). Marc Feeley (1986) (véase también Feeley y Lapalme, 1987) inventó esta técnica de forma independiente en su tesis de maestría.

[Sin](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_570) embargo, hay una parte importante de la búsqueda de variables que*se puede*realizar como parte del análisis sintáctico. Como mostraremos en la sección [5.5.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.6), se puede determinar la posición en la estructura del entorno donde se encontrará el valor de la variable, obviando así la necesidad de escanear el entorno en busca de la entrada que coincida con la variable.

[30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_571) Consulte el ejercicio [4.23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.23)para obtener más información sobre el procesamiento de secuencias.

[**4.2 Variaciones de un esquema: evaluación diferida**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_4.2)

Ahora que tenemos un evaluador expresado como un programa Lisp, podemos experimentar con opciones alternativas en el diseño de lenguajes simplemente modificando el evaluador. De hecho, los lenguajes nuevos a menudo se inventan escribiendo primero un evaluador que incorpora el nuevo lenguaje dentro de un lenguaje de alto nivel existente. Por ejemplo, si deseamos discutir algún aspecto de una modificación propuesta a Lisp con otro miembro de la comunidad Lisp, podemos proporcionar un evaluador que incorpore el cambio. El destinatario puede entonces experimentar con el nuevo evaluador y enviar comentarios como modificaciones adicionales. La base de implementación de alto nivel no sólo hace que sea más fácil probar y depurar el evaluador; además, la incorporación permite al diseñador extraer [31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_575) características del lenguaje subyacente, de la misma manera que nuestro evaluador Lisp incorporado utiliza primitivas y estructura de control del Lisp subyacente. Sólo más tarde (si es que alguna vez) el diseñador necesita tomarse la molestia de construir una implementación completa en un lenguaje de bajo nivel o en hardware. En esta sección y en la siguiente exploramos algunas variaciones de Scheme que proporcionan un poder expresivo adicional significativo.

**[4.2.1 Orden normal y orden de aplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.2.1)**

En la sección  [1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1) , donde comenzamos nuestra discusión de los modelos de evaluación, notamos que Scheme es un lenguaje *de orden aplicativo , es decir, que todos los argumentos de los procedimientos de Scheme se evalúan cuando se aplica el procedimiento. Por el contrario, los lenguajes de orden normal* retrasan la evaluación de los argumentos del procedimiento hasta que se necesitan los valores de los argumentos reales. Retrasar la evaluación de los argumentos del procedimiento hasta el último momento posible (por ejemplo, hasta que los requiere una operación primitiva) se denomina*evaluación perezosa* . [32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_576) Considere el procedimiento

(define (prueba a b)   
  (si (= a 0) 1 b))

La evaluación de (try 0 (/ 1 0)) genera un error en Scheme. Con una evaluación diferida, no habría ningún error. La evaluación de la expresión devolvería 1, porque el argumento (/ 1 0) nunca se evaluaría.

Un ejemplo que explota la evaluación perezosa es la definición de un procedimiento a menos que

(define (a menos que condición valor-habitual valor-excepcional)   
  (si condición valor-excepcional valor-habitual))

que se puede utilizar en expresiones como

(a menos que (= b 0)   
        (/ a b)   
        (begin (display "excepción: devolviendo 0")   
               0))

Esto no funcionará en un lenguaje de orden aplicativo porque tanto el valor habitual como el valor excepcional se evaluarán antes de llamar a a menos que se llame (compare el ejercicio  [1.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.6) ). Una ventaja de la evaluación diferida es que algunos procedimientos, como a menos que , pueden realizar cálculos útiles incluso si la evaluación de algunos de sus argumentos produjera errores o no terminara.

Si se ingresa el cuerpo de un procedimiento antes de que se haya evaluado un argumento, decimos que el procedimiento es*No es estricto* en ese argumento. Si el argumento se evalúa antes de que se ingrese el cuerpo del procedimiento, decimos que el procedimiento es*estricto* en ese argumento. [33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_577) En un lenguaje de orden puramente aplicativo, todos los procedimientos son estrictos en cada argumento. En un lenguaje de orden puramente normal, todos los procedimientos compuestos son no estrictos en cada argumento, y los procedimientos primitivos pueden ser estrictos o no estrictos. También hay lenguajes (ver ejercicio  [4.31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.31) ) que dan a los programadores un control detallado sobre la estrictez de los procedimientos que definen.

Un ejemplo sorprendente de un procedimiento que puede convertirse en no estricto es cons (o, en general, casi cualquier constructor de estructuras de datos). Se pueden realizar cálculos útiles combinando elementos para formar estructuras de datos y operando sobre las estructuras de datos resultantes, incluso si no se conocen los valores de los elementos. Tiene mucho sentido, por ejemplo, calcular la longitud de una lista sin conocer los valores de los elementos individuales de la lista. Explotaremos esta idea en la sección  [4.2.3 para implementar los flujos del capítulo 3 como listas formadas por pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2.3) de cons no estrictos .

**Ejercicio 4.25.**   Supongamos que (en el esquema de orden aplicativo ordinario) definimos a menos que como se muestra arriba y luego definimos factorial en términos de a menos que como

(define (factorial n)   
  (a menos que (= n 1)   
          (\* n (factorial (- n 1)))   
          1))

¿Qué sucede si intentamos evaluar (factorial 5) ? ¿Funcionarán nuestras definiciones en un lenguaje de orden normal?

**Ejercicio 4.26.**  Ben Bitdiddle y Alyssa P. Hacker no están de acuerdo sobre la importancia de la evaluación diferida para implementar cosas como, por ejemplo, a menos que . Ben señala que es posible implementar a menos que en orden aplicativo como una forma especial. Alyssa contraataca diciendo que, si se hiciera eso, a menos que sería meramente sintaxis, no un procedimiento que se pudiera usar junto con procedimientos de orden superior. Complete los detalles en ambos lados del argumento. Muestre cómo implementar a menos que como una expresión derivada (como cond o let ), y dé un ejemplo de una situación en la que podría ser útil tener a menos que esté disponible como un procedimiento, en lugar de como una forma especial.

**[4.2.2 Un intérprete con evaluación perezosa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.2.2)**

En esta sección implementaremos un lenguaje de orden normal que es igual a Scheme, excepto que los procedimientos compuestos no son estrictos en cada argumento. Los procedimientos primitivos seguirán siendo estrictos. No es difícil modificar el evaluador de la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) para que el lenguaje que interpreta se comporte de esta manera. Casi todos los cambios necesarios se centran en la aplicación de procedimientos.

La idea básica es que, al aplicar un procedimiento, el intérprete debe determinar qué argumentos se van a evaluar y cuáles se van a retrasar. Los argumentos retrasados ​​no se evalúan, sino que se transforman en objetos llamados*thunk* s. [34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_580) El thunk debe contener la información necesaria para producir el valor del argumento cuando se lo necesita, como si se hubiera evaluado en el momento de la aplicación. Por lo tanto, el thunk debe contener la expresión del argumento y el entorno en el que se está evaluando la aplicación del procedimiento.

El proceso de evaluar la expresión en un thunk se llama *forzado* . [35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_581) En general, un thunk se forzará solo cuando se necesite su valor: cuando se pasa a un procedimiento primitivo que usará el valor del thunk; cuando es el valor de un predicado de un condicional; y cuando es el valor de un operador que está a punto de aplicarse como un procedimiento. Una opción de diseño que tenemos disponible es si se debe o no*Memorizar* los thunks, como hicimos con los objetos retrasados ​​en la sección  [3.5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.1) . Con la memorización, la primera vez que se fuerza un thunk, almacena el valor que se calcula. Los forzamientos posteriores simplemente devuelven el valor almacenado sin repetir el cálculo. Haremos que nuestro intérprete memorice, porque esto es más eficiente para muchas aplicaciones. Sin embargo, aquí hay consideraciones complicadas. [36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_582)

**[Modificando el evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_583)**

La principal diferencia entre el evaluador perezoso y el de la sección  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) está en el manejo de las aplicaciones de procedimientos en eval y apply .

La cláusula de aplicación de eval se convierte en

((aplicación? exp)   
 (aplicar (valor real (operador exp) env)   
        (operandos exp)   
        env))

Esto es casi lo mismo que la cláusula application? de eval en la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) . Sin embargo, para la evaluación diferida, llamamos a apply con las expresiones de operandos, en lugar de los argumentos producidos al evaluarlas. Dado que necesitaremos que el entorno construya procesadores si los argumentos se van a retrasar, también debemos pasar esto. Aún evaluamos el operador, porque apply necesita el procedimiento real que se aplicará para poder despachar en su tipo (primitivo versus compuesto) y aplicarlo.

Siempre que necesitamos el valor real de una expresión, usamos

(define (valor-actual exp env)   
  (force-it (eval exp env)))

en lugar de solo eval , de modo que si el valor de la expresión es un thunk, se forzará.

Nuestra nueva versión de apply es casi la misma que la versión de la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) . La diferencia es que eval ha pasado expresiones de operandos no evaluados: para los procedimientos primitivos (que son estrictos), evaluamos todos los argumentos antes de aplicar el primitivo; para los procedimientos compuestos (que no son estrictos) demoramos todos los argumentos antes de aplicar el procedimiento.

(define (apply procedimiento argumentos env)   
  (cond ((procedimiento-primitivo? procedimiento)   
         (apply-procedimiento-primitivo   
          procedimiento   
          (lista-de-valores-arg argumentos env)))   *; modified*  
        ((procedimiento-compuesto? procedimiento)   
         (eval-secuencia   
          (cuerpo-procedimiento procedimiento)   
          (extend-entorno   
           (parámetros-procedimiento procedimiento)   
           (lista-de-argumentos-retrasados ​​argumentos env)  *; modified*  
           (entorno-procedimiento procedimiento))))   
        (else   
         (error   
          "Tipo de procedimiento desconocido -- APPLY" procedimiento))))

Los procedimientos que procesan los argumentos son iguales a list-of-values ​​de la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) , excepto que list-of-delayed-args retrasa los argumentos en lugar de evaluarlos, y list-of-arg-values ​​usa actual-value en lugar de eval :

(define (lista-de-valores-argumentos exps env)   
  (if (¿sin-operandos? exps)   
      '()   
      (cons (valor-actual (primer-operando exps) env)   
            (lista-de-valores-argumentos (resto-operandos exps)   
                                env))))  
(define (lista-de-argumentos-retrasados ​​exps env)   
  (if (¿sin-operandos? exps)   
      '()   
      (cons (delay-it (primer-operando exps) env)   
            (lista-de-argumentos-retrasados ​​(resto-operandos exps)   
                                  env))))

El otro lugar donde debemos cambiar el evaluador es en el manejo de if , donde debemos usar actual-value en lugar de eval para obtener el valor de la expresión del predicado antes de probar si es verdadero o falso:

(define (eval-if exp env)   
  (if (true? (valor-actual (if-predicado exp) env))   
      (eval (if-consecuente exp) env)   
      (eval (if-alternativa exp) env)))

Por último, debemos cambiar el procedimiento  del bucle del controlador (sección [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4) ) para utilizar el valor actual en lugar de eval , de modo que si un valor retrasado se propaga de nuevo al bucle de lectura-eval-impresión, se impondrá antes de imprimirse. También cambiamos los mensajes para indicar que este es el evaluador diferido:

(define el mensaje de entrada ";;; entrada L-Eval:")   
(define el mensaje de salida ";;; valor L-Eval:")  
(define (bucle-del-controlador)   
  (solicitud-de-entrada solicitud-de-entrada)   
  (let ((entrada (leer)))   
    (let ((salida   
           (valor-real entrada el-entorno-global)))   
      (salida-del-anunciador solicitud-de-salida)   
      (impresión-del-usuario salida)))   
  (bucle-del-controlador))

Una vez realizados estos cambios, podemos iniciar el evaluador y probarlo. La evaluación exitosa de la expresión  try analizada en la sección [4.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2.1) indica que el intérprete está realizando una evaluación diferida:

(define el entorno global (entorno de configuración))   
(bucle del controlador)   
*;;; Entrada de L-Eval:*  
(define (try a b)   
  (if (= a 0) 1 b))   
*;;; Valor de L-Eval:*   
*ok*   
*;;; Entrada de L-Eval:*  
(try 0 (/ 1 0))   
*;;; Valor de L-Eval:*   
*1*

**[Representando trozos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_584)**

Nuestro evaluador debe organizar la creación de thunks cuando se aplican procedimientos a los argumentos y forzar estos thunks más tarde. Un thunk debe empaquetar una expresión junto con el entorno, de modo que el argumento pueda producirse más tarde. Para forzar el thunk, simplemente extraemos la expresión y el entorno del thunk y evaluamos la expresión en el entorno. Usamos actual-value en lugar de eval de modo que en caso de que el valor de la expresión sea en sí mismo un thunk, lo forzaremos, y así sucesivamente, hasta que alcancemos algo que no sea un thunk:

(define (force-it obj)   
  (si (thunk? obj)   
      (valor-actual (thunk-exp obj) (thunk-env obj))   
      obj))

Una forma sencilla de empaquetar una expresión con un entorno es crear una lista que contenga la expresión y el entorno. De este modo, creamos un thunk de la siguiente manera:

(define (exp env de retraso)   
  (lista 'thunk exp env))   
  
(define (thunk? obj)   
  (lista-etiquetada? obj 'thunk))   
  
(define (exp-thunk thunk) (cadr thunk))   
  
(define (env-thunk thunk) (caddr thunk))

En realidad, lo que queremos para nuestro intérprete no es exactamente esto, sino fragmentos que hayan sido memorizados. Cuando se fuerza un fragmento, lo convertiremos en un fragmento evaluado reemplazando la expresión almacenada con su valor y cambiando la etiqueta del fragmento para que pueda reconocerse como ya evaluado. [37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_585)

(define (thunk-evaluado? obj)   
  (lista-etiquetada? obj 'thunk-evaluado))   
  
(define (valor-thunk thunk-evaluado) (cadr thunk-evaluado))  
(define (force-it obj)   
  (cond ((thunk? obj)   
         (let ((result (valor-actual   
                        (thunk-exp obj)   
                        (thunk-env obj))))   
           (set-car! obj 'evaluated-thunk)   
           (set-car! (cdr obj) result)   *; reemplaza  exp  con su valor*  
           (set-cdr! (cdr obj) '())      ; olvida el resultado *env innecesario*  
           ))   
        ((evaluated-thunk? obj)   
         (thunk-value obj))   
        (else obj)))

Tenga en cuenta que el mismo procedimiento de retraso funciona tanto con memorización como sin ella.

**Ejercicio 4.27.**   Supongamos que escribimos las siguientes definiciones en el evaluador perezoso:

(define el conteo 0)   
(define (id x)   
  (establece! conteo (+ conteo 1))   
  x)

Proporcione los valores que faltan en la siguiente secuencia de interacciones y explique sus respuestas. [38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_587)

(define w (id (id 10)))   
*;;; Entrada L-Eval:*  
conteo   
*;;; Valor L-Eval:*  
< *respuesta* >   
*;;; Entrada L-Eval:*  
w   
*;;; Valor L-Eval:*  
< *respuesta* >   
*;;; Entrada L-Eval:*  
conteo   
*;;; Valor L-Eval:*  
< *respuesta* >

**Ejercicio 4.28.**   Eval utiliza actual-value en lugar de eval para evaluar el operador antes de pasárselo a apply , con el fin de forzar el valor del operador. Dé un ejemplo que demuestre la necesidad de este forzamiento.

**Ejercicio 4.29.**   Muestre un programa que esperaría que se ejecute mucho más lentamente sin memorización que con memorización. Además, considere la siguiente interacción, donde el procedimiento id se define como en el ejercicio  [4.27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.27) y count comienza en 0:

(define (cuadrado x)   
  (\* x x))   
*;;; Entrada de L-Eval:*  
(cuadrado (id 10))   
*;;; Valor de L-Eval:*  
< *respuesta* >   
*;;; Entrada de L-Eval:*  
conteo   
*;;; Valor de L-Eval:*  
< *respuesta* >

Dar las respuestas tanto cuando el evaluador las memoriza como cuando no lo hace.

**Ejercicio 4.30.**   Cy D. Fect, un programador de C reformado, está preocupado de que algunos efectos secundarios nunca se produzcan, porque el evaluador perezoso no fuerza las expresiones en una secuencia. Dado que el valor de una expresión en una secuencia distinta de la última no se utiliza (la expresión está allí sólo por su efecto, como asignar a una variable o imprimir), no puede haber ningún uso posterior de este valor (por ejemplo, como un argumento para un procedimiento primitivo) que haga que se fuerce. Por lo tanto, Cy piensa que al evaluar secuencias, debemos forzar todas las expresiones en la secuencia excepto la última. Propone modificar eval-sequence de la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) para utilizar actual-value en lugar de eval :

(define (eval-secuencia exps env)   
  (cond ((ultima exp? exps) (eval (primera exp exps) env))   
        (else (valor-actual (primera exp exps) env)   
              (eval-secuencia (resto-exps exps) env))))

a. Ben Bitdiddle piensa que Cy está equivocado. Le muestra el procedimiento for-each descrito en el ejercicio  [2.23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.23) , que ofrece un ejemplo importante de una secuencia con efectos secundarios:

(define (para cada procedimiento elementos)   
  (si (¿nulo? elementos)   
      'hecho   
      (comienza (proc (car elementos))   
             (para cada procedimiento (cdr elementos)))))

Afirma que el evaluador en el texto (con la secuencia eval-sequence original ) maneja esto correctamente:

*;;; Entrada de L-Eval:*  
(for-each (lambda (x) (nueva línea) (mostrar x))   
          (lista 57 321 88))   
*57*   
*321*   
*88*   
*;;; Valor de L-Eval:*   
*hecho*

Explique por qué Ben tiene razón sobre el comportamiento de for-each .

b. Cy está de acuerdo en que Ben tiene razón sobre el ejemplo de for-each , pero dice que ese no es el tipo de programa en el que estaba pensando cuando propuso su cambio a eval-sequence . Define los dos procedimientos siguientes en el evaluador perezoso:

(define (p1 x)   
  (establece! x (cons x '(2)))   
  x)   
  
(define (p2 x)   
  (define (p e)   
    e   
    x)   
  (p (establece! x (cons x '(2)))))

¿Cuáles son los valores de (p1 1) y (p2 1) con la secuencia de evaluación original ? ¿Cuáles serían los valores con el cambio propuesto por Cy a la secuencia de evaluación ?

c. Cy también señala que cambiar la secuencia de evaluación como él propone no afecta el comportamiento del ejemplo de la parte a. Explique por qué esto es cierto.

d. ¿Cómo crees que se deben tratar las secuencias en el evaluador perezoso? ¿Te gusta el enfoque de Cy, el enfoque del texto o algún otro enfoque?

**Ejercicio 4.31.**  El enfoque adoptado en esta sección es algo desagradable, porque realiza un cambio incompatible con Scheme. Podría ser mejor implementar la evaluación diferida como una *extensión compatible con versiones anteriores* , es decir, de modo que los programas Scheme ordinarios funcionen como antes. Podemos hacer esto extendiendo la sintaxis de las declaraciones de procedimientos para permitir que el usuario controle si los argumentos se retrasarán o no. Mientras estamos en ello, también podemos darle al usuario la opción de retrasar con o sin memorización. Por ejemplo, la definición

(define (f a (b perezoso) c (d perezoso-memo))   
  ... )

definiría f como un procedimiento de cuatro argumentos, donde el primer y tercer argumento se evalúan cuando se llama al procedimiento, el segundo argumento se retrasa y el cuarto argumento se retrasa y se memoriza. Por lo tanto, las definiciones de procedimientos ordinarios producirán el mismo comportamiento que Scheme ordinario, mientras que agregar la declaración lazy-memo a cada parámetro de cada procedimiento compuesto producirá el comportamiento del evaluador perezoso definido en esta sección. Diseñe e implemente los cambios necesarios para producir dicha extensión de Scheme. Tendrá que implementar nuevos procedimientos de sintaxis para manejar la nueva sintaxis para define . También debe hacer que eval o apply determinen cuándo se deben retrasar los argumentos y forzar o retrasar los argumentos en consecuencia, y debe hacer que se fuerce o no se memoice, según corresponda.

**[4.2.3 Flujos como listas diferidas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.2.3)**

En la sección  [3.5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.1) , mostramos cómo implementar streams como listas retrasadas. Introdujimos las formas especiales delay y cons-stream , que nos permitieron construir una ``promesa'' para calcular el cdr de un stream, sin cumplir realmente esa promesa hasta más tarde. Podríamos usar esta técnica general de introducir formas especiales siempre que necesitemos más control sobre el proceso de evaluación, pero esto es incómodo. Por un lado, una forma especial no es un objeto de primera clase como un procedimiento, por lo que no podemos usarlo junto con procedimientos de orden superior. [39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_592) Además, nos vimos obligados a crear streams como un nuevo tipo de objeto de datos similar pero no idéntico a las listas, y esto nos obligó a reimplementar muchas operaciones de lista ordinarias ( map , append , etc.) para usarlas con streams.

Con la evaluación diferida, los flujos y las listas pueden ser idénticos, por lo que no hay necesidad de formas especiales ni de operaciones separadas para listas y flujos. Todo lo que necesitamos hacer es organizar las cosas de modo que cons no sea estricto. Una forma de lograr esto es extender el evaluador diferido para permitir primitivos no estrictos e implementar cons como uno de ellos. Una forma más fácil es recordar (sección  [2.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.1.3) ) que no hay ninguna necesidad fundamental de implementar cons como primitivo. En cambio, podemos representarpares como procedimientos: [40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_593)

(define (cons x y)   
  (lambda (m) (m x y)))  
(define (coche z)   
  (z (lambda (p q) p)))  
(define (cdr z)   
  (z (lambda (p q) q)))

En términos de estas operaciones básicas, las definiciones estándar de las operaciones de lista funcionarán con listas infinitas (flujos) así como con listas finitas, y las operaciones de flujo pueden implementarse como operaciones de lista. A continuación se muestran algunos ejemplos:

(define (lista-ref elementos n)   
  (si (= n 0)   
      (auto elementos)   
      (lista-ref (cdr elementos) (- n 1))))  
(define (map proc items)   
  (if (null? items)   
      '()   
      (cons (proc (car items))   
            (map proc (cdr items)))))  
(define (factor de elementos de la lista de escala)   
  (mapa (lambda (x) (\* factor x))   
       elementos))  
(define (add-lists lista1 lista2)   
  (cond ((null? lista1) lista2)   
        ((null? lista2) lista1)   
        (else (cons (+ (car lista1) (car lista2))   
                    (add-lists (cdr lista1) (cdr lista2))))))  
(define unos (cons 1 unos))  
(define enteros (cons 1 (add-lists ones wholes)))   
*;;; Entrada L-Eval:*  
(list-ref wholes 17)   
*;;; Valor L-Eval:*   
*18*

Tenga en cuenta que estas listas perezosas son incluso más perezosas que los flujos del capítulo 3: el car de la lista, así como el cdr , se retrasan. [41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_594) De hecho, incluso acceder al car o al cdr de un par perezoso no necesita forzar el valor de un elemento de la lista. El valor se forzará solo cuando sea realmente necesario, por ejemplo, para su uso como argumento de un primitivo o para imprimirse como respuesta.

Los pares perezosos también ayudan con el problema que surgió con los flujos en la sección  [3.5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.4) , donde encontramos que formular modelos de flujo de sistemas con bucles puede requerir que rociemos nuestros programas conOperaciones de retardo explícitas , más allá de las proporcionadas por cons-stream . Con la evaluación diferida, todos los argumentos de los procedimientos se retrasan de manera uniforme. Por ejemplo, podemos implementar procedimientos para integrar listas y resolver ecuaciones diferenciales como lo habíamos previsto originalmente en la sección  [3.5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.4) :

(define (integrando integral valor inicial dt)   
  (define int   
    (cons valor inicial   
          (listas-de-suma (lista-de-escala integrando dt)   
                    int)))   
  int)  
(define (solve f y0 dt)   
  (define y (integral dy y0 dt))   
  (define dy (map f y))   
  y)   
*;;; Entrada de L-Eval:*  
(list-ref (solve (lambda (x) x) 1 0.001) 1000)   
*;;; Valor de L-Eval:*   
*2.716924*

**Ejercicio 4.32.**   Dé algunos ejemplos que ilustren la diferencia entre los flujos del capítulo 3 y las listas perezosas «más perezosas» descritas en esta sección. ¿Cómo puede aprovechar esta pereza adicional?

**Ejercicio 4.33.**   Ben Bitdiddle prueba la implementación de la lista perezosa dada anteriormente evaluando la expresión

(coche '(a b c))

Para su sorpresa, esto produce un error. Después de pensarlo un poco, se da cuenta de que las "listas" obtenidas al leer expresiones entre comillas son diferentes de las listas manipuladas por las nuevas definiciones de cons , car y cdr . Modifique el tratamiento del evaluador de las expresiones entre comillas de modo que las listas entre comillas escritas en el bucle del controlador produzcan listas perezosas verdaderas.

**Ejercicio 4.34.**   Modifique el bucle controlador del evaluador de modo que los pares perezosos y las listas se impriman de alguna manera razonable. (¿Qué va a hacer con las listas infinitas?) También puede que necesite modificar la representación de los pares perezosos de modo que el evaluador pueda identificarlos para imprimirlos.

[31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_575) Snarf: ``Agarrar, especialmente un documento grande oarchivo con el propósito de usarlo con o sin el permiso del propietario.'' Snarf down: ``Para snarf, a veces con la connotación de absorber, procesar o comprender.'' (Estas definiciones fueron snarfed de Steele et al. 1983. Véase también Raymond 1993.)

[32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_576) La diferencia entre la terminología «perezosa» y la terminología «de orden normal» es un tanto difusa. En general, «perezosa» se refiere a los mecanismos de evaluadores particulares, mientras que «de orden normal» se refiere a la semántica de los lenguajes, independientemente de cualquier estrategia de evaluación particular. Pero no se trata de una distinción estricta y las dos terminologías suelen usarse indistintamente.

[33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_577) La terminología "estricto" versus "no estricto" significa esencialmente lo mismo que "orden aplicativo" versus "orden normal", excepto que se refiere a procedimientos y argumentos individuales en lugar de al lenguaje en su conjunto. En una conferencia sobre lenguajes de programación, es posible que escuche a alguien decir: "El lenguaje de orden normal"El problema tiene ciertas primitivas estrictas. Otros procedimientos toman sus argumentos mediante una evaluación perezosa".

[34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_580) La palabra*thunk*fue inventada por un informalgrupo de trabajo que estaba discutiendo la implementación de llamada por nombre en Algol 60. Observaron que la mayor parte del análisis (``pensamiento sobre'') de la expresión se podía hacer en tiempo de compilación; por lo tanto, en tiempo de ejecución, la expresión ya habría sido ``pensada'' (Ingerman et al. 1960).

[35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_581) Esto es análogo al uso dela fuerza sobre los objetos retrasados ​​que se introdujeron en el capítulo 3 para representar secuencias. La diferencia fundamental entre lo que estamos haciendo aquí y lo que hicimos en el capítulo 3 es que estamos incorporando el retraso y la forzada en el evaluador, y por lo tanto haciendo que esto sea uniforme y automático en todo el lenguaje.

[36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_582) La evaluación perezosa combinada con la memorización es a vecesse denomina paso de argumentos *por necesidad* , en contraste al paso de argumentos *por nombre* .(La llamada por nombre, introducida en Algol 60, es similar a la evaluación diferida no memorizada). Como diseñadores de lenguajes, podemos construir nuestro evaluador para que memorice, para que no memorice, o dejar esta opción a los programadores (ejercicio  [4.31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.31) ). Como se podría esperar del capítulo 3, estas opciones plantean problemas que se vuelven sutiles y confusos en presencia de asignaciones. (Véase los ejercicios  [4.27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.27) y  [4.29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.29) ).Un excelente artículo de Clinger (1982) intenta aclarar las múltiples dimensiones de confusión que surgen aquí.

[37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_585) Observe que también borramos elenvdel thunk una vez que se ha calculado el valor de la expresión. Esto no hace ninguna diferencia en los valores devueltos por el intérprete. Sin embargo, ayuda a ahorrar espacio, porque eliminar la referencia del thunk alenvuna vez que ya no se necesita permite que esta estructura sea*basura recolectada* y su espacio reciclado, como discutiremos en la sección  [5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.3) .

De manera similar, podríamos haber permitido que los entornos innecesarios en los objetos retrasados ​​memorizados de la sección  [3.5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.1) fueran recolectados como basura, haciendo que memo-proc hiciera algo como (set! proc '()) para descartar el procedimiento proc (que incluye el entorno en el que se evaluó el retraso ) después de almacenar su valor.

[38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_587) Este ejercicio demuestra que la interacción entre la evaluación perezosa y los efectos secundarios puede ser muy confusa. Esto es justo lo que se podría esperar de la discusión del capítulo 3.

[Éste](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_592) es precisamente el problema con ela menos que, como en el ejercicio [4.26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.26).

[40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_593) Esta es la representación procedimental descrita en el ejercicio [2.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.4). Básicamente, cualquier representación procedimental (por ejemplo, una implementación de paso de mensajes) también funcionaría. Observe que podemos instalar estas definiciones en el evaluador perezoso simplemente escribiéndolas en el bucle del controlador. Si originalmente hubiéramos incluidocons,carycdrcomo primitivos en el entorno global, se redefinirán. (Vea también los ejercicios [4.33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.33)y [4.34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.34)).

[41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_594) Esto nos permite crear versiones retrasadas de tipos más generales deestructuras de listas, no sólo secuencias. Hughes 1990 analiza algunasAplicaciones de los "árboles perezosos".

[**4.3 Variaciones de un esquema: computación no determinista**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_4.3)

En esta sección, ampliamos el evaluador de Scheme para que admita un paradigma de programación llamado *computación no determinista* , incorporando al evaluador una función que admita la búsqueda automática. Este es un cambio mucho más profundo en el lenguaje que la introducción de la evaluación diferida en la sección  [4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2) .

La computación no determinista, como el procesamiento de flujos, es útil para aplicaciones de "generación y prueba". Considere la tarea de comenzar con dos listas de números enteros positivos y encontrar un par de números enteros (uno de la primera lista y uno de la segunda lista) cuya suma sea primo. Vimos cómo manejar esto con operaciones de secuencia finita en la sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) y con flujos infinitos en la sección  [3.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.3) . Nuestro enfoque fue generar la secuencia de todos los pares posibles y filtrarlos para seleccionar los pares cuya suma sea primo. Si realmente generamos primero la secuencia completa de pares como en el capítulo 2, o intercalamos la generación y el filtrado como en el capítulo 3, es irrelevante para la imagen esencial de cómo se organiza el cálculo.

El enfoque no determinista evoca una imagen diferente. Imaginemos simplemente que elegimos (de alguna manera) un número de la primera lista y un número de la segunda lista y exigimos (utilizando algún mecanismo) que su suma sea primo. Esto se expresa mediante el siguiente procedimiento:

(define (par-suma-primo lista1 lista2)   
  (let ((a (un-elemento-de lista1))   
        (b (un-elemento-de lista2)))   
    (requiere (¿primo? (+ a b)))   
    (lista a b)))

Puede parecer que este procedimiento simplemente replantea el problema, en lugar de especificar una forma de resolverlo. Sin embargo, se trata de un programa no determinista legítimo. [42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_598)

La idea clave aquí es que las expresiones en un lenguaje no determinista pueden tener más de un valor posible. Por ejemplo, an-element-of podría devolver cualquier elemento de la lista dada. Nuestro evaluador de programas no determinista funcionará eligiendo automáticamente un valor posible y haciendo un seguimiento de la elección. Si no se cumple un requisito posterior, el evaluador intentará una elección diferente y seguirá probando nuevas opciones hasta que la evaluación tenga éxito o hasta que nos quedemos sin opciones. Así como el evaluador perezoso liberó al programador de los detalles de cómo se retrasan y fuerzan los valores, el evaluador de programas no determinista liberará al programador de los detalles de cómo se toman las decisiones.

Resulta instructivo contrastar las diferentes imágenes del tiempo que evocan la evaluación no determinista y el procesamiento de flujo. El procesamiento de flujo utiliza la evaluación perezosa para disociar el momento en que se ensambla el flujo de posibles respuestas del momento en que se producen los elementos reales del flujo. El evaluador sustenta la ilusión de que todas las posibles respuestas se presentan ante nosotros en una secuencia atemporal. Con la evaluación no determinista, una expresión representa la exploración de un conjunto de mundos posibles, cada uno determinado por un conjunto de opciones. Algunos de los mundos posibles conducen a callejones sin salida, mientras que otros tienen valores útiles. El evaluador de programas no determinista sustenta la ilusión de que el tiempo se bifurca y que nuestros programas tienen diferentes historias de ejecución posibles. Cuando llegamos a un callejón sin salida, podemos volver a visitar un punto de elección anterior y continuar por una rama diferente.

El evaluador de programas no determinista implementado a continuación se denomina evaluador amb porque se basa en una nueva forma especial llamada amb . Podemos escribir la definición anterior de prime-sum-pair en el bucle del controlador del evaluador amb (junto con las definiciones de prime? , an-element-of y require ) y ejecutar el procedimiento de la siguiente manera:

*;;; Entrada de Amb-Eval:*  
(par-suma-primo '(1 3 5 8) '(20 35 110))   
*;;; Comenzando un nuevo problema*   
*;;; Valor de Amb-Eval:*   
*(3 20)*

El valor devuelto se obtuvo después de que el evaluador eligió repetidamente elementos de cada una de las listas, hasta que se realizó una elección exitosa.

La sección  [4.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3.1) presenta amb y explica cómo admite el no determinismo a través del mecanismo de búsqueda automática del evaluador. La sección [4.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3.2) presenta ejemplos de programas no deterministas y la sección  [4.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3.3) brinda detalles sobre cómo implementar el evaluador amb modificando el evaluador Scheme común.

**[4.3.1 Ambientador y búsqueda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.3.1)**

Para ampliar Scheme para que admita el no determinismo, introducimos una nueva forma especial llamada amb . [43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_599) La expresión (amb < *e 1* > < *e 2* > ... < *e n* >) devuelve el valor de una de las *n* expresiones < *e i* > "ambiguamente". Por ejemplo, la expresión

(lista (amb 1 2 3) (amb 'a 'b))

Puede tener seis valores posibles:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (1a) | (1b) | (2a) | (2b) | (3a) | (3b) |

Amb con una sola elección produce un valor ordinario (único).

Amb sin opciones - la expresión (amb) - es una expresión sin valores aceptables. Operativamente, podemos pensar en (amb) como una expresión que cuando se evalúa hace que el cálculo "falle": el cálculo se cancela y no se produce ningún valor. Usando esta idea, podemos expresar el requisito de que una expresión predicativa particular p debe ser verdadera de la siguiente manera:

(definir (requerir p)   
  (si (no p) (amb)))

Con amb y require , podemos implementar el procedimiento an-element-of utilizado anteriormente:

(define (un-elemento-de-elementos)   
  (requiere (no (¿elementos nulos?)))   
  (amb (car elementos) (un-elemento-de (cdr elementos))))

An-element-of falla si la lista está vacía. De lo contrario, devuelve de forma ambigua el primer elemento de la lista o un elemento elegido del resto de la lista.

También podemos expresar rangos infinitos de opciones. El siguiente procedimiento potencialmente devuelve cualquier entero mayor o igual a un número  *n* dado :

(define (un entero que comienza desde n)   
  (amb n (un entero que comienza desde (+ n 1))))

Esto es como el procedimiento de flujo enteros-empezando-desde descrito en la sección  [3.5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.2) , pero con una diferencia importante: el procedimiento de flujo devuelve un objeto que representa la secuencia de todos los enteros que comienzan con *n* , mientras que el procedimiento amb devuelve un solo entero. [44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_600)

De manera abstracta, podemos imaginar que evaluar una expresión amb hace que el tiempo se divida en ramas, donde el cálculo continúa en cada rama con uno de los valores posibles de la expresión. Decimos que amb representa una*punto de elección no determinista* . Si tuviéramos una máquina con una cantidad suficiente de procesadores que pudieran asignarse dinámicamente, podríamos implementar la búsqueda de una manera sencilla. La ejecución procedería como en una máquina secuencial, hasta que se encuentre una expresión amb . En este punto, se asignarían e inicializarían más procesadores para continuar con todas las ejecuciones paralelas implicadas por la elección. Cada procesador procedería secuencialmente como si fuera la única opción, hasta que termine al encontrar una falla, o se subdivida más, o finalice. [45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_601)

Por otro lado, si tenemos una máquina que puede ejecutar sólo un proceso (o unos pocos procesos concurrentes), debemos considerar las alternativas secuencialmente. Uno podría imaginar modificar un evaluador para elegir al azar una rama a seguir cada vez que encuentre un punto de elección. La elección aleatoria, sin embargo, puede conducir fácilmente a valores fallidos. Podríamos intentar ejecutar el evaluador una y otra vez, haciendo elecciones aleatorias y esperando encontrar un valor que no falle, pero es mejor*buscar sistemáticamente* todas las posibles rutas de ejecución. El evaluador amb que desarrollaremos y con el que trabajaremos en esta sección implementa una búsqueda sistemática de la siguiente manera: cuando el evaluador encuentra una aplicación de amb , inicialmente selecciona la primera alternativa. Esta selección puede conducir a una elección adicional. El evaluador siempre elegirá inicialmente la primera alternativa en cada punto de elección. Si una elección falla, entonces el evaluadorautomágicamente [46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_602) *retrocede* hasta el punto de elección más reciente y prueba la siguiente alternativa. Si se queda sin alternativas en cualquier punto de elección, el evaluador retrocederá hasta el punto de elección anterior y continuará desde allí. Este proceso conduce a una estrategia de búsqueda conocida como*búsqueda en profundidad* o *retroceso cronológico* . [47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_603)

**[Bucle del conductor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_604)**

El bucle de controlador del evaluador amb tiene algunas propiedades inusuales. Lee una expresión e imprime el valor de la primera ejecución que no falla, como en el ejemplo de par de suma prima que se muestra arriba. Si queremos ver el valor de la próxima ejecución exitosa, podemos pedirle al intérprete que retroceda e intente generar una segunda ejecución que no falle. Esto se indica escribiendo el símbolotry-again . Si se proporciona cualquier expresión excepto try-again , el intérprete iniciará un nuevo problema y descartará las alternativas no exploradas en el problema anterior. A continuación se muestra un ejemplo de interacción:

*;;; Entrada Amb-Eval:*  
(prime-sum-pair '(1 3 5 8) '(20 35 110))   
*;;; Comenzando un nuevo problema*   
*;;; Valor Amb-Eval:*   
*(3 20)*   
*;;; Entrada Amb-Eval:*  
intentar de nuevo   
*;;; Valor Amb-Eval:*   
*(3 110)*   
*;;; Entrada Amb-Eval:*  
intentar de nuevo   
*;;; Valor Amb-Eval:*   
*(8 35)*   
*;;; Entrada Amb-Eval:*  
intentar de nuevo   
*;;; No hay más valores de*   
*(prime-sum-pair (quote (1 3 5 8)) (quote (20 35 110)))*   
*;;; Entrada Amb-Eval:*  
(prime-sum-pair '(19 27 30) '(11 36 58))   
*;;; Comenzando un nuevo problema*   
*;;; Valor de Amb-Eval:*   
*(30 11)*

**Ejercicio 4.35.**  Escriba un procedimiento an-integer-between que devuelva un entero entre dos límites dados. Esto se puede utilizar para implementar un procedimiento que encuentre ternas pitagóricas, es decir, ternas de enteros ( *i* , *j* , *k* ) entre los límites dados tales que *i* < *j* y *i* 2 + *j* 2 = *k* 2 , de la siguiente manera:

(define (una-triple-pitagórica-entre bajo alto)   
  (sea ((i (un-entero-entre bajo alto)))   
    (sea ((j (un-entero-entre i alto)))   
      (sea ((k (un-entero-entre j alto)))   
        (requiere (= (+ (\* i i) (\* j j)) (\* k k)))   
        (lista i j k)))))

**Ejercicio 4.36.**  En el ejercicio  [3.69](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.69) se analizó cómo generar el flujo de *todas* las ternas pitagóricas, sin límite superior en el tamaño de los enteros que se buscarán. Explique por qué simplemente reemplazar un entero entre por un entero a partir de en el procedimiento del ejercicio  [4.35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.35) no es una forma adecuada de generar ternas pitagóricas arbitrarias. Escriba un procedimiento que realmente logre esto. (Es decir, escriba un procedimiento para el cual escribir repetidamente try-again eventualmente generaría en principio todas las ternas pitagóricas).

**Ejercicio 4.37.**  Ben Bitdiddle afirma que el siguiente método para generar ternas pitagóricas es más eficiente que el del ejercicio  [4.35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.35) . ¿Tiene razón? (Pista: considere la cantidad de posibilidades que se deben explorar).

(define (una-triple-pitagórica-entre bajo alto)   
  (sea ((i (un-entero-entre bajo alto))   
        (hsq (\* alto alto)))   
    (sea ((j (un-entero-entre i alto)))   
      (sea ((ksq (+ (\* i i) (\* j j))))   
        (requiere (>= hsq ksq))   
        (sea ((k (sqrt ksq)))   
          (requiere (entero? k))   
          (lista i j k))))))

**[4.3.2 Ejemplos de programas no deterministas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.3.2)**

La sección  [4.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3.3) describe la implementación del evaluador amb . Sin embargo, primero damos algunos ejemplos de cómo se puede utilizar. La ventaja de la programación no determinista es que podemos suprimir los detalles de cómo se lleva a cabo la búsqueda,expresando nuestros programas a un nivel más alto de abstracción.

**[Rompecabezas de lógica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_608)**

El siguiente rompecabezas (tomado de Dinesman 1968) es típico de una gran clase de rompecabezas de lógica simples:

Baker, Cooper, Fletcher, Miller y Smith viven en pisos diferentes de un edificio de apartamentos que tiene solo cinco pisos. Baker no vive en el piso superior. Cooper no vive en el piso inferior. Fletcher no vive ni en el piso superior ni en el inferior. Miller vive en un piso más alto que Cooper. Smith no vive en un piso adyacente al de Fletcher. Fletcher no vive en un piso adyacente al de Cooper. ¿Dónde vive cada uno?

Podemos determinar quién vive en cada piso de forma sencilla enumerando todas las posibilidades e imponiendo las restricciones dadas: [48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_609)

(define (vivienda múltiple)   
  (let ((panadero (amb 1 2 3 4 5))   
        (tonelero (amb 1 2 3 4 5))   
        (fletcher (amb 1 2 3 4 5)   
        ) (molinero (amb 1 2 3 4 5))   
        (herrero (amb 1 2 3 4 5)))   
    (requiere   
     (distinto? (lista panadero tonelero fletcher molinero herrero)))   
    (requiere (no (= panadero 5)))   
    (requiere (no (= tonelero 1)))   
    (requiere (no (= fletcher 5)))   
    (requiere (no (= fletcher 1)))   
    (requiere (> molinero tonelero))   
    (requiere (no (= (abs (- herrero fletcher)) 1)))   
    (requiere (no (= (abs (- fletcher tonelero)) 1)))   
    (lista (lista 'panadero panadero) (   
          lista 'tonelero tonelero) (   
          lista 'fletcher fletcher)   
          (lista 'molinero molinero)   
          (lista 'herrero herrero))))

La evaluación de la expresión (vivienda múltiple) produce el resultado

((panadero 3) (tonelero 2) (flechero 4) (molinero 5) (herrero 1))

Aunque este procedimiento sencillo funciona, es muy lento. En los ejercicios  [4.39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.39) y  [4.40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.40) se analizan algunas posibles mejoras.

**Ejercicio 4.38.**   Modifique el procedimiento de viviendas múltiples para omitir el requisito de que Smith y Fletcher no vivan en pisos adyacentes. ¿Cuántas soluciones hay para este acertijo modificado?

**Ejercicio 4.39.**   ¿El orden de las restricciones en el procedimiento de residencia múltiple afecta la respuesta? ¿Afecta el tiempo necesario para encontrar una respuesta? Si cree que es importante, demuestre un programa más rápido obtenido a partir del programa dado reordenando las restricciones. Si cree que no es importante, argumente su caso.

**Ejercicio 4.40.**   En el problema de las viviendas múltiples, ¿cuántos conjuntos de asignaciones de personas a pisos hay, tanto antes como después del requisito de que las asignaciones de pisos sean distintas? Es muy ineficiente generar todas las asignaciones posibles de personas a pisos y luego dejar que se vuelva atrás para eliminarlas. Por ejemplo, la mayoría de las restricciones dependen de solo una o dos de las variables persona-piso y, por lo tanto, se pueden imponer antes de que se hayan seleccionado pisos para todas las personas. Escriba y demuestre un procedimiento no determinista mucho más eficiente que resuelva este problema basándose en generar solo aquellas posibilidades que no estén ya descartadas por restricciones anteriores. (Pista: esto requerirá un anidamiento de expresiones let ).

**Ejercicio 4.41.**  Escriba un programa Scheme ordinario para resolver el problema de las viviendas múltiples.

**Ejercicio 4.42.**  Resuelva el siguiente rompecabezas de "Mentirosos" (de Phillips 1934):

Cinco alumnas se presentaron a un examen. Sus padres, según creían, mostraban un interés excesivo por el resultado. Por ello, acordaron que, al escribir a casa sobre el examen, cada una de ellas debía hacer una declaración verdadera y otra falsa. A continuación, se reproducen los pasajes pertinentes de sus cartas:

* Betty: "Kitty quedó en segundo lugar en el examen. Yo quedé en tercer lugar".
* Ethel: ``Te alegrará saber que yo estaba en la cima. Joan quedó en segundo lugar''.
* Joan: «Yo quedé tercera y la pobre Ethel última».
* Kitty: ``Yo quedé segunda. Mary quedó en cuarta posición''.
* María: “Yo quedé cuarta. El primer puesto lo ocupó Betty”.

¿Cuál fue en realidad el orden en que fueron colocadas las cinco niñas?

**Ejercicio 4.43.**   Utilice el evaluador amb para resolver el siguiente acertijo: [49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_616)

El padre de Mary Ann Moore tiene un yate, al igual que cada uno de sus cuatro amigos: el coronel Downing, el señor Hall, Sir Barnacle Hood y el doctor Parker. Cada uno de los cinco tiene también una hija y cada uno ha bautizado su yate con el nombre de la hija de uno de los otros. El yate de Sir Barnacle es el Gabrielle, el señor Moore es dueño del Lorna y el señor Hall, del Rosalind. El Melissa, propiedad del coronel Downing, lleva el nombre de la hija de Sir Barnacle. El padre de Gabrielle es dueño del yate que lleva el nombre de la hija del doctor Parker. ¿Quién es el padre de Lorna?

Intente escribir el programa de forma que se ejecute de manera eficiente (consulte el ejercicio  [4.40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.40) ). Determine también cuántas soluciones hay si no nos dicen que el apellido de Mary Ann es Moore.

**Ejercicio 4.44.**  El ejercicio  [2.42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.42) describe el «problema de las ocho reinas»: colocar reinas en un tablero de ajedrez de modo que ninguna de ellas se ataque entre sí. Escriba un programa no determinista para resolver este problema.

**[Analizando lenguaje natural](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_618)**

Los programas diseñados para aceptar lenguaje natural como entrada generalmente comienzan intentando *analizar* la entrada, es decir, hacerla coincidir con alguna estructura gramatical. Por ejemplo, podríamos intentar reconocer oraciones simples que consten de un artículo seguido de un sustantivo seguido de un verbo, como "El gato come". Para lograr un análisis de este tipo, debemos ser capaces de identificar las partes del discurso de palabras individuales. Podríamos comenzar con algunas listas que clasifiquen varias palabras: [50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_619)

(define sustantivos '(sustantivo estudiante profesor gato clase))  
(define verbos '(verbo estudia conferencias come duerme))  
(define artículos '(artículo el a))

También necesitamos una *gramática* , es decir, un conjunto de reglas que describan cómo se componen los elementos gramaticales a partir de elementos más simples. Una gramática muy simple podría estipular que una oración siempre consta de dos partes: una frase nominal seguida de un verbo, y que una frase nominal consta de un artículo seguido de un sustantivo. Con esta gramática, la oración "El gato come" se analiza de la siguiente manera:

(oración (sintagma nominal (artículo el) (sustantivo gato))   
          (verbo come))

Podemos generar un análisis de este tipo con un programa simple que tenga procedimientos separados para cada una de las reglas gramaticales. Para analizar una oración, identificamos sus dos partes constituyentes y devolvemos una lista de estos dos elementos, etiquetados con el símbolo oración :

(define (analizar-oración)   
  (lista 'oración   
         (analizar-frase-sustantivo)   
         (analizar-palabra-verbos)))

De manera similar, una frase nominal se analiza buscando un artículo seguido de un sustantivo:

(define (parse-noun-phrase)   
  (lista 'parse-phrase   
        (parse-word articles)   
        (parse-word sustantivos)))

En el nivel más bajo, el análisis se reduce a verificar repetidamente que la siguiente palabra no analizada sea miembro de la lista de palabras para la parte del discurso requerida. Para implementar esto, mantenemos una variable global \*unparsed\* , que es la entrada que aún no se ha analizado. Cada vez que verificamos una palabra, requerimos que \*unparsed\* no esté vacía y que comience con una palabra de la lista designada. Si es así, eliminamos esa palabra de \*unparsed\* y devolvemos la palabra junto con su parte del discurso (que se encuentra al principio de la lista): [51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_620)

(define (parse-word lista-de-palabras)   
  (require (not (null? \*unparsed\*)))   
  (require (memq (car \*unparsed\*) (cdr lista-de-palabras)))   
  (let ((palabra-encontrada (car \*unparsed\*)))   
    (set! \*unparsed\* (cdr \*unparsed\*))   
    (list (car lista-de-palabras) palabra-encontrada)))

Para comenzar el análisis, todo lo que necesitamos hacer es establecer \*unparsed\* como la entrada completa, intentar analizar una oración y verificar que no quede nada:

(define \*sin analizar\* '())  
(define (analizar entrada)   
  (establece! entrada \*sin analizar\*)   
  (let ((enviado (analizar-oración)))   
    (requiere (¿nulo? \*sin analizar\*))   
    enviado))

Ahora podemos probar el analizador y verificar que funciona para nuestra oración de prueba simple:

*;;; Entrada de Amb-Eval:*  
(parse '(el gato come))   
*;;; Iniciar un nuevo problema*   
*;;; Valor de Amb-Eval:*   
*(oración (sintagma nominal (artículo el) (sustantivo gato)) (verbo come))*

El evaluador amb es útil aquí porque es conveniente expresar las restricciones de análisis con la ayuda de require . Sin embargo, la búsqueda automática y el retroceso realmente dan sus frutos cuando consideramos gramáticas más complejas donde existen opciones para descomponer las unidades.

Agreguemos a nuestra gramática una lista de preposiciones:

(define preposiciones '(prep para a en por con))

y definir una frase preposicional (por ejemplo, ``para el gato'') como una preposición seguida de una frase nominal:

(define (analizar-frase-preposicional)   
  (lista 'prep-phrase   
        (analizar-preposiciones-de-palabras)   
        (analizar-frase-sustantivo)))

Ahora podemos definir una oración como una frase nominal seguida de una frase verbal, donde una frase verbal puede ser un verbo o una frase verbal extendida por una frase preposicional: [52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_621)

(define (parse-oración)   
  (list 'oración   
         (parse-frase-sustantivo)   
         (parse-frase-verbal)))   
(define (parse-frase-verbal)   
  (define (tal vez-extender frase-verbal)   
    (amb frase-verbal   
         (tal vez-extender (list 'frase-verbal   
                             frase-verbal   
                             (parse-frase-preposicional)))))   
  (tal vez-extender (parse-palabra verbos)))

Mientras estamos en eso, también podemos elaborar la definición de frases nominales para permitir cosas como "un gato en la clase". Lo que solíamos llamar una frase nominal, ahora lo llamaremos una frase nominal simple, y una frase nominal ahora será una frase nominal simple o una frase nominal extendida por una frase preposicional:

(define (parse-simple-noun-phrase)   
  (list 'simple-noun-phrase   
        (parse-word-articles)   
        (parse-word-nouns)))   
(define (parse-noun-phrase)   
  (define (maybe-extend noun-phrase)   
    (amb noun-phrase   
         (maybe-extend (list 'noun-phrase   
                             noun-phrase   
                             (parse-prepositional-phrase)))))   
  (maybe-extend (parse-simple-noun-phrase)))

Nuestra nueva gramática nos permite analizar oraciones más complejas. Por ejemplo

(parse '(el estudiante con el gato duerme en la clase))

produce

(oración   
 (frase nominal   
  (frase nominal simple (artículo el) (sustantivo estudiante))   
  (frase de preparación (preparar con)   
               (frase nominal simple   
                (artículo el) (sustantivo gato))))   
 (frase verbal   
  (verbo duerme)   
  (frase de preparación (preparar en)   
               (frase nominal simple   
                (artículo el) (sustantivo clase)))))

Observe que una entrada dada puede tener más de un análisis legal. En la oración ``El profesor da una clase al estudiante con el gato'', puede ser que el profesor esté dando una clase con el gato, o que el estudiante tenga el gato. Nuestro programa no determinista encuentra ambas posibilidades:

(parse '(el profesor le da una clase al estudiante con el gato))

produce

(oración   
 (frase nominal simple (artículo el) (sustantivo profesor))   
 (frase verbal   
  (frase verbal   
   (verbo da clases)   
   (frase preparatoria (preparar para)   
                (frase nominal simple   
                 (artículo el) (sustantivo estudiante))))   
  (frase preparatoria (preparar con)   
               (frase nominal simple   
                (artículo el) (sustantivo gato)))))

Pedirle al evaluador que lo intente nuevamente da como resultado

(oración   
 (frase nominal simple (artículo el) (sustantivo profesor))   
 (frase verbal   
  (verbo da clases)   
  (frase preparatoria (preparar para)   
               (frase nominal   
                (frase nominal simple   
                 (artículo el) (sustantivo estudiante))   
                (frase preparatoria (preparar con)   
                             (frase nominal simple   
                              (artículo el) (sustantivo gato)))))))

**Ejercicio 4.45.**   Con la gramática dada anteriormente, la siguiente oración puede analizarse de cinco maneras diferentes: ``El profesor da una clase al estudiante en la clase con el gato''. Dé los cinco análisis y explique las diferencias en los matices de significado entre ellos.

**Ejercicio 4.46.**  Los evaluadores de las secciones  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) y [4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2) no determinan en qué orden se evalúan los operandos. Veremos que el evaluador amb los evalúa de izquierda a derecha. Explique por qué nuestro programa de análisis no funcionaría si los operandos se evaluaran en otro orden.

**Ejercicio 4.47.**   Louis Reasoner sugiere que, puesto que un sintagma verbal es un verbo o un sintagma verbal seguido de un sintagma preposicional, sería mucho más sencillo definir el procedimiento analizar-sintagma-verbal de la siguiente manera (y de manera similar para los sintagmas nominales):

(define (analizar-frase-verbal)   
  (amb (analizar-palabra-verbos)   
       (lista 'frase-verbal   
             (analizar-frase-verbal)   
             (analizar-frase-preposicional))))

¿Funciona esto? ¿Cambia el comportamiento del programa si intercambiamos el orden de las expresiones en el amb ?

**Ejercicio 4.48.**   Amplíe la gramática dada anteriormente para manejar oraciones más complejas. Por ejemplo, podría extender frases nominales y verbales para incluir adjetivos y adverbios, o podría manejar oraciones compuestas. [53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_626)

**Ejercicio 4.49.**  A Alyssa P. Hacker le interesa más generar oraciones interesantes que analizarlas. Razona que, si simplemente cambiamos el procedimiento parse-word para que ignore la "oración de entrada" y, en su lugar, siempre tenga éxito y genere una palabra apropiada, podemos usar los programas que habíamos creado para analizar para realizar la generación en su lugar. Implementemos la idea de Alyssa y mostremos la primera media docena de oraciones generadas. [54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_628)

**[4.3.3 Implementación del evaluador ambiental](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.3.3)**

La evaluación de una expresión de Scheme ordinaria puede devolver un valor, no terminar nunca o indicar un error. En Scheme no determinista, la evaluación de una expresión puede además dar como resultado el descubrimiento de un callejón sin salida, en cuyo caso la evaluación debe retroceder a un punto de elección anterior. La interpretación de Scheme no determinista se complica por este caso adicional.

Construiremos el evaluador amb para Scheme no determinista modificando el evaluador analizador de la sección  [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) . [55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_629) Al igual que en el evaluador analizador, la evaluación de una expresión se logra llamando a unprocedimiento de ejecución producido por el análisis de esa expresión. La diferencia entre la interpretación del esquema ordinario y la interpretación del esquema no determinista estará enteramente en los procedimientos de ejecución.

**[Procedimientos de ejecución y continuaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_630)**

Recuerde que los procedimientos de ejecución del evaluador ordinario toman un argumento: el entorno de ejecución. En cambio, los procedimientos de ejecución del evaluador amb toman tres argumentos: el entorno y dos procedimientos llamados *procedimientos de continuación* . La evaluación de una expresión finalizará llamando a una de estas dos continuaciones: Si la evaluación da como resultado un valor, el*La continuación del éxito* se llama con ese valor; si la evaluación da como resultado el descubrimiento de un callejón sin salida,*Se llama a la continuación de la falla* . Construir y llamar a las continuaciones apropiadas es el mecanismo por el cual el evaluador no determinista implementa el retroceso.

La función de continuación exitosa es recibir un valor y continuar con el cálculo. Junto con ese valor, a la continuación exitosa se le pasa otra continuación fallida, que se debe llamar posteriormente si el uso de ese valor conduce a un callejón sin salida.

La tarea de la continuación de fracaso es probar otra rama del proceso no determinista. La esencia del lenguaje no determinista reside en el hecho de que las expresiones pueden representar elecciones entre alternativas. La evaluación de una expresión de este tipo debe realizarse con una de las alternativas indicadas, aunque no se sepa de antemano qué opciones conducirán a resultados aceptables. Para ello, el evaluador elige una de las alternativas y pasa este valor a la continuación de éxito. Junto con este valor, el evaluador construye y pasa una continuación de fracaso que puede ser invocada más tarde para elegir una alternativa diferente.

Se activa un fallo durante la evaluación (es decir, se invoca una continuación de fallo) cuando un programa de usuario rechaza explícitamente la línea de ataque actual (por ejemplo, una llamada a require puede dar como resultado la ejecución de (amb) , una expresión que siempre falla - consulte la sección  [4.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3.1) ). La continuación de fallo en mano en ese punto hará que el punto de elección más reciente elija otra alternativa. Si no hay más alternativas para considerar en ese punto de elección, se activa un fallo en un punto de elección anterior, y así sucesivamente. Las continuaciones de fallo también son invocadas por el bucle del controlador en respuesta a una solicitud de intentar de nuevo , para encontrar otro valor de la expresión.

Además, si se produce una operación de efecto secundario (como la asignación a una variable) en una rama del proceso resultante de una elección, puede ser necesario, cuando el proceso llega a un punto muerto, deshacer el efecto secundario antes de realizar una nueva elección. Esto se logra haciendo que la operación de efecto secundario produzca una continuación de falla que deshaga el efecto secundario y propague la falla.

En resumen, las continuaciones de fallas se construyen mediante

* expresiones amb - para proporcionar un mecanismo para realizar elecciones alternativas si la elección actual hecha por la expresión amb conduce a un callejón sin salida;
* el controlador de nivel superior: proporcionar un mecanismo para informar el fallo cuando se agotan las opciones;
* asignaciones - para interceptar fallas y deshacer asignaciones durante el retroceso.

Los fallos se inician solo cuando se encuentra un callejón sin salida. Esto ocurre

* si el programa de usuario ejecuta (amb) ;
* Si el usuario escribe "try-again" en el controlador de nivel superior.

Las continuaciones de fallos también se denominan durante el procesamiento de un fallo:

* Cuando la continuación de falla creada por una asignación termina de deshacer un efecto secundario, llama a la continuación de falla que interceptó para propagar la falla nuevamente al punto de elección que condujo a esta asignación o al nivel superior.
* Cuando la continuación de falla de un amb se queda sin opciones, llama a la continuación de falla que se le dio originalmente al amb , para propagar la falla nuevamente al punto de elección anterior o al nivel superior.

**[Estructura del evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_631)**

Los procedimientos de representación de datos y sintaxis para el evaluador amb , y también el procedimiento de análisis básico , son idénticos a los del evaluador de la sección  [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) , excepto por el hecho de que necesitamos procedimientos de sintaxis adicionales para reconocer la forma especial amb : [56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_632)

(definir (amb? exp) (lista de etiquetas? exp 'amb))   
(definir (amb-choices exp) (cdr exp))

También debemos agregar al envío en análisis una cláusula que reconozca esta forma especial y genere un procedimiento de ejecución apropiado:

((amb? exp) (analizar-amb exp))

El procedimiento de nivel superior ambeval (similar a la versión de eval dada en la sección  [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) ) analiza la expresión dada y aplica el procedimiento de ejecución resultante al entorno dado, junto con dos continuaciones dadas:

(define (ambeval exp env éxito falla)   
  ((analiza exp) env éxito falla))

Una continuación exitosa es un procedimiento con dos argumentos: el valor que se acaba de obtener y otra continuación fallida que se utilizará si ese valor conduce a una falla posterior. Una continuación fallida es un procedimiento sin argumentos.La forma general de un procedimiento de ejecución es

(lambda (env éxito falla)   
  *;;  éxito  es  (lambda (valor falla)  ... )*  
   *;;  falla  es  (lambda ()  ... )*  
   ... )

Por ejemplo, ejecutar

(ambeval < *exp* >   
         el-entorno-global   
         (lambda (valor falla) valor)   
         (lambda () 'falló))

intentará evaluar la expresión dada y devolverá el valor de la expresión (si la evaluación tiene éxito) o el símbolo fallido (si la evaluación falla). La llamada a ambeval en el bucle del controlador que se muestra a continuación utiliza procedimientos de continuación mucho más complicados, que continúan el bucle y admiten la solicitud de intentarlo de nuevo .

La mayor parte de la complejidad del evaluador amb resulta de la mecánica de pasar las continuaciones a medida que los procedimientos de ejecución se llaman entre sí. Al revisar el código siguiente, debe comparar cada uno de los procedimientos de ejecución con el procedimiento correspondiente para el evaluador ordinario que se proporciona en la sección  [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) .

**[Expresiones simples](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_633)**

Los procedimientos de ejecución para los tipos de expresiones más simples son básicamente los mismos que para el evaluador ordinario, excepto por la necesidad de gestionar las continuaciones. Los procedimientos de ejecución simplemente tienen éxito con el valor de la expresión, pasando la continuación fallida que se les pasó.

(define (analiza-autoevaluación exp)   
  (lambda (entorno éxito fallo)   
    (éxito exp fallo)))   
(define (analiza-entrecomillado exp)   
  (let ((qval (texto-de-comilla exp)))   
    (lambda (entorno éxito fallo)   
      (éxito qval fallo))))   
(define (analiza-variable exp)   
  (lambda (entorno éxito fallo)   
    (éxito (valor-variable-búsqueda exp env)   
             fallo)))   
(define (analiza-lambda exp)   
  (let ((vars (parámetros-lambda exp))   
        (bproc (analiza-secuencia (cuerpo-lambda exp))))   
    (lambda (entorno éxito fallo)   
      (éxito (procedimiento-hacer vars bproc env)   
               fallo))))

Tenga en cuenta que la búsqueda de una variable siempre "tiene éxito". Si lookup-variable-value no encuentra la variable, indica un error, como es habitual. Un "error" de este tipo indica un error del programa (una referencia a una variable no vinculada); no es una indicación de que debamos probar otra opción no determinista en lugar de la que se está probando actualmente.

**[Condicionales y secuencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_634)**

Los condicionales también se manejan de manera similar al evaluador ordinario. El procedimiento de ejecución generado por analyze-if invoca el procedimiento de ejecución de predicado pproc con una continuación de éxito que verifica si el valor del predicado es verdadero y continúa ejecutando el consecuente o la alternativa. Si la ejecución de pproc falla, se llama a la continuación de falla original para la expresión if .

(define (analyze-if exp)   
  (let ((pproc (analyze (if-predicate exp)))   
        (cproc (analyze (if-consequen exp)))   
        (aproc (analyze (if-alternative exp))))   
    (lambda (env success fail)   
      (pproc env   
             *;; continuación de éxito para evaluar el predicado*   
             *;; para obtener  pred-value*  
             (lambda (pred-value fail2)   
               (if (true? pred-value)   
                   (cproc env success fail2)   
                   (aproc env success fail2)))   
             *;; continuación de error para evaluar el predicado*  
             fail))))

Las secuencias también se manejan de la misma manera que en el evaluador anterior, excepto por las maquinaciones en el subprocedimiento sequentially que se requieren para pasar las continuaciones. Es decir, para ejecutar secuencialmente a y luego b , llamamos a con una continuación exitosa que llama a b .

(define (analyze-secuencia exps)   
  (define (secuencialmente a b)   
    (lambda (env success fail)   
      (a env   
         *;; éxito continuación para llamar  a un*  
         (lambda (a-value fail2)   
           (b env success fail2))   
         *;; error continuación para llamar  a un*  
         fail)))   
  (define (loop first-proc rest-procs)   
    (if (null? rest-procs)   
        first-proc   
        (loop (secuencialmente first-proc (car rest-procs))   
              (cdr rest-procs))))   
  (let ((procs (map analyze exps)))   
    (if (null? procs)   
        (error "Secuencia vacía -- ANALYZE"))   
    (loop (car procs) (cdr procs))))

**[Definiciones y asignaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_635)**

Las definiciones son otro caso en el que debemos tomarnos algunas molestias para gestionar las continuaciones, ya que es necesario evaluar la expresión del valor de la definición antes de definir realmente la nueva variable. Para lograr esto, se llama al procedimiento de ejecución del valor de la definición vproc con el entorno, una continuación de éxito y la continuación de error. Si la ejecución de vproc tiene éxito y se obtiene un valor val para la variable definida, se define la variable y se propaga el éxito:

(define (analiza-definición exp)   
  (let ((var (variable-definición exp))   
        (vproc (analiza (valor-definición exp))))   
    (lambda (env éxito fallo)   
      (vproc env                           
             (lambda (val fallo2)   
               (variable-definición! var val env)   
               (éxito 'ok fallo2))   
             fallo))))

Las asignaciones son más interesantes. Este es el primer lugar donde realmente usamos las continuaciones, en lugar de simplemente pasarlas de mano en mano. El procedimiento de ejecución de las asignaciones comienza como el de las definiciones. Primero intenta obtener el nuevo valor que se asignará a la variable. Si esta evaluación de vproc falla, la asignación falla.

Sin embargo, si vproc tiene éxito y continuamos con la asignación, debemos considerar la posibilidad de que esta rama del cálculo pueda fallar más adelante, lo que nos obligará a dar marcha atrás en la asignación. Por lo tanto, debemos organizarnos para deshacer la asignación como parte del proceso de retroceso. [57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_636)

Esto se logra dando a vproc una continuación de éxito (marcada con el comentario ``\*1\*'' a continuación) que guarda el valor anterior de la variable antes de asignar el nuevo valor a la variable y continuar con la asignación. La continuación de error que se pasa junto con el valor de la asignación (marcada con el comentario ``\*2\*'' a continuación) restaura el valor anterior de la variable antes de continuar con el error. Es decir, una asignación exitosa proporciona una continuación de error que interceptará un error posterior; cualquier error que de otro modo hubiera llamado a fail2 llama a este procedimiento en su lugar, para deshacer la asignación antes de llamar realmente a fail2 .

(define (analiza-asignación exp)   
  (let ((var (variable-asignación exp))   
        (vproc (analiza (valor-asignación exp))))   
    (lambda (entorno éxito fallo)   
      (vproc env   
             (lambda (val fallo2)         *; \*1\**  
               (let ((valor-antiguo   
                      (valor-variable-búsqueda var env)))    
                 (valor-variable-establecido! var val env)   
                 (éxito 'ok   
                          (lambda ()     *; \*2\**  
                            (valor-variable-establecido! var   
                                                 valor-antiguo   
                                                 env)   
                            (fallo2)))))   
             fallo))))

**[Aplicaciones de procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_637)**

El procedimiento de ejecución de aplicaciones no contiene ideas nuevas, excepto la complejidad técnica de la gestión de las continuaciones. Esta complejidad surge en analyze-application , debido a la necesidad de realizar un seguimiento de las continuaciones exitosas y fallidas a medida que evaluamos los operandos. Usamos un procedimiento get-args para evaluar la lista de operandos, en lugar de un mapa simple como en el evaluador ordinario.

(define (analiza-aplicación exp)   
  (let ((fproc (analiza (operador exp)))   
        (aprocs (map analiza (operandos exp))))   
    (lambda (entorno éxito fallo)   
      (fproc env   
             (lambda (proc fallo2)   
               (obtiene-argumentos aprocs   
                         env   
                         (lambda (argumentos fallo3)   
                           (ejecuta-aplicación   
                            proc argumentos éxito fallo3))   
                         fallo2))   
             fallo))))

En get-args , observe cómo la ejecución de cdr en la lista de procedimientos de ejecución de aproc y la ejecución de cons en la lista resultante de args se logra llamando a cada aproc en la lista con una continuación de éxito que llama recursivamente a get-args . Cada una de estas llamadas recursivas a get-args tiene una continuación de éxito cuyo valor es el cons del argumento recién obtenido en la lista de argumentos acumulados:

(define (get-args aprocs env success fail)   
  (if (null? aprocs)   
      (succeed '() fail)   
      ((car aprocs) env   
                    *;; éxito continuación para este  aproc*  
                    (lambda (arg fail2)   
                      (get-args (cdr aprocs)   
                                env   
                                *;; éxito continuación para la llamada recursiva*   
                                *;; a  get-args*  
                                (lambda (args fail3)   
                                  (succeed (cons arg args)   
                                           fail3))   
                                fail2))   
                    fail)))

La aplicación del procedimiento real, que se lleva a cabo mediante la ejecución de la aplicación , se lleva a cabo de la misma manera que para el evaluador ordinario, excepto por la necesidad de gestionar las continuaciones.

(define (execute-application proc args success fail)   
  (cond ((primitive-procedure? proc)   
         (succeed (apply-primitive-procedure proc args)   
                  fail))   
        ((comound-procedure? proc)   
         ((procedure-body proc)   
          (extend-environment (procedure-parameters)   
                              args   
                              (procedure-environment proc))   
          success   
          fail))   
        (else   
         (error   
          "Tipo de procedimiento desconocido -- EXECUTE-APPLICATION"   
          proc))))

**[Evaluación de expresiones amb](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_638)**

La forma especial de amb es el elemento clave en el lenguaje no determinista. Aquí vemos la esencia del proceso de interpretación y la razón para mantener un registro de las continuaciones. El procedimiento de ejecución de amb define un bucle try-next que recorre los procedimientos de ejecución para todos los valores posibles de la expresión amb . Cada procedimiento de ejecución se llama con una continuación fallida que intentará la siguiente. Cuando no hay más alternativas para probar, falla toda la expresión amb .

(define (analiza-amb exp)   
  (let ((cprocs (map analiza (amb-elecciones exp))))   
    (lambda (entorno éxito falla)   
      (define (try-next elecciones)   
        (if (null? elecciones)   
            (falla)   
            ((car elecciones) env   
                           éxito   
                           (lambda ()   
                             (try-next (cdr elecciones)))))   
      (try-next cprocs))))

**[Bucle del conductor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_639)**

El bucle del controlador para el evaluador amb es complejo debido al mecanismo que permite al usuario volver a intentar evaluar una expresión. El controlador utiliza un procedimiento llamado internal-loop , que toma como argumento un procedimiento try-again . La intención es que al llamar a try-again se continúe con la siguiente alternativa no probada en la evaluación no determinista. Internal-loop llama a try-again en respuesta al usuario que escribe try-again en el bucle del controlador, o bien inicia una nueva evaluación llamando a ambeval .

La continuación de la falla de esta llamada a ambeval informa al usuario que no hay más valores y vuelve a invocar el bucle del controlador.

La continuación de éxito para la llamada a ambeval es más sutil. Imprimimos el valor obtenido y luego invocamos el bucle interno nuevamente con un procedimiento de prueba de nuevo que podrá probar la siguiente alternativa. Este procedimiento de siguiente alternativa es el segundo argumento que se pasó a la continuación de éxito. Por lo general, pensamos en este segundo argumento como una continuación de falla que se utilizará si la rama de evaluación actual falla más adelante. En este caso, sin embargo, hemos completado una evaluación exitosa, por lo que podemos invocar la rama alternativa de ``falla'' para buscar evaluaciones exitosas adicionales.

(define el mensaje de entrada ";;; Entrada de Amb-Eval:")   
(define el mensaje de salida ";;; Valor de Amb-Eval:")  
(define (bucle-controlador)   
  (define (bucle-interno intentar-de-nuevo)   
    (solicitud-de-entrada solicitud-de-entrada)   
    (let ((entrada (leer)))   
      (if (eq? entrada 'intentar-de-nuevo)   
          (intentar-de-nuevo)   
          (begin   
            (nueva-línea)   
            (mostrar ";;; Comenzando un nuevo problema ")   
            (ambeval entrada   
                     el-entorno-global   
                     *;;  ambeval  éxito*  
                     (lambda (val siguiente-alternativa)   
                       (anuncio-salida solicitud-de-salida)   
                       (impresión-usuario val)   
                       (bucle-interno siguiente-alternativa))   
                     *;;  ambeval  falla*  
                     (lambda ()   
                       (anuncio-salida   
                        ";;; No hay más valores de")   
                       (impresión-usuario entrada)   
                       (bucle-controlador))))))   
  (bucle-interno   
   (lambda ()   
     (nueva-línea)   
     (mostrar ";;; No hay ningún problema actual")   
     (bucle-controlador))))

La llamada inicial a internal-loop utiliza un procedimiento try-again que se queja de que no hay ningún problema actual y reinicia el bucle del controlador. Este es el comportamiento que se producirá si el usuario escribe try-again cuando no hay ninguna evaluación en curso.

**Ejercicio 4.50.**   Implemente una nueva forma especial ramb que sea como amb, excepto que busque alternativas en un orden aleatorio, en lugar de hacerlo de izquierda a derecha. Muestre cómo esto puede ayudar con el problema de Alyssa en el ejercicio  [4.49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.49) .

**Ejercicio 4.51.**   Implemente un nuevo tipo de asignación llamada ¡conjunto permanente! que no se deshaga en caso de falla. Por ejemplo, podemos elegir dos elementos distintos de una lista y contar la cantidad de intentos necesarios para realizar una elección correcta de la siguiente manera:

(define count 0)   
(let ((x (un-elemento-de '(a b c)))   
      (y (un-elemento-de '(a b c))))   
  (conjunto-permanente! count (+ count 1))   
  (require (not (eq? x y)))   
  (list x y count))   
*;;; Comenzando un nuevo problema*   
*;;; Valor de Amb-Eval:*   
*(a b 2)*   
*;;; Entrada de Amb-Eval:*  
try-again   
*;;; Valor de Amb-Eval:*   
*(a c 3)*

¿Qué valores se habrían mostrado si hubiéramos utilizado set! aquí en lugar de permanent-set !?

**Ejercicio 4.52.**   Implemente una nueva construcción denominada if-fail que permita al usuario detectar el error de una expresión. If-fail toma dos expresiones. Evalúa la primera expresión como siempre y retorna como siempre si la evaluación es correcta. Sin embargo, si la evaluación falla, se retorna el valor de la segunda expresión, como en el siguiente ejemplo:

*;;; Entrada de Amb-Eval:*  
(if-fail (let ((x (an-element-of '(1 3 5))))   
           (require (even? x))   
           x)   
         'all-odd)   
*;;; Comenzando un nuevo problema*   
*;;; Valor de Amb-Eval:*   
*all-odd*   
*;;; Entrada de Amb-Eval:*  
(if-fail (let ((x (an-element-of '(1 3 5 8))))   
           (require (even? x))   
           x)   
         'all-odd)   
*;;; Comenzando un nuevo problema*   
*;;; Valor de Amb-Eval:*   
*8*

**Ejercicio 4.53.**   Con ¡definición permanente! como se describe en el ejercicio  [4.51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.51) y si falla como en el ejercicio  [4.52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.52) , ¿cuál será el resultado de evaluar

(let ((pares '()))   
  (si falla (let ((p (par-suma-prima '(1 3 5 8) '(20 35 110))))   
             (conjunto-permanente! pares (cons p pares))   
             (amb))   
           pares))

**Ejercicio 4.54.**  Si no nos hubiéramos dado cuenta de que require se podía implementar como un procedimiento ordinario que utiliza amb , para ser definido por el usuario como parte de un programa no determinista, habríamos tenido que implementarlo como una forma especial. Esto requeriría procedimientos de sintaxis

(define (requiere? exp) (lista-etiquetada? exp 'requiere))   
  
(define (requiere-predicado exp) (cadr exp))

y una nueva cláusula en el despacho en analizar

((requerir? exp) (analizar-requerir exp))

así como el procedimiento analyze-require que maneja expresiones require . Complete la siguiente definición de analyze-require .

(define (analiza-requiere exp)   
  (let ((pproc (analiza (requiere-predicado exp))))   
    (lambda (entorno éxito fallo)   
      (pproc entorno   
             (lambda (valor-predecesor fallo2)   
               (si < *??* >   
                   < *??* >   
                   (éxito 'ok fallo2)))   
             fallo))))

[42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_598) Suponemos que hemos definido previamente un procedimientoprime?que prueba si los números son primos. Incluso conprime?definido, elde pares de suma de primospuede parecerse sospechosamente al inútil intento de ``pseudo-Lisp'' de definir la función raíz cuadrada, que describimos al principio de la sección [1.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.7). De hecho, un procedimiento de raíz cuadrada en esa línea puede formularse realmente como un programa no determinista. Al incorporar un mecanismo de búsqueda en el evaluador, estamos erosionando laDistinción entre descripciones puramente declarativas y especificaciones imperativas de cómo calcular las respuestas. Iremos más allá en esta dirección en la sección  [4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4) .

[43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_599) La idea deambpara la programación no determinista fuedescrito por primera vez en 1961 por John McCarthy (ver McCarthy 1967).

[44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_600) En realidad, la distinción entre devolver de manera no determinista una única opción y devolver todas las opciones depende en cierta medida de nuestro punto de vista. Desde la perspectiva del código que utiliza el valor, la opción no determinista devuelve un único valor. Desde la perspectiva del programador que diseña el código, la opción no determinista devuelve potencialmente todos los valores posibles y el cálculo se ramifica de modo que cada valor se investiga por separado.

[45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_601) Se podría objetar que se trata de un mecanismo desesperadamente ineficiente. Podrían necesitarse millones de procesadores para resolver un problema fácil de resolver de esta manera, y la mayor parte del tiempo la mayoría de esos procesadores estarían inactivos. Esta objeción debe tomarse en el contexto de la historia. La memoria solía considerarse un producto tan caro.En 1964, un megabyte de RAM costaba unos 400.000 dólares. Hoy en día, todos los ordenadores personales tienen muchos megabytes de RAM y, la mayor parte del tiempo, la mayor parte de esa RAM no se utiliza. Es difícil subestimar el coste de los productos electrónicos producidos en serie.

[46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_602) Automágicamente: ``Automáticamente, pero de una manera que, por alguna razón (normalmente porque es demasiado complicado, o demasiado feo, o quizás incluso demasiado trivial), el hablante no tiene ganas de explicarlo.'' (Steele 1983, Raymond 1993)

[47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_603) La integración de estrategias de búsqueda automáticaLa introducción de algoritmos no deterministas en lenguajes de programación tiene una larga y accidentada historia. Las primeras sugerencias de que los algoritmos no deterministas podrían codificarse elegantemente en un lenguaje de programación con búsqueda y retroceso automático surgieron deRobert Floyd (1967).Carl Hewitt (1969) inventó un lenguaje de programación llamadoPlanificador que admitía explícitamente el retroceso cronológico automático y proporcionaba una estrategia de búsqueda en profundidad incorporada.Sussman, Winograd y Charniak (1971) implementaron un subconjunto de este lenguaje, llamadoMicroPlanner, que se utilizó para apoyar el trabajo de resolución de problemas y planificación de robots. Ideas similares, surgidas de la lógica y la demostración de teoremas, llevaron a la génesis en Edimburgo y Marsella del elegante lenguajeProlog (que analizaremos en la sección  [4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4) ). Después de una frustración suficiente con la búsqueda automática,McDermott y Sussman (1972) desarrollaron un lenguaje llamadoConniver, que incluía mecanismos para poner la estrategia de búsqueda bajo el control del programador. Sin embargo, esto resultó difícil de manejar ySussman y Stallman (1975) encontraron un enfoque más manejable mientras investigaban métodos de análisis simbólico para circuitos eléctricos. Desarrollaron un esquema de rastreo no cronológico que se basaba en rastrear las dependencias lógicas que conectaban hechos, una técnica que se conoce como*retroceso dirigido por dependencia* . Aunque su método era complejo, producía programas razonablemente eficientes porque realizaba pocas búsquedas redundantes.Doyle (1979) y McAllester (1978, 1980) generalizaron y aclararon los métodos de Stallman y Sussman, desarrollando un nuevo paradigma para formular la búsqueda que ahora se llama*mantenimiento de la verdad* . Todos los sistemas de resolución de problemas modernos utilizan algún tipo de sistema de mantenimiento de la verdad como sustrato. VéaseForbus y deKleer 1993 para una discusión de formas elegantes de construir sistemas y aplicaciones de mantenimiento de la verdad utilizando el mantenimiento de la verdad.Zabih, McAllester y Chapman 1987 describen una extensión no determinista de Scheme que se basa en amb ; es similar al intérprete descrito en esta sección, pero más sofisticado, porque utiliza un retroceso dirigido por dependencia en lugar de un retroceso cronológico.retroceso. Winston 1992 ofrece una introducción a ambos tipos de retroceso.

Nuestro programa utiliza el siguiente procedimiento para determinar si los elementos de una lista son distintos [:](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_609)

(define (distinct? items)   
  (cond ((null? items) true)   
        ((null? (cdr items)) true)   
        ((member (car items) (cdr items)) false)   
        (else (distinct? (cdr items)))))

Member es como memq excepto que usa equal? ​​en lugar de eq? para probar la igualdad.

[49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_616) Esto está tomado de un folleto llamado "Recreaciones problemáticas", publicado en la década de 1960 por Litton Industries, donde se atribuye al*ingeniero estatal de Kansas*.

[50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_619) Aquí utilizamos la convención de que el primer elemento de cada lista designa la parte del discurso para el resto de las palabras de la lista.

[51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_620) Observe queparse-wordutilizaset!para modificar la lista de entrada no analizada. Para que esto funcione, nuestroambdebe deshacer los efectos deset!cuando retrocede.

[52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_621) Observe que esta definición es recursiva: un verbo puede ser seguido por cualquier número de frases preposicionales.

[53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_626) Este tipo de gramática puede llegar a ser arbitrariamente compleja, peroEn lo que respecta a la comprensión del lenguaje real, el análisis sintáctico es sólo un juguete. La comprensión real del lenguaje natural por computadora requiere una elaborada combinación de análisis sintáctico e interpretación del significado. Por otra parte, incluso los analizadores sintácticos de juguete pueden ser útiles para respaldar lenguajes de comandos flexibles para programas como los sistemas de recuperación de información.Winston 1992 analiza los enfoques computacionales para la comprensión del lenguaje real y también las aplicaciones de gramáticas simples para dominar los lenguajes.

[54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_628) Aunque la idea de Alyssa funciona bien (y es sorprendentemente simple), las oraciones que genera son un poco aburridas: no muestran las oraciones posibles de este lenguaje de una manera muy interesante. De hecho, la gramática es altamente recursiva en muchos lugares, y la técnica de Alyssa "cae" en una de estas recursiones y se queda atascada. Vea el ejercicio [4.50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.50)para encontrar una manera de lidiar con esto.

Elegimos implementar el evaluador perezoso en la sección [4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2)como una modificación del evaluador metacircular ordinario de la sección [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) [. Por el](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_629) contrario, basaremos elamben el evaluador analizador de la sección [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7), porque los procedimientos de ejecución en ese evaluador proporcionan un marco conveniente para implementar el retroceso.

[56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_632) Suponemos que el evaluador admitelet(ver ejercicio [4.22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.22)), que hemos utilizado en nuestros programas no deterministas.

[57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_636) No nos preocupamos por deshacer definiciones, ya que podemosSupongamos que se escanean las definiciones internas (sección  [4.1.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.6) ).

[**4.4 Programación lógica**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_4.4)

En el capítulo 1, hemos subrayado que la informática se ocupa del conocimiento imperativo (cómo hacer), mientras que las matemáticas se ocupan del conocimiento declarativo (qué es). De hecho, los lenguajes de programación exigen que el programador exprese el conocimiento de una forma que indique los métodos paso a paso para resolver problemas concretos. Por otra parte, los lenguajes de alto nivel proporcionan, como parte de la implementación del lenguaje, una cantidad sustancial de conocimiento metodológico que libera al usuario de la preocupación por numerosos detalles sobre cómo progresará un cálculo específico.

La mayoría de los lenguajes de programación, incluido Lisp, se organizan en torno al cálculo de los valores de funciones matemáticas. Los lenguajes orientados a expresiones (como Lisp, Fortran y Algol) aprovechan el "juego de palabras" de que una expresión que describe el valor de una función también puede interpretarse como un medio para calcular ese valor. Debido a esto, la mayoría de los lenguajes de programación están fuertemente sesgados hacia los cálculos unidireccionales (cálculos con entradas y salidas bien definidas). Sin embargo, hay lenguajes de programación radicalmente diferentes que relajan este sesgo. Vimos un ejemplo de este tipo en la sección  [3.3.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.5) , donde los objetos de cálculo eran restricciones aritméticas. En un sistema de restricciones, la dirección y el orden de cálculo no están tan bien especificados; al realizar un cálculo, el sistema debe, por lo tanto, proporcionar un conocimiento "cómo" más detallado que el que sería el caso de un cálculo aritmético ordinario. Sin embargo, esto no significa que el usuario esté completamente exento de la responsabilidad de proporcionar conocimiento imperativo. Hay muchas redes de restricciones que implementan el mismo conjunto de restricciones, y el usuario debe elegir del conjunto de redes matemáticamente equivalentes una red adecuada para especificar un cálculo particular.

El evaluador de programas no determinista de la sección  [4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3) también se aleja de la idea de que la programación consiste en construir algoritmos para calcular funciones unidireccionales. En un lenguaje no determinista, las expresiones pueden tener más de un valor y, como resultado, el cálculo esSe trata de relaciones en lugar de funciones de un solo valor. La programación lógica extiende esta idea combinando una visión relacional de la programación con un poderoso tipo de coincidencia de patrones simbólicos llamada *unificación* . [58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_645)

Este enfoque, cuando funciona, puede ser una forma muy poderosa de escribir programas. Parte de su poder proviene del hecho de que un único hecho de "qué es" se puede utilizar para resolver varios problemas diferentes que tendrían diferentes componentes de "cómo". Como ejemplo, considere eloperación append , que toma dos listas como argumentos y combina sus elementos para formar una sola lista. En un lenguaje procedimental como Lisp, podríamos definir append en términos del constructor de lista básico cons , como hicimos en la sección  [2.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.1) :

(define (añade x y)   
  (si (nulo? x)   
      y   
      (cons (car x) (añade (cdr x) y))))

Este procedimiento puede considerarse como una traducción a Lisp de las dos reglas siguientes, la primera de las cuales cubre el caso en que la primera lista está vacía y la segunda maneja el caso de una lista no vacía, que es un cons de dos partes:

* Para cualquier lista y , la lista vacía y y se agregan para formar y .
* Para cualquier u , v , y y z , (cons uv) e y se agregan para formar (cons u z) si v e y se agregan para formar z . [59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_646)

Usando el procedimiento de anexar , podemos responder preguntas como:

Encuentra el apéndice de (ab) y (cd) .

Pero las mismas dos reglas también son suficientes para responder los siguientes tipos de preguntas, que el procedimiento no puede responder:

Encuentra una lista y que agrega s con (ab) para producir (abcd) .

Encuentra todas las x e y que se agregan para formar (abcd) .

En un lenguaje de programación lógica, el programador escribe un procedimiento de anexión indicando las dos reglas sobre anexión dadas anteriormente. El intérprete proporciona automáticamente el conocimiento de cómo hacerlo para permitir que este único par de reglas se utilice para responder los tres tipos de preguntas sobre anexión . [60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_647)

Los lenguajes de programación lógica contemporáneos (incluido el que implementamos aquí) tienen deficiencias sustanciales, ya que sus métodos generales de "cómo hacer" pueden llevarlos a bucles infinitos espurios u otros comportamientos indeseables. La programación lógica es un campo activo de investigación en la ciencia informática. [61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_648)

Anteriormente en este capítulo exploramos la tecnología de implementación de intérpretes y describimos los elementos que son esenciales para un intérprete de un lenguaje similar a Lisp (de hecho, para un intérprete de cualquier lenguaje convencional). Ahora aplicaremos estas ideas para analizar un intérprete para un lenguaje de programación lógica. Lo llamamoslenguaje el *lenguaje de consulta* , porque es muy útil para recuperar información de bases de datos mediante la formulación*consultas* , o preguntas, expresadas en el lenguaje. Aunque el lenguaje de consultas es muy diferente de Lisp, nos resultará conveniente describir el lenguaje en términos del mismo marco general que hemos estado usando todo el tiempo: como una colección de elementos primitivos, junto con medios de combinación que nos permiten combinar elementos simples para crear elementos más complejos y medios de abstracción que nos permiten considerar elementos complejos como unidades conceptuales individuales. Un intérprete para un lenguaje de programación lógica es considerablemente más complejo que un intérprete para un lenguaje como Lisp. Sin embargo, veremosque nuestro intérprete de lenguaje de consulta contiene muchos de los mismos elementos que se encuentran en el intérprete de la sección  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) . En particular, habrá una parte ``eval'' que clasifica las expresiones según el tipo y una parte ``apply'' que implementa el mecanismo de abstracción del lenguaje (procedimientos en el caso de Lisp y *reglas* en el caso de la programación lógica). Además, una estructura de datos de marco juega un papel central en la implementación, que determina la correspondencia entre los símbolos y sus valores asociados. Un aspecto interesante adicional de nuestra implementación de lenguaje de consulta es que hacemos un uso sustancial de flujos, que se introdujeron en el capítulo 3.

[**4.4.1 Recuperación de información deductiva**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_4.4.1)

La programación lógica se destaca por brindar interfaces a bases de datos para la recuperación de información. El lenguaje de consulta que implementaremos en este capítulo está diseñado para usarse de esta manera.

Para ilustrar lo que hace el sistema de consulta, mostraremos cómo se puede utilizar para gestionar la base de datos de registros de personal paraMicroshaft, una próspera empresa de alta tecnología del área de Boston. El lenguaje proporciona acceso basado en patrones a la información del personal y también puede aprovechar las reglas generales para realizar deducciones lógicas.

**[Una base de datos de muestra](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_649)**

La base de datos de personal de Microshaft contiene *información* sobre el personal de la empresa. Aquí se incluye información sobre Ben Bitdiddle, el mago informático residente:

(dirección (Bitdiddle Ben) (Slumerville (Ridge Road) 10))   
(trabajo (Bitdiddle Ben) (mago de computadoras))   
(salario (Bitdiddle Ben) 60000)

Cada afirmación es una lista (en este caso un triple) cuyos elementos pueden ser en sí mismos listas.

Como mago residente, Ben está a cargo de la división informática de la empresa y supervisa a dos programadores y un técnico. Aquí se incluye información sobre ellos:

(dirección (Hacker Alyssa P) (Cambridge (Mass Ave) 78))   
(trabajo (Hacker Alyssa P) (programador de computadoras))   
(salario (Hacker Alyssa P) 40000)   
(supervisor (Hacker Alyssa P) (Bitdiddle Ben))   
(dirección (Fect Cy D) (Cambridge (Ames Street) 3))   
(trabajo (Fect Cy D) (programador de computadoras))   
(salario (Fect Cy D) 35000)   
(supervisor (Fect Cy D) (Bitdiddle Ben))   
(dirección (Tweakit Lem E) (Boston (Bay State Road) 22))   
(trabajo (Tweakit Lem E) (técnico en computadoras))   
(salario (Tweakit Lem E) 25000)   
(supervisor (Tweakit Lem E) (Bitdiddle Ben))

También hay un programador en prácticas, que está supervisado por Alyssa:

(dirección (Reasoner Louis) (Slumerville (Pine Tree Road) 80))   
(trabajo (Reasoner Louis) (aprendiz de programador informático))   
(salario (Reasoner Louis) 30000)   
(supervisor (Reasoner Louis) (Hacker Alyssa P))

Todas estas personas trabajan en la división de informática, como lo indica la palabra " computadora" como el primer elemento en sus descripciones de trabajo.

Ben es un empleado de alto nivel. Su supervisor es el pez gordo de la empresa:

(supervisor (Bitdiddle Ben) (Warbucks Oliver))   
(dirección (Warbucks Oliver) (Swellesley (Top Heap Road)))   
(trabajo (Warbucks Oliver) (noria administrativa))   
(salario (Warbucks Oliver) 150000)

Además de la división informática supervisada por Ben, la empresa cuenta con una división de contabilidad, formada por un jefe de contabilidad y su asistente:

(dirección (Scrooge Eben) (Weston (Shady Lane) 10))   
(trabajo (Scrooge Eben) (contador jefe))   
(salario (Scrooge Eben) 75000)   
(supervisor (Scrooge Eben) (Warbucks Oliver))   
(dirección (Cratchet Robert) (Allston (N Harvard Street) 16))   
(trabajo (Cratchet Robert) (escritor contable))   
(salario (Cratchet Robert) 18000)   
(supervisor (Cratchet Robert) (Scrooge Eben))

También hay una secretaria para la gran rueda:

(dirección (Aull DeWitt) (Slumerville (Onion Square) 5))   
(trabajo (Aull DeWitt) (secretaria administrativa))   
(salario (Aull DeWitt) 25000)   
(supervisor (Aull DeWitt) (Warbucks Oliver))

La base de datos también contiene afirmaciones sobre qué tipos de trabajos pueden realizar personas que tienen otros tipos de trabajos. Por ejemplo, un experto en informática puede realizar los trabajos de un programador y de un técnico informático:

(trabajo que puede hacer (mago de computadoras) (programador de computadoras))   
(trabajo que puede hacer (mago de computadoras) (técnico en computadoras))

Un programador informático podría sustituir a un aprendiz:

(trabajo que se puede hacer (programador informático)   
            (aprendiz de programador informático))

Además, como es bien sabido,

(trabajo que se puede hacer (secretaria administrativa)   
            (enorme figura administrativa))

**[Consultas sencillas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_650)**

El lenguaje de consulta permite a los usuarios recuperar información de la base de datos planteando consultas en respuesta a las indicaciones del sistema. Por ejemplo, para encontrar a todos los programadores de computadoras, se puede decir:

*;;; Entrada de consulta:*  
(trabajo ?x (programador de computadoras))

El sistema responderá con los siguientes elementos:

*;;; Resultados de la consulta:*  
(trabajo (Hacker Alyssa P) (programadora de computadoras))   
(trabajo (Fect Cy D) (programadora de computadoras))

La consulta de entrada especifica que estamos buscando entradas en la base de datos que coincidan con un *patrón* determinado . En este ejemplo, el patrón especifica entradas que constan de tres elementos, de los cuales el primero es el símbolo literal job , el segundo puede ser cualquier cosa y el tercero es la lista literal (programador informático) . El ``cualquier cosa'' que puede ser el segundo elemento de la lista coincidente se especifica mediante un*variable de patrón* , ?x . La forma general de una variable de patrón es un símbolo, que se toma como el nombre de la variable, precedido por un signo de interrogación. A continuación veremos por qué es útil especificar nombres para las variables de patrón en lugar de simplemente poner ? en los patrones para representar "cualquier cosa". El sistema responde a una consulta simple mostrando todas las entradas en la base de datos que coinciden con el patrón especificado.

Un patrón puede tener más de una variable. Por ejemplo, la consulta

(dirección ?x ?y)

Enumerará las direcciones de todos los empleados.

Un patrón no puede tener variables, en cuyo caso la consulta simplemente determina si ese patrón es una entrada en la base de datos. Si es así, habrá una coincidencia; si no, no habrá coincidencias.

La misma variable de patrón puede aparecer más de una vez en una consulta, lo que especifica que debe aparecer el mismo ``cualquier cosa'' en cada posición. Por eso las variables tienen nombres. Por ejemplo,

(supervisor ?x ?x)

Encuentra a todas las personas que se supervisan a sí mismas (aunque no existen tales afirmaciones en nuestra base de datos de muestra).

La consulta

(trabajo ?x (computadora ?tipo))

coincide con todas las entradas de trabajo cuyo tercer elemento es una lista de dos elementos cuyo primer elemento es computadora :

(trabajo (Bitdiddle Ben) (mago de computadoras))   
(trabajo (Hacker Alyssa P) (programadora de computadoras))   
(trabajo (Fect Cy D) (programadora de computadoras))   
(trabajo (Tweakit Lem E) (técnico en computadoras))

Este mismo patrón *no* coincide

(trabajo (Razonador Louis) (aprendiz de programador informático))

porque el tercer elemento de la entrada es una lista de tres elementos y el tercer elemento del patrón especifica que debe haber dos elementos. Si quisiéramos cambiar el patrón para que el tercer elemento pudiera ser cualquier lista que comience con computer , podríamos especificar[62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_651)

(trabajo ?x (computadora . ?tipo))

Por ejemplo,

(ordenador . ?tipo)

coincide con los datos

(aprendiz de programador informático)

con ?type como lista (programador en prácticas) . También coincide con los datos

(programador de computadoras)

con ?type como la lista (programador) , y coincide con los datos

(computadora)

con ?type como la lista vacía () .

Podemos describir el procesamiento de consultas simples del lenguaje de consulta de la siguiente manera:

* El sistema encuentra todas las asignaciones a variables en la consulta.patrón que *satisface* el patrón, es decir, todos los conjuntos de valores para las variables tales que si las variables del patrón son*instanciado con* (reemplazado por) los valores, el resultado está en la base de datos.
* El sistema responde a la consulta enumerando todas las instancias del patrón de consulta con las asignaciones de variables que lo satisfacen.

Tenga en cuenta que si el patrón no tiene variables, la consulta se reduce a determinar si ese patrón está en la base de datos. En caso afirmativo, la asignación vacía, que no asigna valores a las variables, satisface ese patrón para esa base de datos.

**Ejercicio 4.55.**   Realice consultas sencillas que recuperen la siguiente información de la base de datos:

a. todas las personas supervisadas por Ben Bitdiddle;

b. los nombres y cargos de todas las personas de la división de contabilidad;

c. los nombres y direcciones de todas las personas que viven en Slumerville.

**[Consultas compuestas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_653)**

Las consultas simples forman las operaciones primitivas del lenguaje de consultas. Para formar operaciones compuestas, el lenguaje de consultas proporciona medios de combinación. Una cosa que hace que el lenguaje de consultas sea un lenguaje de programación lógica es que los medios de combinación reflejan los medios de combinación utilizados para formar expresiones lógicas: and , or , and not . (Aquí and , or , and not no son las primitivas de Lisp, sino más bien operaciones integradas en el lenguaje de consultas).

Podemos utilizar y de la siguiente manera para encontrar las direcciones de todos los programadores de computadoras:

(y (trabajo ?persona (programador informático))   
     (dirección ?persona ?dónde))

La salida resultante es

(y (trabajo (Hacker Alyssa P) (programadora de computadoras))   
     (dirección (Hacker Alyssa P) (Cambridge (Mass Ave) 78)))   
(y (trabajo (Fect Cy D) (programadora de computadoras))   
     (dirección (Fect Cy D) (Cambridge (Ames Street) 3)))

En general,

(y < *consulta 1* > < *consulta 2* >  ... < *consulta n* >)

se satisface para todos los conjuntos de valores para las variables de patrón que satisfacen simultáneamente < *consulta 1* > ... < *consulta n* >.

En cuanto a las consultas simples, el sistema procesa una consulta compuesta encontrando todas las asignaciones a las variables de patrón que satisfacen la consulta y luego mostrando instancias de la consulta con esos valores.

Otro medio para construir consultas compuestas es mediante o . Por ejemplo,

(o (supervisor ?x (Bitdiddle Ben))   
    (supervisor ?x (Hacker Alyssa P)))

Encontrará a todos los empleados supervisados ​​por Ben Bitdiddle o Alyssa P. Hacker:

(o (supervisor (Hacker Alyssa P) (Bitdiddle Ben))   
    (supervisor (Hacker Alyssa P) (Hacker Alyssa P)))   
(o (supervisor (Fect Cy D) (Bitdiddle Ben))   
    (supervisor (Fect Cy D) (Hacker Alyssa P)))   
(o (supervisor (Tweakit Lem E) (Bitdiddle Ben))   
    (supervisor (Tweakit Lem E) (Hacker Alyssa P)))   
(o (supervisor (Reasoner Louis) (Bitdiddle Ben))   
    (supervisor (Reasoner Louis) (Hacker Alyssa P)))

En general,

(o < *consulta 1* > < *consulta 2* >  ... < *consulta n* >)

se satisface con todos los conjuntos de valores para las variables de patrón que satisfacen al menos uno de < *consulta 1* > ... < *consulta n* >.

También se pueden formar consultas compuestas con not . Por ejemplo,

(y (supervisor ?x (Bitdiddle Ben))   
     (no (trabajo ?x (programador de computadoras))))

Encuentra a todas las personas supervisadas por Ben Bitdiddle que no sean programadores informáticos. En general,

(no < *consulta 1* >)

se satisface con todas las asignaciones a las variables de patrón que no satisfacen < *consulta 1* >. [63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_654)

La forma de combinación final se llama lisp-value . Cuando lisp-value es el primer elemento de un patrón, especifica que el siguiente elemento es un predicado de Lisp que se aplicará al resto de los elementos (instanciados) como argumentos. En general,

(lisp-valor < *predicado* > < *arg 1* >  ... < *arg n* >)

se cumplirá con las asignaciones a las variables de patrón para las que el < *predicado* > aplicado a la instancia < *arg 1* > ... < *arg n* > es verdadero. Por ejemplo, para encontrar a todas las personas cuyo salario sea mayor a $30,000 podríamos escribir [64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_655)

(y (salario ?persona ?monto)   
     (lisp-value > ?monto 30000))

**Ejercicio 4.56.**   Formular consultas compuestas que recuperen la siguiente información:

a. los nombres de todas las personas supervisadas por Ben Bitdiddle, junto con sus direcciones;

b. todas las personas cuyo salario sea inferior al de Ben Bitdiddle, junto con su salario y el salario de Ben Bitdiddle;

c. todas las personas que estén supervisadas por alguien que no esté en la división de informática, junto con el nombre y el puesto del supervisor.

**[Normas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_657)**

Además de las consultas primitivas y las consultas compuestas, el lenguaje de consultas proporciona medios para abstraer consultas. Estos se dan mediante *reglas* . La regla

(regla (vive cerca de ?persona-1 ?persona-2)   
      (y (dirección ?persona-1 (?ciudad . ?resto-1))   
           (dirección ?persona-2 (?ciudad . ?resto-2))   
           (no (la misma ?persona-1 ?persona-2))))

especifica que dos personas viven cerca una de la otra si viven en la misma ciudad. La cláusula final no impide que la regla diga que todas las personas viven cerca de sí mismas. La misma relación se define mediante una regla muy simple: [65](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_658)

(regla (mismo ?x ?x))

La siguiente regla declara que una persona es una "rueda" en una organización si supervisa a alguien que a su vez es un supervisor:

(regla (rueda ?persona)   
      (y (supervisor ?gerente intermedio ?persona)   
           (supervisor ?x ?gerente intermedio)))

La forma general de una regla es

(regla < *conclusión* > < *cuerpo* >)

donde < *conclusión* > es un patrón y <cuerpo> *es* cualquier consulta. [66](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_659) Podemos pensar en una regla como la representación de un conjunto grande (incluso infinito) de afirmaciones, es decir, todas las instancias de la conclusión de la regla con asignaciones de variables que satisfacen el cuerpo de la regla. Cuando describimos consultas simples (patrones), dijimos que una asignación a variables satisface un patrón si el patrón instanciado está en la base de datos. Pero el patrón no necesita estar explícitamente en la base de datos como una afirmación.puede ser una afirmación implícita implícita en una regla. Por ejemplo, la consulta

(vive cerca de ?x (Bitdiddle Ben))

resultados en

(vive cerca de (Reasoner Louis) (Bitdiddle Ben))   
(vive cerca de (Aull DeWitt) (Bitdiddle Ben))

Para encontrar a todos los programadores de computadoras que viven cerca de Ben Bitdiddle, podemos preguntar

(y (trabajo ?x (programador de computadoras))   
     (vive cerca de ?x (Bitdiddle Ben)))

Al igual que en el caso de los procedimientos compuestos, las reglas se pueden utilizar como partes de otras reglas (como vimos con la regla de vidas cercanas más arriba) o incluso definirse recursivamente. Por ejemplo, la regla

(regla (superado en rango por ?personal ?jefe)   
      (o (supervisor ?personal ?jefe)   
          (y (supervisor ?personal ?gerente intermedio)   
               (superado en rango por ?gerente intermedio ?jefe))))

dice que un miembro del personal tiene un rango superior al de un jefe en la organización si el jefe es el supervisor de la persona o (recursivamente) si el supervisor de la persona tiene un rango superior al del jefe.

**Ejercicio 4.57.**   Defina una regla que diga que la persona 1 puede reemplazar a la persona 2 si la persona 1 hace el mismo trabajo que la persona 2 o si alguien que hace el trabajo de la persona 1 también puede hacer el trabajo de la persona 2, y si la persona 1 y la persona 2 no son la misma persona. Usando su regla, proporcione consultas que encuentren lo siguiente:

a. todas las personas que puedan sustituir a Cy D. Fect;

b. todas las personas que puedan sustituir a alguien que recibe un salario superior al suyo, junto con los dos salarios.

**Ejercicio 4.58.**   Defina una regla que diga que una persona es un “pez gordo” en una división si trabaja en la división pero no tiene un supervisor que trabaje en la división.

**Ejercicio 4.59.**   Ben Bitdiddle ha faltado a una reunión de más. Temiendo que su hábito de olvidar reuniones pueda costarle el trabajo, Ben decide hacer algo al respecto. Añade todas las reuniones semanales de la empresa a la base de datos de Microshaft afirmando lo siguiente:

(contabilidad de la reunión (lunes 9 a. m.))   
(administración de la reunión (lunes 10 a. m.))   
(ordenador de la reunión (miércoles 3 p. m.))   
(administración de la reunión (viernes 1 p. m.))

Cada una de las afirmaciones anteriores corresponde a una reunión de una división entera. Ben también agrega una entrada para la reunión de toda la empresa que abarca todas las divisiones. Todos los empleados de la empresa asisten a esta reunión.

(reunión de toda la empresa (miércoles 16 h))

a. El viernes por la mañana, Ben quiere consultar la base de datos para conocer todas las reuniones que se realizarán ese día. ¿Qué consulta debería utilizar?

b. A Alyssa P. Hacker no le convence. Cree que sería mucho más útil poder solicitar sus reuniones especificando su nombre. Por eso diseña una regla que dice que las reuniones de una persona incluyen todas las reuniones de toda la empresa más todas las reuniones de la división de esa persona. Complete el cuerpo de la regla de Alyssa.

(regla (hora de la reunión ?persona ?día y hora)   
      < *cuerpo de la regla* >)

c. Alyssa llega al trabajo el miércoles por la mañana y se pregunta a qué reuniones debe asistir ese día. Una vez definida la regla anterior, ¿qué consulta debería realizar para averiguarlo?

**Ejercicio 4.60.**  Al dar la consulta

(vive cerca de ?persona (Hacker Alyssa P))

Alyssa P. Hacker es capaz de encontrar personas que viven cerca de ella, con las que puede viajar al trabajo. Por otro lado, cuando intenta encontrar todos los pares de personas que viven cerca entre sí consultando

(vive cerca de ?persona-1 ?persona-2)

Ella observa que cada par de personas que viven cerca una de la otra aparece enumerado dos veces; por ejemplo,

(vive cerca de (Hacker Alyssa P) (Fect Cy D))   
(vive cerca de (Fect Cy D) (Hacker Alyssa P))

¿Por qué ocurre esto? ¿Hay alguna manera de encontrar una lista de personas que viven cerca unas de otras, en la que cada par aparezca solo una vez? Explíquelo.

**[La lógica como programas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_664)**

Podemos considerar una regla como una especie de implicación lógica: *si* una asignación de valores a las variables de patrón satisface el cuerpo, *entonces* satisface la conclusión. En consecuencia, podemos considerar que el lenguaje de consulta tiene la capacidad de realizar *deducciones lógicas* basadas en las reglas. Como ejemplo, considere la operación de anexión descrita al comienzo de la sección  [4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4) . Como dijimos, la operación de anexión se puede caracterizar por las siguientes dos reglas:

* Para cualquier lista y , la lista vacía y y se agregan para formar y .
* Para cualquier u , v , y y z , (cons uv) e y se añaden para formar (cons u z) si v e y se añaden para formar z .

Para expresar esto en nuestro lenguaje de consulta, definimos dos reglas para una relación

(agregar al formulario x y z)

Lo cual podemos interpretar como " x e y se añaden para formar z ":

(regla (añadir-a-forma () ?y ?y))   
(regla (añadir-a-forma (?u . ?v) ?y (?u . ?z))   
      (añadir-a-forma ?v ?y ?z))

La primera regla no tiene cuerpo, lo que significa que la conclusión es válida para cualquier valor de ?y . Observe cómo la segunda regla hace uso deNotación de cola punteada para nombrar el automóvil y el CDR de una lista.

Dadas estas dos reglas, podemos formular consultas que calculen la adición de dos listas:

*;;; Entrada de consulta:*  
(append-to-form (a b) (c d) ?z)   
*;;; Resultados de la consulta:*  
(append-to-form (a b) (c d) (a b c d))

Lo que es más sorprendente es que podemos usar las mismas reglas para preguntarnos: "¿Qué lista, cuando se añade a (ab) , produce (abcd) ? " Esto se hace de la siguiente manera:

*;;; Entrada de consulta:*  
(append-to-form (a b) ?y (a b c d))   
*;;; Resultados de la consulta:*  
(append-to-form (a b) (c d) (a b c d))

También podemos pedir todos los pares de listas que se agregan para formar (abcd) :

*;;; Entrada de consulta:*  
(append-to-form ?x ?y (a b c d))   
*;;; Resultados de la consulta:*  
(append-to-form () (a b c d) (a b c d))   
(append-to-form (a) (b c d) (a b c d))   
(append-to-form (a b) (c d) (a b c d))   
(append-to-form (a b c) (d) (a b c d))   
(append-to-form (a b c d) () (a b c d))

El sistema de consultas puede parecer bastante inteligente al utilizar las reglas para deducir las respuestas a las consultas anteriores. En realidad, como veremos en la siguiente sección, el sistema sigue un algoritmo bien determinado para desentrañar las reglas. Desafortunadamente, aunque el sistema funciona de manera impresionante en el caso de la anexión , los métodos generales pueden fallar en casos más complejos, como veremos en la sección  [4.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.3) .

**Ejercicio 4.61.**   Las siguientes reglas implementan una relación de "próximo a" que encuentra elementos adyacentes de una lista:

(regla (?x junto a ?y en (?x ?y . ?u)))   
  
(regla (?x junto a ?y en (?v . ?z))   
      (?x junto a ?y en ?z))

¿Cuál será la respuesta a las siguientes preguntas?

(?x junto a ?y en (1 (2 3) 4))   
  
(?x junto a 1 en (2 1 3 1))

**Ejercicio 4.62.**  Defina reglas para implementar la operación de último par del ejercicio  [2.17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.17) , que devuelve una lista que contiene el último elemento de una lista no vacía. Verifique sus reglas en consultas como (last-pair (3) ?x) , (last-pair (1 2 3) ?x) y (last-pair (2 ?x) (3)) . ¿Funcionan correctamente sus reglas en consultas como (last-pair ?x (3)) ?

**Ejercicio 4.63.**  La siguiente base de datos (ver Génesis 4) rastrea la genealogía de los descendientes de Ada hasta Adán, pasando por Caín:

(hijo Adam Caín)   
(hijo Caín Enoc)   
(hijo Enoc Irad)   
(hijo Irad Mehujael)   
(hijo Mehujael Methushael)   
(hijo Methushael Lamech)   
(esposa Lamech Ada)   
(hijo Ada Jabal)   
(hijo Ada Jubal)

Formular reglas como ``Si *S* es el hijo de *F* , y *F* es el hijo de *G* , entonces *S* es el nieto de *G* '' y ``Si *W* es la esposa de *M* , y *S* es el hijo de *W* , entonces *S* es el hijo de *M* '' (lo que supuestamente era más cierto en tiempos bíblicos que en la actualidad) que permitirán al sistema de consulta encontrar al nieto de Caín; los hijos de Lamec; los nietos de Metusael. (Véase el ejercicio  [4.69](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.69) para algunas reglas para deducir relaciones más complicadas.)

**[4.4.2 Cómo funciona el sistema de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.2)**

En la sección  [4.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4) presentaremos una implementación del intérprete de consultas como una colección de procedimientos. En esta sección daremos una visión general que explica la estructura general del sistema independientemente de los detalles de implementación de bajo nivel. Después de describir la implementación del intérprete, estaremos en condiciones de entender algunas de sus limitaciones y algunas de las formas sutiles en las que las operaciones lógicas del lenguaje de consultas difieren de las operaciones de la lógica matemática.

Debería ser evidente que el evaluador de consultas debe realizar algún tipo de búsqueda para hacer coincidir las consultas con los hechos y las reglas de la base de datos. Una forma de hacerlo sería implementar el sistema de consultas como un programa no determinista, utilizando el evaluador amb de la sección  [4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3) (ver ejercicio  [4.78](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.78) ). Otra posibilidad es gestionar la búsqueda con la ayuda de flujos. Nuestra implementación sigue este segundo enfoque.

El sistema de consultas está organizado en torno a dos operaciones centrales llamadas *coincidencia de patrones* y *unificación* . En primer lugar, describimos la coincidencia de patrones y explicamos cómo esta operación, junto con la organización de la información en términos de flujos de marcos, nos permite implementar consultas simples y compuestas. A continuación, analizamos la unificación, una generalización de la coincidencia de patrones necesaria para implementar reglas. Por último, mostramos cómo todo el intérprete de consultas se integra mediante un procedimiento que clasifica las expresiones de una manera análoga a la forma en que eval clasifica las expresiones para el intérprete descrito en la sección  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) .

**[Coincidencia de patrones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_668)**

Un *comparador de patrones* es un programa que prueba si algún dato se ajusta a un patrón especificado. Por ejemplo, la lista de datos ((ab) c (ab)) coincide con el patrón (?xc ?x) con la variable de patrón ?x asociada a (ab) . La misma lista de datos coincide con el patrón (?x ?y ?z) con ?x y ?z asociadas a (ab) y ?y asociada a c . También coincide con el patrón ((?x ?y) c (?x ?y)) con ?x asociada a a y ?y asociada a b . Sin embargo, no coincide con el patrón (?xa ?y) , ya que ese patrón especifica una lista cuyo segundo elemento es el símbolo a .

El comparador de patrones que utiliza el sistema de consulta toma como entradas un patrón, un dato y un *marco* que especifica enlaces para varias variables de patrón. Comprueba si el dato coincide con el patrón de una manera que sea coherente con los enlaces que ya existen en el marco. Si es así, devuelve el marco dado ampliado con cualquier enlace que pueda haberse determinado mediante la coincidencia. De lo contrario, indica que la coincidencia ha fallado.

Por ejemplo, si se utiliza el patrón (?x ?y ?x) para hacer coincidir (aba) con un marco vacío, se obtendrá un marco que especifica que ?x está vinculado a a y ?y está vinculado a b . Si se intenta hacer la coincidencia con el mismo patrón, el mismo dato y un marco que especifica que ?y está vinculado a a , se producirá un error. Si se intenta hacer la coincidencia con el mismo patrón, el mismo dato y un marco en el que ?y está vinculado a b y ?x no está vinculado, se obtendrá el marco dado aumentado por un vínculo de ?x a a .

El comparador de patrones es todo el mecanismo que se necesita para procesar consultas simples que no involucran reglas. Por ejemplo, para procesar la consulta

(¿trabajo ?x (programador de computadoras))

Analizamos todas las afirmaciones de la base de datos y seleccionamos aquellas que coinciden con el patrón con respecto a un marco inicialmente vacío. Para cada coincidencia que encontramos, utilizamos el marco devuelto por la coincidencia para crear una instancia del patrón con un valor para ?x .

[**Flujos de fotogramas**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_669)

La prueba de patrones contra marcos se organiza mediante el uso de flujos. Dado un solo marco, el proceso de comparación recorre las entradas de la base de datos una por una. Para cada entrada de la base de datos, el comparador genera un símbolo especial que indica que la comparación ha fallado o una extensión del marco. Los resultados de todas las entradas de la base de datos se recopilan en un flujo, que se pasa a través de un filtro para eliminar los errores. El resultado es un flujo de todos los marcos que extienden el marco dado a través de una coincidencia con alguna afirmación en la base de datos. [67](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_670)

En nuestro sistema, una consulta toma un flujo de entrada de fotogramas y realiza la operación de coincidencia anterior para cada fotograma del flujo, como se indica en la figura  [4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.4) . Es decir, para cada fotograma del flujo de entrada, la consulta genera un nuevo flujo que consta de todas las extensiones de ese fotograma mediante coincidencias con las afirmaciones de la base de datos. Todos estos flujos se combinan para formar un flujo enorme, que contiene todas las extensiones posibles de cada fotograma del flujo de entrada. Este flujo es la salida de la consulta.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 4.4:**   Una consulta procesa un flujo de fotogramas. |

Para responder a una consulta sencilla, utilizamos la consulta con un flujo de entrada que consta de un único marco vacío. El flujo de salida resultante contiene todas las extensiones del marco vacío (es decir, todas las respuestas a nuestra consulta). Este flujo de marcos se utiliza luego para generar un flujo de copias del patrón de consulta original con las variables instanciadas por los valores de cada marco, y este es el flujo que finalmente se imprime.

**[Consultas compuestas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_671)**

La verdadera elegancia de la implementación del flujo de cuadros es evidente cuando tratamos con consultas compuestas. El procesamiento de consultas compuestas hace uso de la capacidad de nuestro comparador para exigir que se realice una coincidenciaser coherente con un marco especificado. Por ejemplo, para manejar el y de dos consultas, como

(y (¿puede hacer el trabajo? x (aprendiz de programador informático))   
     (¿trabajo? persona? x))

(informalmente, ``Encuentra a todas las personas que puedan hacer el trabajo de un aprendiz de programador de computadoras''), primero encontramos todas las entradas que coinciden con el patrón

(¿Puedo hacer un trabajo? x (aprendiz de programador informático))

Esto produce un flujo de cuadros, cada uno de los cuales contiene un enlace para ?x . Luego, para cada cuadro del flujo, encontramos todas las entradas que coinciden.

(trabajo ?persona ?x)

de una manera que sea consistente con el enlace dado para ?x . Cada una de estas coincidencias producirá un marco que contiene enlaces para ?x y ?person . El y de dos consultas se puede ver como una combinación en serie de las dos consultas componentes, como se muestra en la figura  [4.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.5) . Los marcos que pasan por el primer filtro de consulta se filtran y se extienden aún más mediante la segunda consulta.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 4.5:**   La combinación de dos consultas se produce operando sobre el flujo de tramas en serie. |

La figura  [4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.6) muestra el método análogo para calcular la función or de dos consultas como una combinación paralela de las dos consultas componentes. El flujo de entrada de tramas se extiende por separado en cada consulta. Los dos flujos resultantes se fusionan para producir el flujo de salida final.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 4.6:**   La consulta o combinación de dos consultas se produce operando sobre el flujo de tramas en paralelo y fusionando los resultados. |

Incluso a partir de esta descripción de alto nivel, es evidente que el procesamiento de consultas compuestas puede ser lento. Por ejemplo, dado que una consulta puede producir más de un marco de salida para cada marco de entrada, y cada consulta en un y obtiene sus marcos de entrada de la consulta anterior, una consulta y podría, en el peor de los casos, tener que realizar una cantidad de coincidencias que sea exponencial en la cantidad de consultas (ver ejercicio  [4.76](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.76) ). [68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_672) Aunque los sistemas para manejar solo consultas simples son bastante prácticos, manejar consultas complejas es extremadamente difícil. [69](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_673)

Desde el punto de vista del flujo de cuadros, el no de alguna consulta actúa como un filtro que elimina todos los cuadros para los que se puede satisfacer la consulta. Por ejemplo, dado el patrón

(no (trabajo ?x (programador de computadoras)))

Intentamos, para cada cuadro en el flujo de entrada, producir cuadros de extensión que satisfagan (trabajo ?x (programador de computadoras)) . Eliminamos del flujo de entrada todos los cuadros para los que existen dichas extensiones. El resultado es un flujo que consta solo de aquellos cuadros en los que el enlace para ?x no satisface (trabajo ?x (programador de computadoras)) . Por ejemplo, al procesar la consulta

(y (supervisor ?x ?y)   
     (no (trabajo ?x (programador de computadoras))))

La primera cláusula generará marcos con enlaces para ?x y ?y . La cláusula not los filtrará eliminando todos los marcos en los que el enlace para ?x satisface la restricción de que ?x es un programador de computadoras. [70](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_674)

La forma especial de valor Lisp se implementa como un filtro similar en los flujos de cuadros. Usamos cada cuadro del flujo para instanciar cualquier variable en el patrón y luego aplicamos el predicado Lisp. Eliminamos del flujo de entrada todos los cuadros para los que falla el predicado.

[**Unificación**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_675)

Para manejar reglas en el lenguaje de consulta, debemos ser capaces de encontrar las reglas cuyas conclusiones coincidan con un patrón de consulta determinado. Las conclusiones de las reglas son como las afirmaciones, excepto que pueden contener variables, por lo que necesitaremos una generalización de la coincidencia de patrones (denominada *unificación* ) en la que tanto el «patrón» como el «dato» puedan contener variables.

Un unificador toma dos patrones, cada uno de los cuales contiene constantes y variables, y determina si es posible asignar valores a las variables que harán que los dos patrones sean iguales. Si es así, devuelve un marco que contiene estas vinculaciones. Por ejemplo, unificar (?xa ?y) y (?y ?za) especificará un marco en el que ?x , ? y y ?z deben estar vinculados a a . Por otro lado, unificar (?x ?ya) y (?xb ?y) fallará, porque no hay ningún valor para ?y que pueda hacer que los dos patrones sean iguales. (Para que los segundos elementos de los patrones sean iguales, ?y tendría que ser b ; sin embargo, para que los terceros elementos sean iguales, ?y tendría que ser a ). El unificador utilizado en el sistema de consulta, al igual que el comparador de patrones, toma un marco como entrada y realiza unificaciones que son consistentes con este marco.

El algoritmo de unificación es la parte técnicamente más difícil del sistema de consulta. Con patrones complejos, realizar la unificación puede parecer que requiere deducción. Para unificar (?x ?x) y ((a ?yc) (ab ?z)) , por ejemplo, el algoritmo debe inferir que ?x debe ser (abc) , ?y debe ser b y ?z debe ser c . Podemos pensar en este proceso como la solución de un conjunto de ecuaciones entre los componentes del patrón. En general, se trata de ecuaciones simultáneas, que pueden requerir una manipulación sustancial para resolverlas. [71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_676) Por ejemplo, la unificación de (?x ?x) y ((a ?yc) (ab ?z)) puede considerarse como la especificación de las ecuaciones simultáneas

?x = (a ?y c)   
?x = (a b ?z)

Estas ecuaciones implican que

(a?yc) = (a?b?z)

lo que a su vez implica que

a = a, ?y = b, c = ?z,

y por lo tanto que

?x = (a b c)

En una coincidencia de patrones exitosa, todas las variables de patrón se vinculan y los valores a los que están vinculadas contienen solo constantes. Esto también es cierto para todos los ejemplos de unificación que hemos visto hasta ahora. Sin embargo, en general, una unificación exitosa puede no determinar por completo los valores de las variables; algunas variables pueden permanecer sin vincular y otras pueden estar vinculadas a valores que contienen variables.

Consideremos la unificación de (?xa) y ((b ?y) ?z) . Podemos deducir que ?x = (b ?y) y a = ?z , pero no podemos resolver más para ?x o  ?y . La unificación no falla, ya que es ciertamente posible hacer que los dos patrones sean iguales asignando valores a ?x y ?y . Dado que esta coincidencia de ninguna manera restringe los valores que ?y puede tomar, no se coloca ningún enlace para ?y en el marco de resultados. La coincidencia, sin embargo, restringe el valor de  ?x . Cualquiera que sea el valor que tenga ?y , ?x debe ser (b ?y) . Por lo tanto, se coloca un enlace de ?x al patrón (b ?y) en el marco. Si más tarde se determina un valor para ?y y se agrega al marco (por una coincidencia de patrón o unificación que se requiere que sea consistente con este marco), el enlace previo ?x se referirá a este valor. [72](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_677)

[**Aplicación de reglas**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_678)

La unificación es la clave del componente del sistema de consultas que realiza inferencias a partir de reglas. Para ver cómo se logra esto, considere procesar una consulta que implica aplicar una regla, como

(vive cerca de ?x (Hacker Alyssa P))

Para procesar esta consulta, primero utilizamos el procedimiento de comparación de patrones habitual descrito anteriormente para ver si hay alguna afirmación en la base de datos que coincida con este patrón. (No habrá ninguna en este caso, ya que nuestra base de datos no incluye ninguna afirmación directa sobre quién vive cerca de quién). El siguiente paso es intentar unificar el patrón de consulta con la conclusión de cada regla. Descubrimos que el patrón se unifica con la conclusión de la regla.

(regla (vive cerca de ?persona-1 ?persona-2)   
      (y (dirección ?persona-1 (?ciudad . ?resto-1))   
           (dirección ?persona-2 (?ciudad . ?resto-2))   
           (no (la misma ?persona-1 ?persona-2))))

lo que da como resultado un marco que especifica que ?person-2 está vinculado a (Hacker Alyssa P) y que ?x debería estar vinculado a (tener el mismo valor que) ?person-1 . Ahora, en relación con este marco, evaluamos la consulta compuesta dada por el cuerpo de la regla. Las coincidencias exitosas extenderán este marco al proporcionar un vínculo para ?person-1 y, en consecuencia, un valor para ?x , que podemos usar para instanciar el patrón de consulta original.

En general, el evaluador de consultas utiliza el siguiente método para aplicar una regla cuando intenta establecer un patrón de consulta en un marco que especifica enlaces para algunas de las variables del patrón:

* Unificar la consulta con la conclusión de la regla para formar, si tiene éxito, una extensión del marco original.
* En relación al marco extendido, evalúa la consulta formada por el cuerpo de la regla.

Observe cuán similar es esto al método para aplicar un procedimiento en el evaluador eval / apply para Lisp:

* Vincula los parámetros del procedimiento a sus argumentos para formar un marco que extienda el entorno del procedimiento original.
* En relación al entorno extendido, evalúe la expresión formada por el cuerpo del procedimiento.

La similitud entre los dos evaluadores no debería sorprender a nadie. Así como las definiciones de procedimientos son los medios de abstracción en Lisp, las definiciones de reglas son los medios de abstracción en el lenguaje de consulta. En cada caso, desenrollamos la abstracción creando enlaces apropiados y evaluando el cuerpo de la regla o el procedimiento en relación con estos.

**[Consultas sencillas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_679)**

En esta sección, vimos cómo evaluar consultas simples en ausencia de reglas. Ahora que hemos visto cómo aplicar reglas, podemos describir cómo evaluar consultas simples utilizando tanto reglas como afirmaciones.

Dado el patrón de consulta y un flujo de fotogramas, producimos, para cada fotograma del flujo de entrada, dos flujos:

* un flujo de marcos extendidos obtenidos al comparar el patrón con todas las afirmaciones en la base de datos (usando el comparador de patrones), y
* un flujo de cuadros extendidos obtenidos al aplicar todas las reglas posibles (utilizando el unificador). [73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_680)

La adición de estos dos flujos produce un flujo que consta de todas las formas en que se puede satisfacer el patrón dado de manera coherente con el marco original. Estos flujos (uno para cada marco del flujo de entrada) ahora se combinan para formar un flujo grande, que por lo tanto consta de todas las formas en que cualquiera de los marcos del flujo de entrada original se puede extender para producir una coincidencia con el patrón dado.

[**El evaluador de consultas y el bucle del controlador**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_681)

A pesar de la complejidad de las operaciones de comparación subyacentes, el sistema está organizado de forma muy similar a un evaluador de cualquier idioma. El procedimiento que coordina las operaciones de comparación se denominaqeval , y desempeña un papel análogo al del procedimiento eval para Lisp. Qeval toma como entradas una consulta y un flujo de marcos. Su salida es un flujo de marcos, correspondientes a coincidencias exitosas con el patrón de consulta, que extienden algún marco en el flujo de entrada, como se indica en la figura  [4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.4) . Al igual que eval , qeval clasifica los diferentes tipos de expresiones (consultas) y envía a un procedimiento apropiado para cada una. Hay un procedimiento para cada forma especial ( and , or , not y lisp-value ) y uno para consultas simples.

El bucle de controlador, que es análogo al procedimiento de bucle de controlador para los demás evaluadores de este capítulo, lee consultas desde la terminal. Para cada consulta, llama a qeval con la consulta y un flujo que consta de un único marco vacío. Esto producirá el flujo de todas las posibles coincidencias (todas las posibles extensiones del marco vacío). Para cada marco del flujo resultante, crea una instancia de la consulta original utilizando los valores de las variables que se encuentran en el marco. A continuación, se imprime este flujo de consultas instanciadas. [74](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_682)

El controlador también comprueba el comando especial assert!, que indica que la entrada no es una consulta sino una afirmación o regla que se agregará a la base de datos. Por ejemplo,

(¡afirma! (trabajo (Bitdiddle Ben) (mago de la computadora)))   
(¡afirma! (regla (rueda ?persona)   
               (y (supervisor ?gerente intermedio ?persona)   
                    (supervisor ?x ?gerente intermedio))))

**[4.4.3 ¿La programación lógica es lógica matemática?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.3)**

Los medios de combinación utilizados en el lenguaje de consulta pueden parecer a primera vista idénticos a las operaciones y , o , y no de lógica matemática, y la aplicación de las reglas del lenguaje de consulta se logra de hecho a través de un método legítimo deinferencia. [75](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_683) Esta identificación del lenguaje de consulta con la lógica matemática no es realmente válida, sin embargo, porque el lenguaje de consulta proporciona una*Estructura de control* que interpreta las declaraciones lógicas de manera procedimental. A menudo podemos aprovechar esta estructura de control. Por ejemplo, para encontrar todos los supervisores de los programadores, podríamos formular una consulta en cualquiera de dos formas lógicamente equivalentes:

(y (trabajo ?x (programador de computadoras))   
     (supervisor ?x ?y))

o

(y (supervisor ?x ?y)   
     (trabajo ?x (programador de computadoras)))

Si una empresa tiene muchos más supervisores que programadores (el caso habitual), es mejor utilizar la primera forma en lugar de la segunda porque se debe escanear la base de datos en busca de cada resultado intermedio (marco) producido por la primera cláusula de y .

El objetivo de la programación lógica es proporcionar al programador técnicas para descomponer un problema computacional en dos problemas separados: «qué» se debe calcular y «cómo» se debe calcular. Esto se logra seleccionando un subconjunto de los enunciados de la lógica matemática que sea lo suficientemente potente como para poder describir cualquier cosa que uno quiera calcular, pero lo suficientemente débil como para tener una interpretación procedimental controlable. La intención aquí es que, por un lado, un programa especificado en un lenguaje de programación lógica sea un programa efectivo que pueda ser llevado a cabo por una computadora. El control («cómo» calcular) se efectúa utilizando el orden de evaluación del lenguaje. Deberíamos poder organizar el orden de las cláusulas y el orden de los subobjetivos dentro de cada cláusula de modo que el cálculo se realice en un orden que se considere efectivo y eficiente. Al mismo tiempo, deberíamos poder ver el resultado del cálculo («qué» calcular) como una simple consecuencia de las leyes de la lógica.

Nuestro lenguaje de consulta puede considerarse como un subconjunto procedimentalmente interpretable de la lógica matemática. Una afirmación representa un hecho simple (una proposición atómica). Una regla representa la implicación de que la conclusión de la regla es válida para aquellos casos en que se cumple el cuerpo de la regla. Una regla tiene una interpretación procedimental natural: para establecer la conclusión de la regla, se establece el cuerpo de la regla. Las reglas, por lo tanto, especifican cálculos. Sin embargo, dado que las reglas también pueden considerarse enunciados de la lógica matemática, podemos justificar cualquier «inferencia» realizada por un programa lógico afirmando que se podría obtener el mismo resultado trabajando completamente dentro de la lógica matemática. [76](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_684)

**[Bucles infinitos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_685)**

Una consecuencia de la interpretación procedimental de los programas lógicos es que es posible construir programas irremediablemente ineficientes para resolver ciertos problemas. Un caso extremo de ineficiencia ocurre cuando el sistema cae en bucles infinitos al hacer deducciones. Como ejemplo simple, supongamos que estamos creando una base de datos de matrimonios famosos, incluidos

(¡afirma! (se casó con Minnie Mickey))

Si ahora preguntamos

(¿se casó con Mickey? ¿quién?)

No obtendremos ninguna respuesta, porque el sistema no sabe que si *A* está casado con *B* , entonces *B* está casado con *A.* Por lo tanto, afirmamos la regla

(afirma! (regla (casado ?x ?y)   
               (casado ?y ?x)))

y otra vez consulta

(¿se casó con Mickey? ¿quién?)

Desafortunadamente, esto llevará al sistema a un bucle infinito, como sigue:

* El sistema determina que la regla casada es aplicable; es decir, la conclusión de la regla (casada ?x ?y) se unifica correctamente con el patrón de consulta (casada Mickey ?who) para producir un marco en el que ?x está ligado a Mickey y ?y está ligado a ?who . Por lo tanto, el intérprete procede a evaluar el cuerpo de la regla (casada ?y ?x) en este marco; en efecto, a procesar la consulta (casada ?who Mickey) .
* Una respuesta aparece directamente como afirmación en la base de datos: (se casó con Minnie Mickey) .
* La regla de casado también es aplicable, por lo que el intérprete evalúa nuevamente el cuerpo de la regla, que esta vez es equivalente a (Mickey casado ?quién) .

El sistema se encuentra ahora en un bucle infinito. De hecho, el hecho de que el sistema encuentre la respuesta simple (se casó con Minnie Mickey) antes de entrar en el bucle depende de los detalles de implementación relacionados con el orden en el que el sistema comprueba los elementos en la base de datos. Este es un ejemplo muy simple de los tipos de bucles que pueden ocurrir. Las colecciones de reglas interrelacionadas pueden dar lugar a bucles que son mucho más difíciles de anticipar, y la aparición de un bucle puede depender del orden de las cláusulas en un and (véase el ejercicio  [4.64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.64) ) o de detalles de bajo nivel relacionados con el orden en el que el sistema procesa las consultas. [77](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_686)

[**Problemas con no**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_687)

Otra peculiaridad del sistema de consultas se refiere a no . Dada la base de datos de la sección  [4.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.1) , considere las dos consultas siguientes:

(y (supervisor ?x ?y)   
     (no (trabajo ?x (programador de computadoras))))   
(y (no (trabajo ?x (programador de computadoras)))   
     (supervisor ?x ?y))

Estas dos consultas no producen el mismo resultado. La primera consulta comienza buscando todas las entradas en la base de datos que coinciden con (supervisor ?x ?y) , y luego filtra los marcos resultantes eliminando aquellos en los que el valor de ?x satisface (job ?x (computer programmer)) . La segunda consulta comienza filtrando los marcos entrantes para eliminar aquellos que pueden satisfacer (job ?x (computer programmer)) . Dado que el único marco entrante está vacío, verifica la base de datos para ver si hay algún patrón que satisfaga (job ?x (computer programmer)) . Dado que generalmente hay entradas de este formato, la cláusula not filtra el marco vacío y devuelve un flujo vacío de marcos. En consecuencia, la consulta compuesta completa devuelve un flujo vacío.

El problema es que nuestra implementación de not en realidad está pensada para servir como filtro de valores para las variables. Si se procesa una cláusula not con un marco en el que algunas de las variables permanecen sin vincular (como ocurre con ?x en el ejemplo anterior), el sistema producirá resultados inesperados. Problemas similares ocurren con el uso delisp-value : el predicado de Lisp no puede funcionar si algunos de sus argumentos no están ligados. Consulte el ejercicio  [4.77](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.77) .

También hay una forma mucho más seria en la que el no del lenguaje de consulta difiere del no de la lógica matemática. En lógica, interpretamos la afirmación "no *P* " como que *P* no es verdadera. Sin embargo, en el sistema de consulta, "no *P* " significa que *P* no es deducible del conocimiento en la base de datos. Por ejemplo, dada la base de datos de personal de la sección  [4.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.1) , el sistema deduciría felizmente todo tipo de afirmaciones no , como que Ben Bitdiddle no es un aficionado al béisbol, que no está lloviendo afuera y que 2 + 2 no es 4. [78](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_688) En otras palabras, el no de los lenguajes de programación lógica refleja la llamada*Supuesto de mundo cerrado* en el que toda la información relevante se ha incluido en la base de datos. [79](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_689)

**Ejercicio 4.64.**  Louis Reasoner elimina por error la regla de superación por (sección  [4.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.1) ) de la base de datos. Cuando se da cuenta de esto, la reinstala rápidamente. Desafortunadamente, hace un pequeño cambio en la regla y la escribe como

(regla (superior en rango a ?miembro del personal ?jefe)   
      (o (supervisor ?miembro del personal ?jefe)   
          (y (superior en rango a ?gerente intermedio ?jefe)   
               (supervisor ?miembro del personal ?gerente intermedio))))

Justo después de que Louis introduce esta información en el sistema, DeWitt Aull aparece para averiguar quién supera en rango a Ben Bitdiddle. Emite la consulta

(superado por (Bitdiddle Ben) ?quién)

Después de responder, el sistema entra en un bucle infinito. Explique por qué.

**Ejercicio 4.65.**  Cy D. Fect, esperando el día en que ascenderá en la organización, plantea una consulta para encontrar todas las ruedas (utilizando la regla de la rueda de la sección  [4.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.1) ):

(rueda ?quien)

Para su sorpresa, el sistema responde

*;;; Resultados de la consulta:*  
(rueda (Warbucks Oliver))   
(rueda (Bitdiddle Ben))   
(rueda (Warbucks Oliver))   
(rueda (Warbucks Oliver))   
(rueda (Warbucks Oliver))

¿Por qué Oliver Warbucks aparece mencionado cuatro veces?

**Ejercicio 4.66.**  Ben ha generalizado el sistema de consultas para proporcionar estadísticas sobre la empresa. Por ejemplo, para encontrar los salarios totales de todos los programadores informáticos, se podrá decir:

(suma ?monto   
     (y (trabajo ?x (programador de computadoras))   
          (salario ?x ?monto)))

En general, el nuevo sistema de Ben permite expresiones de la forma

(función de acumulación < *variable* >   
                       < *patrón de consulta* >)

donde la función de acumulación puede ser cosas como suma , promedio o máximo . Ben piensa que debería ser muy fácil implementar esto. Simplemente alimentará el patrón de consulta a qeval . Esto producirá un flujo de marcos. Luego pasará este flujo a través de una función de mapeo que extrae el valor de la variable designada de cada marco en el flujo y alimentará el flujo de valores resultante a la función de acumulación. Justo cuando Ben completa la implementación y está a punto de probarla, Cy pasa caminando, todavía desconcertado por el resultado de la consulta  de la rueda en el ejercicio [4.65](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.65) . Cuando Cy le muestra a Ben la respuesta del sistema, Ben gruñe: ``¡Oh, no, mi simple esquema de acumulación no funcionará!''

¿Qué acaba de descubrir Ben? Describe un método que pueda utilizar para solucionar la situación.

**Ejercicio 4.67.**  Idee una forma de instalar un detector de bucles en el sistema de consultas para evitar los tipos de bucles simples ilustrados en el texto y en el ejercicio  [4.64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.64) . La idea general es que el sistema debe mantener algún tipo de historial de su cadena actual de deducciones y no debe comenzar a procesar una consulta en la que ya está trabajando. Describa qué tipo de información (patrones y marcos) se incluye en este historial y cómo se debe realizar la verificación. (Después de estudiar los detalles de la implementación del sistema de consultas en la sección  [4.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4) , es posible que desee modificar el sistema para incluir su detector de bucles).

**Ejercicio 4.68.**  Defina reglas para implementar la operación  inversa del ejercicio [2.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.18) , que devuelve una lista que contiene los mismos elementos que una lista dada en orden inverso. (Sugerencia: use append-to-form ). ¿Pueden sus reglas responder tanto a (reverse (1 2 3) ?x) como a (reverse ?x (1 2 3)) ?

**Ejercicio 4.69.**   Partiendo de la base de datos y de las reglas que formuló en el ejercicio  [4.63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.63) , diseñe una regla para añadir "bisnietos" a una relación de nietos. Esto debería permitir al sistema deducir que Irad es el bisnieto de Adán, o que Jabal y Jubal son los tataranietos de Adán. (Sugerencia: represente el hecho sobre Irad, por ejemplo, como ((bisnieto) Adán Irad) . Escriba reglas que determinen si una lista termina en la palabra nieto . Use esto para expresar una regla que permita derivar la relación ((bisnieto. ?rel) ?x ?y) , donde ?rel es una lista que termina en nieto .) Verifique sus reglas en consultas como ((bisnieto) ?g ?ggs) y (?relación Adán Irad) .

**[4.4.4 Implementación del sistema de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.4)**

En la sección  [4.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.2) se describió cómo funciona el sistema de consultas. Ahora completaremos los detalles presentando una implementación completa del sistema.

**[4.4.4.1 El bucle del controlador y la instanciación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.4.1)**

El bucle controlador del sistema de consultas lee repetidamente expresiones de entrada. Si la expresión es una regla o afirmación que se debe agregar a la base de datos, se agrega la información. De lo contrario, se supone que la expresión es una consulta. El controlador pasa esta consulta al evaluador qeval junto con un flujo de cuadros inicial que consiste en un único cuadro vacío. El resultado de la evaluación es un flujo de cuadros generado al satisfacer la consulta con valores de variables que se encuentran en la base de datos. Estos cuadros se utilizan para formar un nuevo flujo que consiste en copias de la consulta original en la que las variables se instancian con valores suministrados por el flujo de cuadros, y este flujo final se imprime en la terminal:

(define input-prompt ";;; Consulta de entrada:")   
(define output-prompt ";;; Consulta de resultados:")  
(define (bucle-controlador-consulta)   
  (solicitud-de-entrada solicitud-de-entrada)   
  (let ((q (proceso-de-sintaxis-consulta (leer))))   
    (cond ((¿afirmación-a-agregar? q)   
           (¡agregar-regla-o-afirmación! (agregar-cuerpo-de-afirmación q))   
           (nueva-línea)   
           (mostrar "Afirmación agregada a la base de datos").   
           (bucle-controlador-consulta))   
          (else   
           (nueva-línea)   
           (mostrar solicitud-de-salida)   
           (flujo-de-visualización (   
            mapa-de-flujo   
             (lambda (marco)   
               (instanciar q   
                            marco   
                            (lambda (v f)   
                              (contraer-signo-de-interrogación v))))   
             (qeval q (flujo-singleton '()))))   
           (bucle-controlador-consulta)))))

Aquí, como en los demás evaluadores de este capítulo, utilizamos una sintaxis abstracta para las expresiones del lenguaje de consulta. La implementación de la sintaxis de expresión, incluido el predicado assertion-to-be-added? y el selector add-assertion-body , se proporciona en la sección  [4.4.4.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.7) . Add-rule-or-assertion! se define en la sección  [4.4.4.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.5) .

Antes de realizar cualquier procesamiento en una expresión de entrada, el bucle controlador la transforma sintácticamente en una forma que hace que el procesamiento sea más eficiente. Esto implica cambiar laRepresentación de variables de patrón. Cuando se crea una instancia de la consulta, las variables que permanecen sin vincular se transforman nuevamente a la representación de entrada antes de imprimirse. Estas transformaciones se realizan mediante los dos procedimientos query-syntax-process y contract-question-mark (sección   [4.4.4.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.7) ).

Para crear una instancia de una expresión, la copiamos y reemplazamos las variables de la expresión por sus valores en un marco determinado. Los valores se instancian a su vez, ya que podrían contener variables (por ejemplo, si ?x en exp está ligado a ?y como resultado de la unificación y ?y está ligado a su vez a 5). La acción a tomar si no se puede crear una instancia de una variable se da mediante un argumento de procedimiento para crear una instancia .

(define (instantiate exp frame unbound-var-handler)  
  (define (copy exp)  
    (cond ((var? exp)  
           (let ((binding (binding-in-frame exp frame)))  
             (if binding  
                 (copy (binding-value binding))  
                 (unbound-var-handler exp frame))))  
          ((pair? exp)  
           (cons (copy (car exp)) (copy (cdr exp))))  
          (else exp)))  
  (copy exp))

The procedures that manipulate bindings are defined in section [4.4.4.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.8).

**[4.4.4.2  The Evaluator](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.4.2)**

The qeval procedure, called by the query-driver-loop, is the basic evaluator of the query system. It takes as inputs a query and a stream of frames, and it returns a stream of extended frames. It identifies special forms by a data-directed dispatch using get and put, just as we did in implementing generic operations in chapter 2. Any query that is not identified as a special form is assumed to be a simple query, to be processed by simple-query.

(define (qeval query frame-stream)  
  (let ((qproc (get (type query) 'qeval)))  
    (if qproc  
        (qproc (contents query) frame-stream)  
        (simple-query query frame-stream))))

Type and contents, defined in section [4.4.4.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.7), implement the abstract syntax of the special forms.

**[Simple queries](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_696)**

The simple-query procedure handles simple queries. It takes as arguments a simple query (a pattern) together with a stream of frames, and it returns the stream formed by extending each frame by all data-base matches of the query.

(define (simple-query query-pattern frame-stream)  
  (stream-flatmap  
   (lambda (frame)  
     (stream-append-delayed  
      (find-assertions query-pattern frame)  
      (delay (apply-rules query-pattern frame))))  
   frame-stream))

For each frame in the input stream, we use find-assertions (section [4.4.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.3)) to match the pattern against all assertions in the data base, producing a stream of extended frames, and we use apply-rules (section [4.4.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.4)) to apply all possible rules, producing another stream of extended frames. These two streams are combined (using stream-append-delayed, section [4.4.4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.6)) to make a stream of all the ways that the given pattern can be satisfied consistent with the original frame (see exercise [4.71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.71)). The streams for the individual input frames are combined using stream-flatmap (section [4.4.4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.6)) to form one large stream of all the ways that any of the frames in the original input stream can be extended to produce a match with the given pattern.

[**Compound queries**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_Temp_697)

And queries are handled as illustrated in figure [4.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.5) by the conjoin procedure. Conjoin takes as inputs the conjuncts and the frame stream and returns the stream of extended frames. First, conjoin processes the stream of frames to find the stream of all possible frame extensions that satisfy the first query in the conjunction. Then, using this as the new frame stream, it recursively applies conjoin to the rest of the queries.

(define (conjoin conjuncts frame-stream)  
  (if (empty-conjunction? conjuncts)  
      frame-stream  
      (conjoin (rest-conjuncts conjuncts)  
               (qeval (first-conjunct conjuncts)  
                      frame-stream))))

The expression

(put 'and 'qeval conjoin)

sets up qeval to dispatch to conjoin when an and form is encountered.

Or queries are handled similarly, as shown in figure [4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.6). The output streams for the various disjuncts of the or are computed separately and merged using the interleave-delayed procedure from section [4.4.4.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.6). (See exercises [4.71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.71) and [4.72](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.72).)

(define (disjoin disjuncts frame-stream)  
  (if (empty-disjunction? disjuncts)  
      the-empty-stream  
      (interleave-delayed  
       (qeval (first-disjunct disjuncts) frame-stream)  
       (delay (disjoin (rest-disjuncts disjuncts)  
                       frame-stream)))))  
(put 'or 'qeval disjoin)

Los predicados y selectores para la sintaxis de conjunciones y disyunciones se dan en la sección  [4.4.4.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.7) .

**[Filtros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_698)**

No se maneja mediante el método descrito en la sección  [4.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.2) . Intentamos extender cada cuadro en el flujo de entrada para satisfacer la consulta que se está negando, e incluimos un cuadro determinado en el flujo de salida solo si no se puede extender.

(define (niega operandos flujo-de-marco)   
  (flujo-plano   
   (lambda (marco)   
     (si (flujo-nulo? (qeval (operandos-de-consulta-negados)   
                              (flujo-singleton marco)))   
         (flujo-singleton marco)   
         el-flujo-vacío))   
   flujo-de-marco))   
(put 'not 'qeval negar)

Lisp-value es un filtro similar a not . Cada cuadro del flujo se utiliza para crear instancias de las variables del patrón, se aplica el predicado indicado y los cuadros para los que el predicado devuelve false se filtran del flujo de entrada. Se produce un error si hay variables de patrón no vinculadas.

(define (lisp-value call frame-stream)   
  (stream-flatmap   
   (lambda (frame)   
     (if (execute   
          (instancyed   
           call   
           frame   
           (lambda (v f)   
             (error "Unknown pat var -- LISP-VALUE" v))))   
         (singleton-stream frame)   
         el-flujo-vacío))   
   frame-stream))   
(put 'lisp-value 'qeval lisp-value)

Execute , que aplica el predicado a los argumentos, debe evaluar la expresión del predicado para que se aplique el procedimiento. Sin embargo, no debe evaluar los argumentos, ya que ya son los argumentos reales, no expresiones cuya evaluación (en Lisp) producirá los argumentos. Tenga en cuenta que la ejecución se implementa utilizandoevaluar y aplicar desde el sistema Lisp subyacente.

(define (ejecuta exp)   
  (aplica (eval (predicado exp) entorno-inicial-del-usuario)   
         (args exp)))

La forma especial siempre verdadera permite realizar una consulta que siempre se satisface. Ignora su contenido (normalmente vacío) y simplemente pasa por todos los marcos del flujo de entrada. El selector de cuerpo de regla utiliza siempre verdadero (sección  [4.4.4.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.7) )proporcionar cuerpos a reglas que se definieron sin cuerpos (es decir, reglas cuyas conclusiones siempre se satisfacen).

(define (siempre-verdadero ignora flujo-de-marcos) flujo-de-marcos)   
(pone 'siempre-verdadero 'qeval siempre-verdadero)

Los selectores que definen la sintaxis de not y lisp-value se dan en la sección  [4.4.4.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.7) .

**[4.4.4.3 Búsqueda de afirmaciones mediante la coincidencia de patrones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.4.3)**

Find-assertions , llamado por simple-query (sección  [4.4.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.2) ), toma como entrada un patrón y un marco. Devuelve un flujo de marcos, cada uno de los cuales extiende el dado por una coincidencia de la base de datos del patrón dado. Utiliza fetch-assertions (sección  [4.4.4.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.5) ) para obtener un flujo de todas las afirmaciones en la base de datos que se deben verificar para una coincidencia con el patrón y el marco. La razón para fetch-assertions aquí es que a menudo podemos aplicar pruebas simples que eliminarán muchas de las entradas en la base de datos del grupo de candidatos para una coincidencia exitosa. El sistema aún funcionaría si eliminamos fetch-assertions y simplemente verificamos un flujo de todas las afirmaciones en la base de datos, pero el cálculo sería menos eficiente porque necesitaríamos hacer muchas más llamadas al comparador.

(define (marco de patrón de búsqueda de afirmaciones)   
  (stream-flatmap (lambda (dato)   
                    (marco de patrón de datos de verificación de una afirmación))   
                  (marco de patrón de obtención de afirmaciones)))

Check-an-assertion toma como argumentos un patrón, un objeto de datos (aserción) y un marco y devuelve un flujo de un elemento que contiene el marco extendido o el flujo vacío si la coincidencia falla.

(define (verifica-una-afirmación consulta-de-afirmación pat-de-consulta marco-de-consulta)   
  (let ((resultado-de-coincidencia   
         (coincidencia-de-patrón consulta-pat aserción marco-de-consulta)))   
    (if (eq? resultado-de-coincidencia 'falló)   
        el-flujo-vacío   
        (flujo-singleton resultado-de-coincidencia))))

El comparador de patrones básico devuelve el símbolo failed o una extensión del marco dado. La idea básica del comparador es comprobar el patrón con los datos, elemento por elemento, acumulando enlaces para las variables del patrón. Si el patrón y el objeto de datos son los mismos, la comparación es exitosa y devolvemos el marco de enlaces acumulados hasta el momento. De lo contrario, si el patrón es una variable, ampliamos el marco actual vinculando la variable a los datos, siempre que esto sea consistente con los enlaces que ya están en el marco. Si el patrón y los datos son ambos pares, hacemos coincidir (de forma recursiva) el car del patrón con el car de los datos para producir un marco; en este marco, hacemos coincidir el cdr del patrón con el cdr de los datos. Si ninguno de estos casos es aplicable, la comparación falla y devolvemos el símbolo failed .

(define (patrón-coincidencia pat dat marco)   
  (cond ((eq? marco 'falló) 'falló)   
        ((igual? pat dat) marco)   
        ((var? pat) (extender-si-es-consistente pat dat marco))   
        ((y (par? pat) (par? dat))   
         (patrón-coincidencia (cdr pat)   
                        (cdr dat)   
                        (patrón-coincidencia (car pat)   
                                       (car dat)   
                                       marco)))   
        (de lo contrario 'falló)))

A continuación se muestra el procedimiento que extiende un marco agregando un nuevo enlace, si este es consistente con los enlaces que ya existen en el marco:

(define (extiende-si-es-consistente var dat frame)   
  (let ((enlace (enlace-en-marco var frame)))   
    (si enlaza   
        (coincidencia-de-patrones (enlace-valor enlazante) dat frame)   
        (extiende var dat frame))))

Si no hay ninguna vinculación para la variable en el marco, simplemente agregamos la vinculación de la variable a los datos. De lo contrario, hacemos coincidir, en el marco, los datos con el valor de la variable en el marco. Si el valor almacenado contiene solo constantes, como debe ser si se almacenó durante la comparación de patrones por extend-if-consistent , entonces la comparación simplemente prueba si los valores almacenados y nuevos son los mismos. Si es así, devuelve el marco sin modificar; si no, devuelve una indicación de falla. Sin embargo, el valor almacenado puede contener variables de patrón si se almacenó durante la unificación (consulte la sección  [4.4.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.4) ). La comparación recursiva del patrón almacenado con los nuevos datos agregará o verificará las vinculaciones para las variables en este patrón. Por ejemplo, supongamos que tenemos un marco en el que ?x está vinculado a (f ?y) y ?y no está vinculado, y deseamos aumentar este marco mediante una vinculación de ?x a (fb) . Buscamos ?x y encontramos que está vinculado a (f ?y) . Esto nos lleva a hacer coincidir (f ?y) con el nuevo valor propuesto (fb) en el mismo marco. Finalmente, esta coincidencia extiende el marco agregando un enlace de ?y a b . ?X permanece enlazado a (f ?y) . Nunca modificamos un enlace almacenado y nunca almacenamos más de un enlace para una variable dada.

Los procedimientos utilizados por extend-if-consistent para manipular enlaces se definen en la sección  [4.4.4.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.8) .

**[Patrones con colas punteadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_699)**

Si un patrón contiene un punto seguido de una variable de patrón, la variable de patrón coincide con el resto de la lista de datos (en lugar del siguiente elemento de la lista de datos), tal como se esperaría con la notación de cola punteada descrita en el ejercicio  [2.20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.20) . Aunque el comparador de patrones que acabamos de implementar no busca puntos, se comporta como queremos. Esto se debe a que la primitiva de lectura de Lisp , que se utiliza por query-driver-loop para leer la consulta y representarla como una estructura de lista, trata los puntos de una manera especial.

Cuando read ve un punto, en lugar de hacer que el siguiente elemento sea el siguiente elemento de una lista (el car de un cons cuyo cdr será el resto de la lista), hace que el siguiente elemento sea el cdr de la estructura de lista. Por ejemplo, la estructura de lista producida por read para el patrón (computer ?type) podría construirse evaluando la expresión (cons 'computer (cons '?type '())) , y la de (computer . ?type) podría construirse evaluando la expresión (cons 'computer '?type) .

Por lo tanto, como la búsqueda de patrones compara recursivamente los car y los cdr de una lista de datos y un patrón que tenía un punto, finalmente busca la variable después del punto (que es un cdr del patrón) con una sublista de la lista de datos, vinculando la variable a esa lista. Por ejemplo, buscar el patrón (computer . ?type) con (computer programmer trainee) buscará ?type con la lista (programmer trainee) .

[**4.4.4.4 Reglas y Unificación**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_4.4.4.4)

Apply-rules es el análogo de reglas de find-assertions (sección  [4.4.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.3) ). Toma como entrada un patrón y un marco, y forma un flujo de marcos de extensión aplicando reglas de la base de datos. Stream-flatmap asigna apply-a-rule al flujo de reglas posiblemente aplicables (seleccionadas por fetch-rules , sección  [4.4.4.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.5) ) y combina los flujos de marcos resultantes.

(define (marco de patrón aplicar-reglas)   
  (stream-flatmap (lambda (regla)   
                    (marco de patrón aplicar-una-regla))   
                  (marco de patrón obtener-reglas)))

Apply-a-rule aplica reglas utilizando el método descrito en la sección [4.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.2) . Primero amplía su marco de argumentos unificando la conclusión de la regla con el patrón en el marco dado. Si esto tiene éxito, evalúa el cuerpo de la regla en este nuevo marco.

Sin embargo, antes de que ocurra nada de esto, el programa renombra todas las variables de la regla con nombres nuevos y únicos. La razón de esto es evitar que las variables para diferentes aplicaciones de reglas se confundan entre sí. Por ejemplo, si dos reglas usan una variable llamada ?x , entonces cada una puede agregar un enlace para ?x al marco cuando se aplica. Estos dos ?x no tienen nada que ver entre sí, y no debemos engañarnos pensando que los dos enlaces deben ser consistentes. En lugar de renombrar las variables, podríamos idear una estructura de entorno más inteligente; sin embargo, el enfoque de renombramiento que hemos elegido aquí es el más sencillo, aunque no el más eficiente. (Véase el ejercicio  [4.79](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.79) .) Aquí está el procedimiento de aplicación de una regla :

(define (aplicar-una-regla regla patrón-de-consulta marco-de-consulta)   
  (let ((regla-limpia (renombrar-variables-en-regla)))   
    (let ((resultado-unificado   
           (coincidencia-unificada patrón-de-consulta   
                        (conclusión regla-limpia)   
                        marco-de-consulta)))   
      (if (eq? resultado-unificado 'falló)   
          el-flujo-vacío   
          (qeval (cuerpo-de-regla regla-limpia)   
                 (flujo-singleton resultado-unificado))))))

El cuerpo de la regla y la conclusión de los selectores que extraen partes de una regla se definen en la sección  [4.4.4.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.7) .

Generamos nombres de variables únicos asociando un identificador único (como un número) con cada aplicación de regla y combinando este identificador con los nombres de variable originales. Por ejemplo, si el identificador de la aplicación de regla es 7, podríamos cambiar cada ?x en la regla a ?x-7 y cada ?y en la regla a ?y-7 . ( Make-new-variable y new-rule-application-id se incluyen con los procedimientos de sintaxis en la sección  [4.4.4.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.7) .)

(define (regla de renombrar variables)   
  (let ((id-aplicacion-regla (id-aplicacion-regla-nueva)))   
    (define (exp-caminar-arbol)   
      (cond ((var? exp)   
             (crear-nueva-variable exp id-aplicacion-regla))   
            ((par? exp)   
             (cons (caminar-arbol (exp-auto))   
                   (caminar-arbol (cdr exp))))   
            (else exp)))   
    (regla de caminar-arbol)))

El algoritmo de unificación se implementa como un procedimiento que toma como entradas dos patrones y un marco y devuelve el marco extendido o el símbolo failed . El unificador es como el comparador de patrones excepto que es simétrico: se permiten variables en ambos lados de la coincidencia. Unify-match es básicamente lo mismo que pattern-match , excepto que hay un código adicional (marcado `` \*\*\* '' a continuación) para manejar el caso en el que el objeto en el lado derecho de la coincidencia es una variable.

(define (unificar-coincidir p1 p2 cuadro)   
  (cond ((eq? cuadro 'falló) 'falló)   
        ((igual? p1 p2) cuadro)   
        ((var? p1) (extender-si-es-posible p1 p2 cuadro))   
        ((var? p2) (extender-si-es-posible p2 p1 cuadro))   *; \*\*\**  
        ((y (pareja? p1) (pareja? p2))   
         (unificar-coincidir (cdr p1)   
                      (cdr p2)   
                      (unificar-coincidir (coche p1)   
                                   (coche p2)   
                                   cuadro)))   
        (de lo contrario 'falló)))

En la unificación, al igual que en la comparación de patrones unilateral, queremos aceptar una extensión propuesta del marco solo si es consistente con las vinculaciones existentes. El procedimiento extend-if-possible utilizado en la unificación es el mismo que el extend-if-consistent utilizado en la comparación de patrones, excepto por dos comprobaciones especiales, marcadas `` \*\*\* '' en el programa siguiente. En el primer caso, si la variable que estamos tratando de comparar no está vinculada, pero el valor con el que estamos tratando de compararla es en sí mismo una variable (diferente), es necesario verificar si el valor está vinculado y, de ser así, comparar su valor. Si ambas partes de la comparación no están vinculadas, podemos vincular cualquiera de ellas con la otra.

La segunda comprobación se ocupa de los intentos de vincular una variable a un patrón que incluye esa variable. Una situación de este tipo puede ocurrir siempre que una variable se repita en ambos patrones. Considere, por ejemplo, la unificación de los dos patrones (?x ?x) y (?y < *expresión que involucra a ?y* >) en un marco donde tanto ?x como ?y no están vinculados. Primero , ?x se compara con ?y , lo que hace que ?x se vincule con ?y . A continuación, el mismo ?x se compara con la expresión dada que involucra a ?y . Dado que ?x ya está vinculado a ?y , esto da como resultado la coincidencia de ?y con la expresión. Si pensamos en el unificador como la búsqueda de un conjunto de valores para las variables de patrón que hacen que los patrones sean iguales, entonces estos patrones implican instrucciones para encontrar un ?y tal que ?y sea igual a la expresión que involucra a ?y . No existe un método general para resolver tales ecuaciones, por lo que rechazamos tales vinculaciones; estos casos se reconocen por el predicado depende de? . [80](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_700) Por otro lado, no queremos rechazar los intentos de vincular una variable a sí misma. Por ejemplo, considere la unificación de (?x ?x) y (?y ?y) . El segundo intento de vincular ?x a ?y hace coincidir ?y (el valor almacenado de ?x ) con ?y (el nuevo valor de ?x ). Esto se soluciona con la cláusula equal? ​​de unify-match .

(define (extiende-si-es-posible var val marco)   
  (let ((enlace (enlace-en-marco var marco)))   
    (cond (enlace   
           (unificar-coincidencia   
            (valor-enlace enlace) val marco))   
          ((var? val)                       *; \*\*\**  
           (let ((enlace (enlace-en-marco val marco)))   
             (if enlace   
                 (unificar-coincidencia   
                  var (valor-enlace enlace) marco)   
                 (extiende var val marco))))   
          ((depende-de? val var marco)      *; \*\*\**  
           'falló)   
          (else (extiende var val marco)))))

¿Depends-on? es un predicado que prueba si una expresión propuesta como valor de una variable de patrón depende de la variable. Esto debe hacerse en relación con el marco actual porque la expresión puede contener ocurrencias de una variable que ya tiene un valor que depende de nuestra variable de prueba. La estructura de ¿ Depends-on? es un recorrido recursivo simple por un árbol en el que sustituimos los valores de las variables cuando sea necesario.

(define (depende-de? exp var marco)   
  (define (caminar-árbol e)   
    (cond ((var? e)   
           (si (igual? var e)   
               verdadero   
               (let ((b (enlace-en-marco e marco)))   
                 (si b   
                     (caminar-árbol (valor-enlace b))   
                     falso))))   
          ((par? e)   
           (o (caminar-árbol (coche e))   
               (caminar-árbol (cdr e))))   
          (sino falso)))   
  (caminar-árbol exp))

**[4.4.4.5 Mantenimiento de la base de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.4.5)**

Un problema importante en el diseño de lenguajes de programación lógica es el de organizar las cosas de modo que se examinen la menor cantidad posible de entradas irrelevantes de la base de datos al verificar un patrón dado. En nuestro sistema, además de almacenar todas las afirmaciones en un gran flujo, almacenamos todas las afirmaciones cuyos coches son símbolos constantes en flujos separados, en una tabla indexada por el símbolo. Para obtener una afirmación que pueda coincidir con un patrón, primero verificamos si el coche del patrón es un símbolo constante. Si es así, devolvemos (para probarlo utilizando el comparador) todas las afirmaciones almacenadas que tienen el mismo coche . Si el coche del patrón no es un símbolo constante, devolvemos todas las afirmaciones almacenadas. Los métodos más inteligentes también podrían aprovechar la información en el marco, o intentar también optimizar el caso en el que el coche del patrón no sea un símbolo constante. Evitamos construir nuestros criterios para indexar (utilizando el coche , manejando solo el caso de símbolos constantes) en el programa; en su lugar, invocamos predicados y selectores que incorporan nuestros criterios.

(define LAS-AFIRMACIONES el-flujo-vacío)  
(define (patrón de obtención de afirmaciones)   
  (si (patrón de uso de índice)   
      (patrón de obtención de afirmaciones indexadas)   
      (obtener todas las afirmaciones)))   
(define (obtener todas las afirmaciones) LAS ASERCIONES)   
(define (patrón de obtención de afirmaciones indexadas)   
  (patrón de obtención de flujo (patrón de clave de índice) 'flujo de afirmaciones))

Get-stream busca una secuencia en la tabla y devuelve una secuencia vacía si no hay nada almacenado allí.

(define (get-stream clave1 clave2)   
  (let ((s (get clave1 clave2)))   
    (if s s the-vacio-stream)))

Las reglas se almacenan de manera similar, utilizando el carro de la conclusión de la regla. Sin embargo, las conclusiones de las reglas son patrones arbitrarios, por lo que difieren de las afirmaciones en que pueden contener variables. Un patrón cuyo carro es un símbolo constante puede coincidir con reglas cuyas conclusiones comiencen con una variable, así como con reglas cuyas conclusiones tengan el mismo carro . Por lo tanto, al obtener reglas que podrían coincidir con un patrón cuyo carro es un símbolo constante, obtenemos todas las reglas cuyas conclusiones comiencen con una variable, así como aquellas cuyas conclusiones tengan el mismo carro que el patrón. Para este propósito, almacenamos todas las reglas cuyas conclusiones comiencen con una variable en un flujo separado en nuestra tabla, indexado por el símbolo ? .

(define LAS-REGLAS el-flujo-vacío)  
(define (patrón de obtención de reglas)   
  (si (patrón de uso-índice)   
      (patrón de obtención de reglas indexadas)   
      (obtener-todas-las-reglas)))   
(define (obtener-todas-las-reglas) LAS-REGLAS)   
(define (patrón de obtención de reglas indexadas)   
  (stream-append   
   (get-stream (patrón de clave-índice) 'rule-stream)   
   (get-stream '? 'rule-stream)))

¡Add-rule-or-assertion! es utilizado por query-driver-loop para agregar aserciones y reglas a la base de datos. Cada elemento se almacena en el índice, si corresponde, y en un flujo de todas las aserciones o reglas en la base de datos.

(define (¡agrega-regla-o-afirmación! afirmación)   
  (si (¿regla? afirmación)   
      (¡agrega-regla! afirmación)   
      (¡agrega-afirmación! afirmación)))   
(define (¡agrega-afirmación! afirmación)   
  (almacena-afirmación-en-índice afirmación)   
  (let ((antiguas-afirmaciones LAS-ASERRACIONES))   
    (set! LAS-ASERRACIONES   
          (cons-stream afirmación aserción aserciones-antiguas))   
    'ok))   
(define (¡agrega-regla! regla)   
  (almacena-regla-en-índice regla)   
  (let ((antiguas-reglas LAS-REGLAS))   
    (set! LAS-REGLAS (cons-stream regla reglas-antiguas))   
    'ok))

Para almacenar una afirmación o una regla, verificamos si se puede indexar. Si es así, la almacenamos en el flujo correspondiente.

(define (almacenar-aserción-en-índice aserción)   
  (si (¿indexable? aserción)   
      (let ((clave (índice-clave-de aserción)))   
        (let ((flujo-de-aserción-actual   
               (obtener-flujo clave 'flujo-de-aserción)))   
          (poner clave   
               'flujo-de-aserción   
               (flujo-de-cons aserción   
                            flujo-de-aserción-actual))))))   
(define (almacenar-regla-en-índice regla)   
  (let ((patrón (conclusión regla)))   
    (si (¿indexable? patrón)   
        (let ((clave (índice-clave-de patrón)))   
          (let ((flujo-de-regla-actual   
                 (obtener-flujo clave 'flujo-de-regla)))   
            (poner clave   
                 'flujo-de-regla   
                 (flujo-de-cons regla flujo   
                              -de-regla-actual)))))))

Los siguientes procedimientos definen cómo se utiliza el índice de la base de datos. Un patrón (una afirmación o una conclusión de regla) se almacenará en la tabla si comienza con un símbolo de variable o constante.

(definir (¿indexable? pat)   
  (o (símbolo-constante? (car pat))   
      (var? (car pat))))

La clave bajo la cual se almacena un patrón en la tabla es ? (si comienza con una variable) o el símbolo de constante con el que comienza.

(define (índice-clave-de pat)   
  (let ((clave (car pat)))   
    (if (var? clave) '? clave)))

El índice se utilizará para recuperar elementos que puedan coincidir con un patrón si el patrón comienza con un símbolo constante.

(define (use-index? pat)   
  (símbolo-constante? (car pat)))

**Ejercicio 4.70.**   ¿Cuál es el propósito de los enlaces let en los procedimientos add-assertion! y add-rule!? ¿Qué estaría mal con la siguiente implementación de add-assertion!? Pista: Recuerde la definición del flujo infinito de unos en la sección  [3.5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.2) : (define ones (cons-stream 1 ones)) .

(define (add-assertion! assertion)   
  (store-assertion-in-index assertion)   
  (set! THE-ASSERTIONS   
        (cons-stream assertion THE-ASSERTIONS))   
  'ok)

**[4.4.4.6 Operaciones de transmisión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.4.6)**

El sistema de consulta utiliza algunas operaciones de flujo que no se presentaron en el capítulo 3.

Stream-append-delayed e interleave-delayed son iguales que stream-append e interleave (sección  [3.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.3) ), excepto que toman un argumento retrasado (como el procedimiento  integral de la sección [3.5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.4) ). Esto pospone el bucle en algunos casos (consulte el ejercicio  [4.71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.71) ).

(define (flujo-anexar-retrasado s1 retrasado-s2)   
  (si (flujo-nulo? s1)   
      (fuerza retrasado-s2)   
      (cons-flujo   
       (flujo-car s1)   
       (flujo-anexar-retrasado (flujo-cdr s1) retrasado-s2))))  
(define (entrelazado-retardado s1 retrasado-s2)   
  (si (flujo-nulo? s1)   
      (fuerza retrasado-s2)   
      (cons-flujo   
       (flujo-car s1)   
       (entrelazado-retardado (fuerza retrasado-s2)   
                           (retraso (flujo-cdr s1))))))

Stream-flatmap , que se utiliza en todo el evaluador de consultas para mapear un procedimiento sobre un flujo de marcos y combinar los flujos de marcos resultantes, es el análogo de flujo del procedimiento flatmap introducido para listas ordinarias en la sección  [2.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.2.3) . Sin embargo, a diferencia del flatmap ordinario , acumulamos los flujos con un proceso de intercalación, en lugar de simplemente anexarlos (ver ejercicios  [4.72](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.72) y   [4.73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.73) ).

(define (procedimientos stream-flatmap)   
  (aplana-flujo (procedimientos stream-map)))  
(define (flujo aplanado)   
  (si (flujo-nulo? flujo)   
      el flujo vacío   
      (entrelazado-retrasado   
       (flujo-car flujo)   
       (retraso (flujo-aplanado (flujo-cdr flujo))))))

El evaluador también utiliza el siguiente procedimiento simple para generar un flujo que consta de un solo elemento:

(define (flujo-singleton x)   
  (flujo-cons x el-flujo-vacío))

**[4.4.4.7 Procedimientos de sintaxis de consulta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.4.7)**

Los procedimientos type-tag y contents , utilizados por qeval (sección  [4.4.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.2) ), especifican que una forma especial se identifica mediante el símbolo en su car . Son iguales a los procedimientos type-tag y contents de la sección  [2.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.4.2) , excepto por el mensaje de error.

(define (tipo exp)   
  (si (¿par? exp)   
      (auto exp)   
      (error "Expresión desconocida TIPO" exp)))   
(define (contenido exp)   
  (si (¿par? exp)   
      (cdr exp)   
      (error "Expresión desconocida CONTENIDO" exp)))

The following procedures, used by query-driver-loop (in section [4.4.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.1)), specify that rules and assertions are added to the data base by expressions of the form (assert! <*rule-or-assertion*>):

(define (assertion-to-be-added? exp)  
  (eq? (type exp) 'assert!))  
(define (add-assertion-body exp)  
  (car (contents exp)))

Here are the syntax definitions for the and, or, not, and lisp-value special forms (section [4.4.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.2)):

(define (empty-conjunction? exps) (null? exps))  
(define (first-conjunct exps) (car exps))  
(define (rest-conjuncts exps) (cdr exps))  
(define (empty-disjunction? exps) (null? exps))  
(define (first-disjunct exps) (car exps))  
(define (rest-disjuncts exps) (cdr exps))  
(define (negated-query exps) (car exps))  
(define (predicate exps) (car exps))  
(define (args exps) (cdr exps))

The following three procedures define the syntax of rules:

(define (rule? statement)  
  (tagged-list? statement 'rule))  
(define (conclusion rule) (cadr rule))  
(define (rule-body rule)  
  (if (null? (cddr rule))  
      '(always-true)  
      (caddr rule)))

Query-driver-loop (section [4.4.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.1)) calls query-syntax-process to transform pattern variables in the expression, which have the form ?symbol, into the internal format (? symbol). That is to say, a pattern such as (job ?x ?y) is actually represented internally by the system as (job (? x) (? y)). This increases the efficiency of query processing, since it means that the system can check to see if an expression is a pattern variable by checking whether the car of the expression is the symbol ?, rather than having to extract characters from the symbol. The syntax transformation is accomplished by the following procedure:[81](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_702)

(define (query-syntax-process exp)  
  (map-over-symbols expand-question-mark exp))  
(define (map-over-symbols proc exp)  
  (cond ((pair? exp)  
         (cons (map-over-symbols proc (car exp))  
               (map-over-symbols proc (cdr exp))))  
        ((symbol? exp) (proc exp))  
        (else exp)))  
(define (expand-question-mark symbol)  
  (let ((chars (symbol->string symbol)))  
    (if (string=? (substring chars 0 1) "?")  
        (list '?  
              (string->symbol  
               (substring chars 1 (string-length chars))))  
        symbol)))

Once the variables are transformed in this way, the variables in a pattern are lists starting with ?, and the constant symbols (which need to be recognized for data-base indexing, section [4.4.4.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.5)) are just the symbols.

(define (var? exp)  
  (tagged-list? exp '?))  
(define (constant-symbol? exp) (symbol? exp))

Unique variables are constructed during rule application (in section [4.4.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.4)) by means of the following procedures. The unique identifier for a rule application is a number, which is incremented each time a rule is applied.

(define rule-counter 0)  
(define (new-rule-application-id)  
  (set! rule-counter (+ 1 rule-counter))  
  rule-counter)  
(define (make-new-variable var rule-application-id)  
  (cons '? (cons rule-application-id (cdr var))))

When query-driver-loop instantiates the query to print the answer, it converts any unbound pattern variables back to the right form for printing, using

(define (contract-question-mark variable)  
  (string->symbol  
   (string-append "?"   
     (if (number? (cadr variable))  
         (string-append (symbol->string (caddr variable))  
                        "-"  
                        (number->string (cadr variable)))  
         (symbol->string (cadr variable))))))

**[4.4.4.8  Frames and Bindings](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_4.4.4.8)**

Frames are represented as lists of bindings, which are variable-value pairs:

(define (make-binding variable value)  
  (cons variable value))  
(define (binding-variable binding)  
  (car binding))  
(define (binding-value binding)  
  (cdr binding))  
(define (binding-in-frame variable frame)  
  (assoc variable frame))  
(define (extend variable value frame)  
  (cons (make-binding variable value) frame))

**Exercise 4.71.**  Louis Reasoner wonders why the simple-query and disjoin procedures (section [4.4.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4.2)) are implemented using explicit delay operations, rather than being defined as follows:

(define (simple-query query-pattern frame-stream)  
  (stream-flatmap  
   (lambda (frame)  
     (stream-append (find-assertions query-pattern frame)  
                    (apply-rules query-pattern frame)))  
   frame-stream))  
(define (disjoin disjuncts frame-stream)  
  (if (empty-disjunction? disjuncts)  
      the-empty-stream  
      (interleave  
       (qeval (first-disjunct disjuncts) frame-stream)  
       (disjoin (rest-disjuncts disjuncts) frame-stream))))

Can you give examples of queries where these simpler definitions would lead to undesirable behavior?

**Exercise 4.72.**  Why do disjoin and stream-flatmap interleave the streams rather than simply append them? Give examples that illustrate why interleaving works better. (Hint: Why did we use interleave in section [3.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.5.3)?)

**Exercise 4.73.**  Why does flatten-stream use delay explicitly? What would be wrong with defining it as follows:

(define (flatten-stream stream)  
  (if (stream-null? stream)  
      the-empty-stream  
      (interleave  
       (stream-car stream)  
       (flatten-stream (stream-cdr stream)))))

**Exercise 4.74.**  Alyssa P. Hacker proposes to use a simpler version of stream-flatmap in negate, lisp-value, and find-assertions. She observes that the procedure that is mapped over the frame stream in these cases always produces either the empty stream or a singleton stream, so no interleaving is needed when combining these streams.

a. Fill in the missing expressions in Alyssa's program.

(define (simple-stream-flatmap proc s)  
  (simple-flatten (stream-map proc s)))  
  
(define (simple-flatten stream)  
  (stream-map <*??*>  
              (stream-filter <*??*> stream)))

b. Does the query system's behavior change if we change it in this way?

**Exercise 4.75.**  Implement for the query language a new special form called unique. Unique should succeed if there is precisely one item in the data base satisfying a specified query. For example,

(unique (job ?x (computer wizard)))

should print the one-item stream

(único (trabajo (Bitdiddle Ben) (mago de la informática)))

Ya que Ben es el único mago de las computadoras, y

(único (trabajo ?x (programador de computadoras)))

Debería imprimir el flujo vacío, ya que hay más de un programador de computadoras. Además,

(y (trabajo ?x ?j) (único (trabajo ?cualquiera ?j)))

Debe enumerar todos los puestos que son ocupados por una sola persona y las personas que los ocupan.

Hay dos partes para implementar unique . La primera es escribir un procedimiento que maneje esta forma especial, y la segunda es hacer que qeval envíe a ese procedimiento. La segunda parte es trivial, ya que qeval realiza su envío de una manera dirigida por datos. Si su procedimiento se llama uniquely-asserted , todo lo que necesita hacer es

(poner 'unique 'qeval afirmado de forma única)

y qeval enviará a este procedimiento para cada consulta cuyo tipo ( car ) sea el símbolo unique .

El verdadero problema es escribir el procedimiento uniquely-asserted . Este debería tomar como entrada el contenido ( cdr ) de la consulta única , junto con un flujo de marcos. Para cada marco del flujo, debería usar qeval para encontrar el flujo de todas las extensiones del marco que satisfacen la consulta dada. Cualquier flujo que no tenga exactamente un elemento en él debería eliminarse. Los flujos restantes deberían devolverse para acumularse en un gran flujo que sea el resultado de la consulta única . Esto es similar a la implementación de la forma no especial.

Pruebe su implementación formulando una consulta que enumere a todas las personas que supervisan exactamente a una persona.

**Ejercicio 4.76.**  Nuestra implementación de y como una combinación en serie de consultas (figura  [4.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_4.5) ) es elegante, pero es ineficiente porque al procesar la segunda consulta de y debemos escanear la base de datos para cada cuadro producido por la primera consulta. Si la base de datos tiene *N* elementos, y una consulta típica produce una cantidad de cuadros de salida proporcional a *N* (digamos *N* / *k* ), entonces escanear la base de datos para cada cuadro producido por la primera consulta requerirá *N* 2 / *k* llamadas al comparador de patrones. Otro enfoque sería procesar las dos cláusulas de y por separado, luego buscar todos los pares de cuadros de salida que sean compatibles. Si cada consulta produce *N* / *k* cuadros de salida, entonces esto significa que debemos realizar *N* 2 / *k* 2 verificaciones de compatibilidad, un factor de *k* menos que la cantidad de coincidencias requeridas en nuestro método actual.

Diseñe una implementación de y que utilice esta estrategia. Debe implementar un procedimiento que tome dos marcos como entradas, verifique si los enlaces en los marcos son compatibles y, si es así, produzca un marco que fusione los dos conjuntos de enlaces. Esta operación es similar a la unificación.

**Ejercicio 4.77.**  En la sección  [4.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.3) vimos que not y lisp-value pueden hacer que el lenguaje de consulta dé respuestas "incorrectas" si estas operaciones de filtrado se aplican a marcos en los que no se han vinculado variables. Piense en una forma de solucionar este problema. Una idea es realizar el filtrado de forma "retardada" añadiendo al marco una "promesa" de filtrado que se cumpla sólo cuando se hayan vinculado suficientes variables para que la operación sea posible. Podríamos esperar a realizar el filtrado hasta que se hayan realizado todas las demás operaciones. Sin embargo, por el bien de la eficiencia, nos gustaría realizar el filtrado lo antes posible para reducir la cantidad de marcos intermedios generados.

**Ejercicio 4.78.**  Rediseñe el lenguaje de consulta como un programa no determinista que se implementará utilizando el evaluador de la sección  [4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3) , en lugar de como un proceso de secuencia. En este enfoque, cada consulta producirá una única respuesta (en lugar de la secuencia de todas las respuestas) y el usuario puede escribir try-again para ver más respuestas. Debería encontrar que gran parte del mecanismo que construimos en esta sección está subsumido por la búsqueda no determinista y el retroceso. Sin embargo, probablemente también encontrará que su nuevo lenguaje de consulta tiene diferencias sutiles en el comportamiento con respecto al implementado aquí. ¿Puede encontrar ejemplos que ilustren esta diferencia?

**Ejercicio 4.79.**  Cuando implementamos el evaluador de Lisp en la sección  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) , vimos cómo usar entornos locales para evitar conflictos de nombres entre los parámetros de los procedimientos. Por ejemplo, al evaluar

(define (cuadrado x)   
  (\* x x))   
(define (suma de cuadrados x y)   
  (+ (cuadrado x) (cuadrado y)))   
(suma de cuadrados 3 4)

No hay confusión entre la x en el cuadrado y la x en la suma de cuadrados , porque evaluamos el cuerpo de cada procedimiento en un entorno que está especialmente construido para contener enlaces para las variables locales. En el sistema de consultas, utilizamos una estrategia diferente para evitar conflictos de nombres al aplicar reglas. Cada vez que aplicamos una regla, renombramos las variables con nuevos nombres que se garantiza que son únicos. La estrategia análoga para el evaluador de Lisp sería eliminar los entornos locales y simplemente renombrar las variables en el cuerpo de un procedimiento cada vez que aplicamos el procedimiento.

Implemente para el lenguaje de consulta un método de aplicación de reglas que utilice entornos en lugar de renombrar. Vea si puede aprovechar la estructura de su entorno para crear construcciones en el lenguaje de consulta para tratar con sistemas grandes, como el análogo de reglas de los procedimientos estructurados en bloques. ¿Puede relacionar algo de esto con el problema de hacer deducciones en un contexto (por ejemplo, ``Si supusiera que *P* fuera cierto, entonces podría deducir *A* y *B* '') como método de resolución de problemas? (Este problema es abierto. Una buena respuesta probablemente valga la pena para un doctorado).

[58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_645) La programación lógica es el resultado de un largo procesoHistoria de la investigación en la demostración automática de teoremas. Los primeros programas de demostración de teoremas podían lograr muy poco, porque buscaban exhaustivamente el espacio de posibles demostraciones. El gran avance que hizo plausible esa búsqueda fue el descubrimiento, a principios de los años 1960, de la*algoritmo de unificación* y el*principio de resolución* (Robinson 1965). La resolución fue utilizada, por ejemplo, porGreen y Raphael (1968) (véase también Green 1969) como base para un sistema deductivo de preguntas y respuestas. Durante la mayor parte de este período, los investigadores se concentraron en algoritmos que garantizaban encontrar una prueba, si es que existía. Tales algoritmos eran difíciles de controlar y de dirigir hacia una prueba.Hewitt (1969) reconoció la posibilidad de fusionar la estructura de control de un lenguaje de programación con las operaciones de un sistema de manipulación lógica, lo que condujo al trabajo en búsqueda automática mencionado en la sección  [4.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3.1) (nota al pie  [47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#footnote_Temp_603) ). Al mismo tiempo que esto se estaba haciendo,Colmerauer, en Marsella, estaba desarrollando sistemas basados ​​en reglas para manipular el lenguaje natural (véase Colmerauer et al. 1973). Inventó un lenguaje de programación llamadoPrólogo para representar esas reglas.Kowalski (1973; 1979), en Edimburgo, reconoció que la ejecución de un programa Prolog podía interpretarse como una demostración de teoremas (utilizando una técnica de demostración llamada demostración lineal).Resolución de la cláusula Horn). La fusión de las dos últimas corrientes dio lugar al movimiento de programación lógica. Por tanto, al atribuirse el mérito del desarrollo de la programación lógica, los franceses pueden señalar la génesis de Prolog en elUniversidad de Marsella, mientras que los británicos pueden destacar el trabajo en laUniversidad de Edimburgo. Según la gente deMIT, la programación lógica fue desarrollada por estos grupos en un intento de averiguar de qué estaba hablando Hewitt en su brillante pero impenetrable tesis doctoral. Para una historia de la lógicaprogramación, véase Robinson 1983.

[59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_646) Para ver la correspondencia entre las reglas y el procedimiento, supongamos quexen el procedimiento (dondexno está vacío) corresponde a(cons uv)en la regla. Entonces,zen la regla corresponde a laadiciónde(cdr x)ey.

[60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_647) Esto ciertamente no libera al usuario de todo el problema de cómo calcular la respuesta. Hay muchos conjuntos de reglas matemáticamente equivalentes para formular lade anexión, de los cuales sólo algunos pueden convertirse en dispositivos efectivos para calcular en cualquier dirección. Además, a veces la información de «qué es» no da ninguna pista de «cómo» calcular una respuesta. Por ejemplo, considere el problema de calcular y*tal*que*y*2=*x*.

[61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_648) El interés por la programación lógica alcanzó su punto máximoA principios de los años 80, el gobierno japonés inició un ambicioso proyecto destinado a construir ordenadores superrápidos optimizados para ejecutar lenguajes de programación lógica. La velocidad de dichos ordenadores se mediría en LIPS (inferencias lógicas por segundo) en lugar de las habituales FLOPS (operaciones de coma flotante por segundo). Aunque el proyecto tuvo éxito en el desarrollo del hardware y el software según lo previsto originalmente, la industria informática internacional se movió en una dirección diferente. VéaseFeigenbaum y Shrobe 1993 para una evaluación general del proyecto japonés. La comunidad de programación lógica también ha pasado a considerar la programación relacional basada en técnicas distintas a la simple comparación de patrones, como la capacidad de manejar restricciones numéricas como las ilustradas en el sistema de propagación de restricciones de la sección  [3.3.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.5) .

[62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_651) Esto utiliza la notación de cola punteada introducida en el ejercicio [2.20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.20).

[63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_654) En realidad, esta descripción denotes válida sólo para casos simples. El comportamiento real denotes más complejo. Examinaremosnoten las secciones [4.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.2)y [4.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.3).

[64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_655) Lisp-valuedebe usarse únicamente para realizar una operación, noSe proporciona en el lenguaje de consulta. En particular, no se debe utilizar para probar la igualdad (ya que para eso está diseñada la coincidencia en el lenguaje de consulta) o la desigualdad (ya que eso se puede hacer con la misma regla que se muestra a continuación).

[65](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_658) Tenga en cuenta que no necesitamossamepara que dos cosas sean iguales: simplemente usamos la misma variable de patrón para cada una; en efecto, tenemos una cosa en lugar de dos cosas en primer lugar. Por ejemplo, vea?townen lalives-neary?middle-manageren lawheela continuación.Samees útil cuando queremos forzar que dos cosas sean diferentes, como?person-1y?person-2en lalives-near. Aunque usar la misma variable de patrón en dos partes de una consulta obliga a que aparezca el mismo valor en ambos lugares, usar diferentes variables de patrón no obliga a que aparezcan diferentes valores. (Los valores asignados a diferentes variables de patrón pueden ser iguales o diferentes).

[66](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_659) También permitiremos reglas sin cuerpos, como enlo mismo , e interpretaremos dicha regla en el sentido de que la conclusión de la regla se satisface para cualquier valor de las variables.

[67](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_670) Debido a que la comparación suele ser muy costosa,Nos gustaría evitar aplicar el comparador completo a cada elemento de la base de datos. Esto se suele solucionar dividiendo el proceso en un comparador rápido y grueso y el comparador final. El comparador grueso filtra la base de datos para producir un pequeño conjunto de candidatos para el comparador final. Con cuidado, podemos organizar nuestra base de datos de modo que parte del trabajo del comparador grueso se pueda realizar cuando se construye la base de datos en lugar deLuego, cuando queremos seleccionar a los candidatos, esto se denomina *indexar* la base de datos. Existe una vasta tecnología basada en esquemas de indexación de bases de datos. Nuestra implementación, descrita en la sección  [4.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.4) , contiene una forma simple de tal optimización.

[68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_672) Pero este tipo de explosión exponencial no es común enlasconsultas porque las condiciones agregadas tienden a reducir en lugar de expandir el número de cuadros producidos.

[Existe una amplia literatura sobre sistemas de gestión de bases de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_673) que aborda cómo gestionar consultas complejas de manera eficiente.

[70](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_674) Existe una diferencia sutil entre esta implementación de filtro denoty el significado habitual denoten lógica matemática. Consulte la sección [4.4.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.4.3).

En la comparación de patrones unilateral, todas las ecuaciones que contienen variables de patrón son explícitas y ya están resueltas para la incógnita (la variable de patrón) [.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_676)

[72](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_677) Otra forma de pensar en la unificación es que genera el patrón más general que es una especialización de los dos patrones de entrada. Es decir, la unificación de(?xa)y((b ?y) ?z)es((b ?y) a), y la unificación de(?xa ?y)y(?y ?za), analizada anteriormente, es(aaa). Para nuestra implementación, es más conveniente pensar en el resultado de la unificación como un marco en lugar de un patrón.

[73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_680) Dado que la unificación es unaSi generalizamos el emparejamiento, podríamos simplificar el sistema utilizando el unificador para producir ambos flujos. Sin embargo, si tratamos el caso fácil con el emparejador simple, veremos cómo el emparejamiento (en contraposición a la unificación completa) puede ser útil por sí mismo.

[La razón por la](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_682) que utilizamos secuencias (en lugar de listas) de cuadros es queLa aplicación recursiva de reglas puede generar una cantidad infinita de valores que satisfacen una consulta. La evaluación diferida incorporada en los flujos es crucial en este caso: el sistema imprimirá las respuestas una por una a medida que se generen, independientemente de si hay una cantidad finita o infinita de respuestas.

[75](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_683) Que un método particular de inferencia sea legítimo no es una afirmación trivial. Hay que demostrar que si se parte de premisas verdaderas, sólo se pueden derivar conclusiones verdaderas. El método de inferencia representado por las aplicaciones de reglas es*modus ponens* , el método familiar de inferencia que dice que si *A* es verdadero y *A implica que B* es verdadero, entonces podemos concluir que *B* es verdadero.

[76](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_684) Debemos matizar esta afirmación aceptando que, al hablar de la ``inferencia'' realizada por un programa lógico, suponemos que el cálculo termina. Desafortunadamente, incluso esta afirmación matizada es falsa para nuestra implementación del lenguaje de consulta (y también falsa para programas en Prolog y la mayoría de los demás lenguajes de programación lógica actuales) debido a nuestro uso denotylisp-value. Como describiremos a continuación, elnotimplementado en el lenguaje de consulta no siempre es consistente con elnotde la lógica matemática, ylisp-valueintroduce complicaciones adicionales. Podríamos implementar un lenguaje consistente con la lógica matemática simplemente eliminandonotylisp-valuedel lenguaje y aceptando escribir programas utilizando solo consultas simples,and, andor. Sin embargo, esto restringiría en gran medida el poder expresivo del lenguaje. Una de las principales preocupaciones de la investigación en programación lógica es encontrar formas de lograr una mayor consistencia con la lógica matemática sin sacrificar indebidamente el poder expresivo.

[77](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_686) No se trata de un problema de lógica, sino de interpretación procedimental de la lógica proporcionada por nuestro intérprete. Podríamos escribir un intérprete que no cayera en un bucle aquí. Por ejemplo, podríamos enumerar todas las pruebas derivables de nuestras afirmaciones y nuestras reglas en un orden de amplitud en lugar de en profundidad. Sin embargo, un sistema de este tipo hace que sea más difícil aprovechar el orden de las deducciones en nuestros programas. Un intento de incorporar un control sofisticado en un programa de este tipo se describe endeKleer et al. 1977. Otra técnica, que no conduce a problemas de control tan graves, es introducir conocimientos especiales, como detectores para tipos particulares de bucles (ejercicio  [4.67](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.67) ). Sin embargo, no puede haber un esquema general para evitar de manera confiable que un sistema recorra caminos infinitos al realizar deducciones. Imaginemos una regla diabólica de la forma ``Para demostrar que *P* ( *x* ) es verdadera, demuestre que *P* ( *f* ( *x* )) es verdadera'', para alguna función *f* elegida adecuadamente .

[78](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_688) Considere la consulta(not (baseball-fan (Bitdiddle Ben))). El sistema encuentra que(baseball-fan (Bitdiddle Ben))no está en la base de datos, por lo que el marco vacío no satisface el patrón y no se filtra del flujo inicial de marcos. El resultado de la consulta es, por lo tanto, el marco vacío, que se utiliza para crear una instancia de la consulta de entrada para producir(not (baseball-fan (Bitdiddle Ben))).

[79](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_689) Una discusión y justificación de esteEl tratamiento del no se puede encontrar en el artículo de Clark (1978).

[80](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_700) En general, unificar?ycon una expresión que involucre?y requeriría que seamos capaces de encontrar un punto fijo de la ecuación ?y = < *expresión que involucra ?y* >. A veces es posible formar sintácticamente una expresión que parece ser la solución. Por ejemplo, ?y   =   (f ?y) parece tener el punto fijo (f (f (f ... ))) , que podemos producir comenzando con la expresión (f ?y) y sustituyendo repetidamente (f ?y) por ?y . Desafortunadamente, no todas las ecuaciones de este tipo tienen un punto fijo significativo. Los problemas que surgen aquí son similares a los problemas de manipulaciónSerie infinita en matemáticas. Por ejemplo, sabemos que 2 es la solución de la ecuación *y* = 1 + *y* /2. Si comenzamos con la expresión 1 + *y* /2 y sustituimos repetidamente 1 + *y* /2 por *y* obtenemos



Lo que conduce a



Sin embargo, si intentamos la misma manipulación comenzando con la observación de que - 1 es la solución de la ecuación *y* = 1 + 2 *y* , obtenemos



Lo que conduce a



Aunque las manipulaciones formales utilizadas para derivar estas dos ecuaciones son idénticas, el primer resultado es una afirmación válida sobre las series infinitas, pero el segundo no. De manera similar, para nuestros resultados de unificación, razonar con una expresión construida sintácticamente de manera arbitraria puede llevar a errores.

[81](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_702) La mayoría de los sistemas Lisp brindan al usuario la capacidad de modificar el procedimiento de lecturaordinariopara realizar dichas transformaciones definiendo*caracteres de la macro del lector* . Las expresiones entre comillas ya se manejan de esta manera: el lector traduce automáticamente 'expresión en (expresión entre comillas) antes de que el evaluador la vea. Podríamos disponer que ?expresión se transforme en (? expresión) de la misma manera; sin embargo, para mayor claridad hemos incluido aquí el procedimiento de transformación explícitamente.

Expandir-signo-de-interrogación y contraer-signo-de-interrogación utilizan varios procedimientos con cadena en sus nombres. Estos son primitivos de Scheme.

[**Capítulo 5**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_5)

[**Computación con máquinas de registro**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_5)

|  |
| --- |
| Mi objetivo es demostrar que la máquina celestial no es una especie de ser divino y vivo, sino una especie de mecanismo de relojería (y quien cree que un reloj tiene alma atribuye la gloria del fabricante a la obra), en la medida en que casi todos los múltiples movimientos son causados ​​por una fuerza muy simple y material, así como todos los movimientos del reloj son causados ​​por un solo peso.  Johannes Kepler (carta a Herwart von Hohenburg, 1605) |

Comenzamos este libro estudiando procesos y describiéndolos en términos de procedimientos escritos en Lisp. Para explicar el significado de estos procedimientos, utilizamos una sucesión de modelos de evaluación: el modelo de sustitución del capítulo 1, el modelo de entorno del capítulo 3 y el evaluador metacircular del capítulo 4. Nuestro examen del evaluador metacircular, en particular, disipó gran parte del misterio de cómo se interpretan los lenguajes similares a Lisp. Pero incluso el evaluador metacircular deja importantes preguntas sin respuesta, porque no logra dilucidar los mecanismos de control en un sistema Lisp. Por ejemplo, el evaluador no explica cómo la evaluación de una subexpresión logra devolver un valor a la expresión que usa este valor, ni tampoco explica cómo algunos procedimientos recursivos generan procesos iterativos (es decir, se evalúan utilizando un espacio constante) mientras que otros procedimientos recursivos generan procesos recursivos. Estas preguntas siguen sin respuesta porque el evaluador metacircular es en sí mismo un programa Lisp y, por lo tanto, hereda la estructura de control del sistema Lisp subyacente. Para proporcionar una descripción más completa de la estructura de control del evaluador Lisp, debemos trabajar en un nivel más primitivo que el propio Lisp.

En este capítulo describiremos los procesos en términos del funcionamiento paso a paso de una computadora tradicional. Una computadora de este tipo, o*máquina de registro* , ejecuta secuencialmente *instrucciones* que manipulan el contenido de un conjunto fijo de elementos de almacenamiento llamados*registros* . Una instrucción típica de máquina de registros aplica una operación primitiva al contenido de algunos registros y asigna el resultado a otro registro. Nuestras descripciones de procesos ejecutados por máquinas de registros se parecerán mucho a programas en "lenguaje de máquina" para computadoras tradicionales. Sin embargo, en lugar de enfocarnos en el lenguaje de máquina de cualquier computadora en particular, examinaremos varios procedimientos Lisp y diseñaremos una máquina de registros específica para ejecutar cada procedimiento. Por lo tanto, abordaremos nuestra tarea desde la perspectiva de un arquitecto de hardware en lugar de la de un programador de computadoras en lenguaje de máquina. Al diseñar máquinas de registros, desarrollaremos mecanismos para implementar construcciones de programación importantes como la recursión. También presentaremos un lenguaje para describir diseños para máquinas de registros. En la sección  [5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2) implementaremos un programa Lisp que usa estas descripciones para simular las máquinas que diseñamos.

La mayoría de las operaciones primitivas de nuestras máquinas de registro son muy simples. Por ejemplo, una operación podría sumar los números obtenidos de dos registros, produciendo un resultado que se almacenará en un tercer registro. Tal operación puede ser realizada por hardware de fácil descripción. Sin embargo, para tratar con la estructura de lista, también utilizaremos las operaciones de memoria car , cdr y cons , que requieren un mecanismo elaborado de asignación de almacenamiento. En la sección  [5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.3) estudiamos su implementación en términos de operaciones más elementales.

En la sección  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4) , después de haber acumulado experiencia formulando procedimientos simples como máquinas de registros, diseñaremos una máquina que lleve a cabo el algoritmo descrito por el evaluador metacircular de la sección  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) . Esto llenará el vacío en nuestra comprensión de cómo se interpretan las expresiones de Scheme, al proporcionar un modelo explícito para los mecanismos de control en el evaluador. En la sección  [5.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5) estudiaremos un compilador simple que traduce programas de Scheme en secuencias de instrucciones que se pueden ejecutar directamente con los registros y operaciones de la máquina de registros del evaluador.

[**5.1 Diseño de máquinas de registro**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_5.1)

Para diseñar una máquina de registros, debemos diseñar sus *rutas de datos* (registros y operaciones) y el *controlador* que secuencia estas operaciones. Para ilustrar el diseño de una máquina de registros simple, examinemos el algoritmo de Euclides, que se utiliza para calcularel máximo común divisor (MCD) de dos números enteros. Como vimos enSección  [1.2.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.5) , el algoritmo de Euclides puede llevarse a cabo mediante un proceso iterativo, como se especifica en el siguiente procedimiento:

(define (mcd a b)   
  (si (= b 0)   
      a   
      (mcd b (resto a b))))

Una máquina que lleve a cabo este algoritmo debe llevar un registro de dos números, *a* y *b* , así que supongamos que estos números están almacenados en dos registros con esos nombres. Las operaciones básicas requeridas son comprobar si el contenido del registro b es cero y calcular el resto del contenido del registro a dividido por el contenido del registro b . La operación de resto es un proceso complejo, pero supongamos por el momento que tenemos un dispositivo primitivo que calcula restos. En cada ciclo del algoritmo MCD, el contenido del registro a debe reemplazarse por el contenido del registro b , y el contenido de b debe reemplazarse por el resto del contenido anterior de a dividido por el contenido anterior de b . Sería conveniente si estos reemplazos pudieran hacerse simultáneamente, pero en nuestro modelo de máquinas de registros asumiremos que solo a un registro se le puede asignar un nuevo valor en cada paso. Para realizar los reemplazos, nuestra máquina utilizará un tercer registro "temporal", al que llamamos t . (Primero se colocará el resto en t , luego el contenido de b se colocará en a , y finalmente el resto almacenado en t se colocará en b .)

Podemos ilustrar los registros y operaciones requeridas para esta máquina usando el diagrama de ruta de datos que se muestra en la figura  [5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.1) . En este diagrama, los registros ( a , b y t ) están representados por rectángulos. Cada forma de asignar un valor a un registro está indicada por una flecha con una X detrás de la cabeza, apuntando desde la fuente de datos al registro. Podemos pensar en la X como un botón que, cuando se presiona, permite que el valor en la fuente "fluya" hacia el registro designado. La etiqueta junto a cada botón es el nombre que usaremos para referirnos al botón. Los nombres son arbitrarios y se pueden elegir para que tengan un valor mnemónico (por ejemplo, a<-b denota presionar el botón que asigna el contenido del registro b al registro a ). La fuente de datos para un registro puede ser otro registro (como en la asignación a<-b ), un resultado de operación (como en la asignación t<-r ) o una constante (un valor incorporado que no se puede cambiar, representado en un diagrama de ruta de datos por un triángulo que contiene la constante).

Una operación que calcula un valor a partir de constantes y el contenido de los registros se representa en un diagrama de ruta de datos mediante un trapezoide que contiene un nombre para la operación. Por ejemplo, el cuadro marcado rem en la figura  [5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.1) representa una operación que calcula el resto del contenido de los registros a y b a los que está asociada. Las flechas (sin botones) apuntan desde los registros de entrada y las constantes hasta el cuadro, y las flechas conectan el valor de salida de la operación con los registros. Una prueba se representa mediante un círculo que contiene un nombre para la prueba. Por ejemplo, nuestra máquina MCD tiene una operación que prueba si el contenido del registro b es cero. Una prueba también tiene flechas desde su entradaregistros y constantes, pero no tiene flechas de salida; su valor lo utiliza el controlador en lugar de las rutas de datos. En general, el diagrama de rutas de datos muestra los registros y las operaciones que se requieren para la máquina y cómo deben conectarse. Si vemos las flechas como cables y los botones X como interruptores, el diagrama de rutas de datos es muy parecido al diagrama de cableado de una máquina que podría construirse a partir de componentes eléctricos.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 5.1:**   Rutas de datos para una máquina GCD. |

Para que las rutas de datos calculen realmente los MCD, los botones deben pulsarse en la secuencia correcta. Describiremos esta secuencia en términos de un diagrama de controlador, como se ilustra en la figura  [5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.2) . Los elementos del diagrama de controlador indican cómo deben operarse los componentes de la ruta de datos. Los cuadros rectangulares en el diagrama de controlador identifican los botones de la ruta de datos que deben pulsarse, y las flechas describen la secuencia de un paso al siguiente. El diamante en el diagrama representa una decisión. Se seguirá una de las dos flechas de secuencia, dependiendo del valor de la prueba de la ruta de datos identificada en el diamante. Podemos interpretar el controlador en términos de una analogía física: piense en el diagrama como un laberinto en el que rueda una canica. Cuando la canica rueda hacia una caja, pulsa el botón de la ruta de datos que lleva el nombre de la caja. Cuando la canica rueda hacia un nodo de decisión (como la prueba para b = 0), abandona el nodo en la ruta determinada por el resultado de la prueba indicada. En conjunto, las rutas de datos y el controlador describen completamente una máquina para calcular MCD. Ponemos en marcha el controlador (la canica que rueda) en el lugar marcado como start , después de colocar números en los registros a y b . Cuando el controlador llegue a done , encontraremos el valor del MCD en el registro a .

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 5.2:**   Controlador para una máquina GCD. |

**Ejercicio 5.1.**  Diseñe una máquina de registro para calcular factoriales utilizando el algoritmo iterativo especificado por el siguiente procedimiento. Dibuje diagramas de ruta de datos y de controlador para esta máquina.

(define (factorial n)   
  (define (iter producto contador)   
    (si (> contador n)   
        producto   
        (iter (\* contador producto)   
              (+ contador 1))))   
  (iter 1 1))

**[5.1.1 Un lenguaje para describir máquinas de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.1.1)**

Los diagramas de ruta de datos y de controlador son adecuados para representar máquinas simples como GCD, pero son difíciles de manejar para describir máquinas grandes como un intérprete de Lisp. Para que sea posible trabajar con máquinas complejas, crearemos un lenguaje que presente, en forma de texto, toda la información proporcionada por los diagramas de ruta de datos y de controlador. Comenzaremos con una notación que refleje directamente los diagramas.

Definimos las rutas de datos de una máquina describiendo los registros y las operaciones. Para describir un registro, le damos un nombre y especificamos los botones que controlan la asignación a él. Damos un nombre a cada uno de estos botones y especificamos la fuente de los datos que ingresan al registro bajo el control del botón. (La fuente es un registro, una constante o una operación). Para describir una operación, le damos un nombre y especificamos sus entradas (registros o constantes).

Definimos el controlador de una máquina como una secuencia de*instrucciones* junto con*Etiquetas* que identifican *los puntos de entrada* en la secuencia. Una instrucción es una de las siguientes:

* El nombre de un botón de ruta de datos que se debe presionar para asignar un valor a un registro. (Esto corresponde a un cuadro en el diagrama del controlador).
* Una instrucción de prueba que realiza una prueba específica.
* Una bifurcación condicional ( instrucción de bifurcación ) a una ubicación indicada por una etiqueta del controlador, en función del resultado de la prueba anterior. (La prueba y la bifurcación juntas corresponden a un rombo en el diagrama del controlador). Si la prueba es falsa, el controlador debe continuar con la siguiente instrucción en la secuencia. De lo contrario, el controlador debe continuar con la instrucción después de la etiqueta.
* Una rama incondicional ( instrucción goto ) que nombra una etiqueta de controlador en la que continuar la ejecución.

La máquina comienza al principio de la secuencia de instrucciones del controlador y se detiene cuando la ejecución llega al final de la secuencia. Excepto cuando una rama cambia el flujo de control, las instrucciones se ejecutan en el orden en que están enumeradas.

|  |
| --- |
| (rutas-de-datos   (registros    ((nombre a)     (botones ((nombre a<-b) (fuente (registro b)))))    ((nombre b)     (botones ((nombre b<-t) (fuente (registro t)))))    ((nombre t)     (botones ((nombre t<-r) (fuente (operación rem))))))    (operaciones    ((nombre rem)     (entradas (registro a) (registro b)))    ((nombre =)     (entradas (registro b) (constante 0)))))   (controlador   prueba-b                            *; etiqueta*    (prueba =)                        *; prueba*    (rama (etiqueta gcd-done))       *; rama condicional*    (t<-r)                          *; pulsar botón*    (a<-b)                          *; pulsar botón*    (b<-t)                          *; pulsar botón*    (goto (etiqueta prueba-b))           *; rama incondicional*  gcd-done)                         *; etiqueta* |
|  |
| **Figura 5.3:**   Una especificación de la máquina GCD. |

La figura  [5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.3) muestra la máquina GCD descrita de esta manera. Este ejemplo solo insinúa la generalidad de estas descripciones, ya que la máquina GCD es un caso muy simple: cada registro tiene solo un botón, y cada botón y prueba se usa solo una vez en el controlador.

Lamentablemente, resulta difícil leer una descripción de este tipo. Para comprender las instrucciones del controlador debemos consultar constantemente las definiciones de los nombres de los botones y de las operaciones, y para comprender lo que hacen los botones es posible que tengamos que consultar las definiciones de los nombres de las operaciones. Por lo tanto, transformaremos nuestra notación para combinar la información de las descripciones de la ruta de datos y del controlador de modo que podamos verla toda en conjunto.

Para obtener esta forma de descripción, reemplazaremos los nombres arbitrarios de los botones y operaciones por las definiciones de su comportamiento. Es decir, en lugar de decir (en el controlador) ``Pulsar botón t<-r '' y decir por separado (en las rutas de datos) ``El botón t<-r asigna el valor de la operación rem al registro t '' y ``Las entradas de la operación rem son los contenidos de los registrosa y b ,'' diremos (en el controlador) ``Presione el botón que asigna al registro t el valor de la operación rem sobre el contenido de los registros a y b .'' De manera similar, en lugar de decir (en el controlador) ``Realice la prueba = '' y decir por separado (en las rutas de datos) ``La prueba = opera sobre el contenido del registro b y la constante 0,'' diremos ``Realice la prueba = sobre elContenido del registro b y la constante 0.'' Omitiremos la descripción de la ruta de datos, dejando solo la secuencia del controlador. Por lo tanto, la máquina GCD se describe de la siguiente manera:

(controlador   
 prueba-b   
   (prueba (op =) (reg b) (const 0))   
   (rama (etiqueta mcd-hecho))   
   (asignar t (op rem) (reg a) (reg b))   
   (asignar a (reg b))   
   (asignar b (reg t))   
   (ir a (etiqueta prueba-b))   
 mcd-hecho)

Esta forma de descripción es más fácil de leer que la ilustrada en la figura  [5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.3) , pero también tiene desventajas:

* Es más detallado para máquinas grandes, porque se repiten descripciones completas de los elementos de la ruta de datos siempre que se mencionan los elementos en la secuencia de instrucciones del controlador. (Esto no es un problema en el ejemplo de GCD, porque cada operación y botón se usa solo una vez). Además, repetir las descripciones de la ruta de datos oscurece la estructura real de la ruta de datos de la máquina; no es obvio para una máquina grande cuántos registros, operaciones y botones hay y cómo están interconectados.
* Como las instrucciones del controlador en una definición de máquina parecen expresiones de Lisp, es fácil olvidar que no son expresiones de Lisp arbitrarias. Pueden indicar únicamente operaciones de máquina legales. Por ejemplo, las operaciones pueden operar directamente solo sobre constantes y el contenido de registros, no sobre los resultados de otras operaciones.

A pesar de estas desventajas, utilizaremos este lenguaje de registro-máquina a lo largo de este capítulo, porque nos interesará más comprender los controladores que los elementos y conexiones en las rutas de datos. Sin embargo, debemos tener en cuenta que el diseño de rutas de datos es crucial para diseñar máquinas reales.

**Ejercicio 5.2.**  Utilice el lenguaje de máquina de registros para describir la máquina factorial iterativa del ejercicio  [5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.1) .

**[Comportamiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_714)**

Modifiquemos la máquina de MCD para que podamos escribir los números cuyo MCD queremos y obtener la respuesta impresa en nuestra terminal. No analizaremos cómo hacer una máquina que pueda leer e imprimir, pero asumiremos (como hacemos cuando usamos lectura y visualización en Scheme) que están disponibles como operaciones primitivas. [1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_715)

Read es como las operaciones que hemos estado usando en el sentido de que produce un valor que se puede almacenar en un registro. Pero read no toma entradas de ningún registro; su valor depende de algo que sucede fuera de las partes de la máquina que estamos diseñando. Permitiremos que las operaciones de nuestra máquina tengan ese comportamiento y, por lo tanto, dibujaremos y anotaremos el uso de read tal como lo hacemos con cualquier otra operación que calcule un valor.

Print , por otro lado, difiere de las operaciones que hemos estado usando de una manera fundamental: no produce un valor de salida para almacenar en un registro. Aunque tiene un efecto, este efecto no está en una parte de la máquina que estamos diseñando. Nos referiremos a este tipo de operación como una *acción* . Representaremos una acción en un diagrama de ruta de datos de la misma manera que representamos una operación que calcula un valor: como un trapezoide que contiene el nombre de la acción. Las flechas apuntan al cuadro de acción desde cualquier entrada (registros o constantes). También asociamos un botón con la acción. Al presionar el botón se realiza la acción. Para hacer que un controlador presione una acciónPara el botón utilizamos un nuevo tipo de instrucción llamada perform . De esta forma, la acción de imprimir el contenido del registro a se representa en una secuencia de controlador mediante la instrucción

(realizar (op imprimir) (reg a))

La figura  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.4) muestra las rutas de datos y el controlador de la nueva máquina MCD. En lugar de detener la máquina después de imprimir la respuesta, hemos hecho que vuelva a empezar, de modo que lea repetidamente un par de números, calcule su MCD e imprima el resultado. Esta estructura es como los bucles de controlador que utilizamos en los intérpretes del capítulo 4.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto.   (controlador    gcd-loop      (asignar a (op leer))      (asignar b (op leer))    prueba-b      (prueba (op =) (reg b) (const 0))      (rama (etiqueta gcd-done))      (asignar t (op rem) (reg a) (reg b))      (asignar a (reg b))      (asignar b (reg t))      (goto (etiqueta prueba-b))    gcd-done      (ejecutar (op imprimir) (reg a))      (goto (etiqueta gcd-loop))) |
|  |
| **Figura 5.4:**   Una máquina GCD que lee entradas e imprime resultados. |

**[5.1.2 Abstracción en el diseño de máquinas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.1.2)**

A menudo definiremos una máquina para incluir operaciones "primitivas" que en realidad son muy complejas. Por ejemplo, en las secciones  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4) y [5.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5) trataremos las manipulaciones del entorno de Scheme como primitivas. Esta abstracción es valiosa porque nos permite ignorar los detalles de las partes de una máquina para que podamos concentrarnos en otros aspectos del diseño. Sin embargo, el hecho de que hayamos ocultado mucha complejidad no significa que el diseño de una máquina sea poco realista. Siempre podemos reemplazar las "primitivas" complejas por operaciones primitivas más simples.

Consideremos la máquina MCD. La máquina tiene una instrucción que calcula el resto del contenido de los registros a y b y asigna el resultado al registro t . Si queremos construir la máquina MCD sin utilizar una operación de resto primitiva, debemos especificar cómo calcular los restos en términos de operaciones más simples, como la resta. De hecho, podemos escribir un procedimiento de Scheme que encuentre los restos de esta manera:

(define (resto n d)   
  (si (< n d)   
      n   
      (resto (- n d) d)))

De esta manera, podemos reemplazar la operación de resto en las rutas de datos de la máquina MCD por una operación de resta y una prueba de comparación. La figura  [5.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.5) muestra las rutas de datos y el controlador para la máquina elaborada. La instrucción

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 5.5:**   Rutas de datos y controlador para la máquina GCD elaborada. |

(asignar t (op rem) (reg a) (reg b))

en la definición del controlador GCD se reemplaza por una secuencia de instrucciones que contiene un bucle, como se muestra en la figura  [5.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.6) .

|  |
| --- |
| (controlador   prueba-b     (prueba (op =) (reg b) (const 0))     (rama (etiqueta mcd-hecho))     (asignar t (reg a))   rem-bucle     (prueba (op <) (reg t) (reg b))     (rama (etiqueta rem-bucle))     (asignar t (op -) (reg t) (reg b))     (goto (etiqueta rem-bucle))   rem-hecho     (asignar a (reg b))     (asignar b (reg t))     (goto (etiqueta prueba-b))   mcd-hecho) |
|  |
| **Figura 5.6:**   Secuencia de instrucciones del controlador para la máquina GCD en la figura  [5.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.5) . |

**Ejercicio 5.3.**  Diseñe una máquina para calcular raíces cuadradas utilizando el método de Newton, como se describe en la sección  [1.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.1.7) :

(define (sqrt x)   
  (define (¿suficientemente bueno? conjetura)   
    (< (abs (- (conjetura al cuadrado) x)) 0.001))   
  (define (mejora conjetura)   
    (conjetura promedio (/ x conjetura)))   
  (define (sqrt-iter conjetura)   
    (si (¿suficientemente bueno? conjetura)   
        conjetura   
        (sqrt-iter (mejora conjetura))))   
  (sqrt-iter 1.0))

Comience por suponer que las operaciones ¿suficientemente bueno? y ¿mejorar ? están disponibles como primitivas. Luego, muestre cómo expandirlas en términos de operaciones aritméticas. Describa cada versión del diseño de la máquina sqrt dibujando un diagrama de ruta de datos y escribiendo una definición de controlador en el lenguaje de la máquina de registros.

**[5.1.3 Subrutinas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.1.3)**

Al diseñar una máquina para realizar un cálculo, a menudo preferiríamos disponer que los componentes sean compartidos por diferentes partes del cálculo en lugar de duplicar los componentes. Considere una máquina que incluye dos cálculos de MCD: uno que encuentra el MCD de los contenidos de los registros a y b y otro que encuentra el MCD de los contenidos de los registros c y d . Podríamos comenzar asumiendo que tenemos una operación de MCD primitiva , luego expandir las dos instancias de MCD en términos de operaciones más primitivas. La Figura  [5.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.7) muestra solo las porciones de MCD de las rutas de datos de la máquina resultante, sin mostrar cómo se conectan con el resto de la máquina. La figura también muestra las porciones correspondientes de la secuencia del controlador de la máquina.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto.  mcd-1   (prueba (op =) (reg b) (const 0))   (rama (etiqueta después de mcd-1))   (asignar t (op rem) (reg a) (reg b))   (asignar a (reg b))   (asignar b (reg t))   (goto (etiqueta mcd-1))  después de mcd-1       mcd-2   (prueba (op =) (reg d) (const 0))   (rama (etiqueta después de mcd-2))   (asignar s (op rem) (reg c) (reg d))   (asignar c (reg d))   (asignar d (reg s))   (goto (etiqueta mcd-2))  después de mcd-2 |
|  |
| **Figura 5.7:**   Partes de las rutas de datos y la secuencia del controlador para una máquina con dos cálculos de MCD. |

Esta máquina tiene dos cajas de operaciones de resto y dos cajas para probar la igualdad. Si los componentes duplicados son complicados, como lo es la caja de resto, esta no será una forma económica de construir la máquina. Podemos evitar duplicar los componentes de la ruta de datos utilizando los mismos componentes para ambos cálculos de MCD, siempre que al hacerlo no afecte el resto del cálculo de la máquina más grande. Si los valores en los registros a y b no son necesarios para cuando el controlador llega a mcd-2 (o si estos valores se pueden mover a otros registros para su custodia), podemos cambiar la máquina para que utilice los registros a y b , en lugar de los registros c y d , para calcular el segundo MCD así como el primero. Si hacemos esto, obtenemos la secuencia del controlador que se muestra en la figura  [5.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.8) .

Hemos eliminado los componentes duplicados de la ruta de datos (de modo que las rutas de datos vuelven a ser como en la figura  [5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.1) ), pero el controlador ahora tiene dos secuencias GCD que difieren solo en sus etiquetas de punto de entrada. Sería mejor reemplazar estas dos secuencias por ramificaciones a una sola secuencia - una *subrutina* gcd - al final de la cual volvemos a ramificar al lugar correcto en la secuencia de instrucciones principal. Podemos lograr esto de la siguiente manera: Antes de ramificar a gcd , colocamos un valor distintivo (como 0 o 1) en un registro especial,continue . Al final de la subrutina gcd volvemos a after-gcd-1 o a after-gcd-2 , dependiendo del valor del registro continue . La figura  [5.9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.9) muestra la parte relevante de la secuencia del controlador resultante, que incluye solo una copia de las instrucciones gcd .

|  |
| --- |
| mcd-1   (prueba (op =) (reg b) (const 0))   (rama (etiqueta después de mcd-1))   (asignar t (op rem) (reg a) (reg b))   (asignar a (reg b))   (asignar b (reg t))   (goto (etiqueta mcd-1))  después de mcd-1     mcd-2   (prueba (op =) (reg b) (const 0))   (rama (etiqueta después de mcd-2))   (asignar t (op rem) (reg a) (reg b))   (asignar a (reg b))   (asignar b (reg t))   (goto (etiqueta mcd-2))  después de mcd-2 |
|  |
| **Figura 5.8:**   Partes de la secuencia del controlador para una máquina que utiliza los mismos componentes de ruta de datos para dos cálculos de MCD diferentes. |

|  |
| --- |
| gcd   (test (op =) (reg b) (const 0))   (branch (label gcd-done))   (assign t (op rem) (reg a) (reg b))   (assign a (reg b))   (assign b (reg t))   (goto (label gcd))  gcd-done   (test (op =) (reg continue) (const 0))          (branch (label after-gcd-1))   (goto (label after-gcd-2))     *;; Antes de bifurcar a  gcd  desde el primer lugar donde*  *;; se necesita, colocamos 0 en el   registro continue*  (assign continue (const 0))   (goto (label gcd))  after-gcd-1     *;; Antes del segundo uso de  gcd , colocamos 1 en el   registro continue*  (assign continue (const 1))   (goto (label gcd))  after-gcd-2 |
|  |
| **Figura 5.9:**   Uso de un registro  continuo para evitar la secuencia de controlador duplicada en la figura [5.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.8) . |

|  |
| --- |
| gcd   (test (op =) (reg b) (const 0))   (branch (label gcd-done))   (assign t (op rem) (reg a) (reg b))   (assign a (reg b))   (assign b (reg t))   (goto (label gcd))  gcd-done   (goto (reg continue))      *;; Antes de llamar  a gcd , asignamos a  continue* *;; la etiqueta a la que   debe regresar gcd* .  (assign continue (label after-gcd-1))   (goto (label gcd))  after-gcd-1      *;; Aquí está la segunda llamada a  gcd , con una continuación diferente.*  (assign continue (label after-gcd-2))   (goto (label gcd))  after-gcd-2 |
|  |
| **Figura 5.10:**   La asignación de etiquetas al registro continuo simplifica y generaliza la estrategia mostrada en la figura  [5.9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.9) . |

Este es un enfoque razonable para manejar pequeños problemas, pero sería extraño si hubiera muchas instancias de cálculos GCD en la secuencia del controlador. Para decidir dónde continuar la ejecución después de la subrutina gcd , necesitaríamos pruebas en las rutas de datos e instrucciones de bifurcación en el controlador para todos los lugares que usan gcd . Un método más poderoso para implementar subrutinas es hacer que el registro continue contenga la etiqueta del punto de entrada en la secuencia del controlador en el que debe continuar la ejecución cuando finalice la subrutina. Implementar esta estrategia requiere un nuevo tipo de conexión entre las rutas de datos y el controlador de una máquina de registros: debe haber una forma de asignar a un registro una etiqueta en la secuencia del controlador de tal manera que este valor pueda obtenerse del registro y usarse para continuar la ejecución en el punto de entrada designado.

Para reflejar esta capacidad, ampliaremos la instrucción de asignación del lenguaje de máquina de registros para permitir que se asigne a un registro como valor una etiqueta de la secuencia del controlador (como un tipo especial de constante). También ampliaremos la instrucción goto para permitir que la ejecución continúe en el punto de entrada descrito por el contenido de un registro en lugar de solo en un punto de entrada descrito por una etiqueta de constante. Usando estas nuevas construcciones podemos terminar la subrutina gcd con una bifurcación a la ubicación almacenada en el registro de continuación . Esto conduce a la secuencia del controlador que se muestra en la figura  [5.10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.10) .

Una máquina con más de una subrutina podría usar múltiples registros de continuación (por ejemplo, gcd-continue , factorial-continue ) o podríamos hacer que todas las subrutinas compartan un único registro de continuación . Compartir es más económico, pero debemos tener cuidado si tenemos una subrutina ( sub1 ) que llama a otra subrutina ( sub2 ). A menos que sub1 guarde el contenido de continue en algún otro registro antes de configurar continue para la llamada a sub2 , sub1 no sabrá dónde ir cuando haya terminado. El mecanismo desarrollado en la siguiente sección para manejar la recursión también proporciona una mejor solución a este problema de llamadas de subrutinas anidadas.

**[5.1.4 Uso de una pila para implementar la recursión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.1.4)**

Con las ideas ilustradas hasta ahora, podemos implementar cualquier proceso iterativo especificando una máquina de registros que tenga un registro correspondiente a cada variable de estado del proceso. La máquina ejecuta repetidamente un bucle de controlador, cambiando el contenido de los registros, hasta que se satisface alguna condición de terminación. En cada punto de la secuencia del controlador, el estado de la máquina (que representa el estado del proceso iterativo) está completamente determinado por el contenido de los registros (los valores de las variables de estado).

Sin embargo, la implementación de procesos recursivos requiere un mecanismo adicional. Consideremos el siguiente método recursivo para calcular factoriales, que examinamos por primera vez en la sección  [1.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.1) :

(define (factorial n)   
  (si (= n 1)   
      1   
      (\* (factorial (- n 1)) n)))

Como vemos en el procedimiento, calcular *n* ! requiere calcular ( *n* - 1)!. Nuestra máquina MCD, modelada según el procedimiento

(define (mcd a b)   
  (si (= b 0)   
      a   
      (mcd b (resto a b))))

De manera similar, se tuvo que calcular otro MCD. Pero hay una diferencia importante entre el procedimiento de MCD , que reduce el cálculo original a un nuevo cálculo de MCD, y factorial , que requiere calcular otro factorial como un subproblema. En MCD, la respuesta al nuevo cálculo de MCD es la respuesta al problema original. Para calcular el próximo MCD, simplemente colocamos los nuevos argumentos en los registros de entrada de la máquina de MCD y reutilizamos las rutas de datos de la máquina ejecutando la misma secuencia de controlador. Cuando la máquina termina de resolver el problema final de MCD, ha completado todo el cálculo.

En el caso del factorial (o cualquier proceso recursivo) la respuesta al nuevo subproblema factorial no es la respuesta al problema original. El valor obtenido para ( *n* - 1)! debe ser multiplicado por *n* para obtener la respuesta final. Si tratamos de imitar el diseño del MCD y resolvemos el subproblema factorial decrementando el registro n y volviendo a ejecutar la máquina factorial, ya no tendremos disponible el antiguo valor de n por el cual multiplicar el resultado. Por lo tanto, necesitamos una segunda máquina factorial para trabajar en el subproblema. Este segundo cálculo factorial tiene en sí un subproblema factorial, que requiere una tercera máquina factorial, y así sucesivamente. Dado que cada máquina factorial contiene otra máquina factorial dentro de ella, la máquina total contiene un nido infinito de máquinas similares y, por lo tanto, no puede construirse a partir de un número fijo y finito de partes.

Sin embargo, podemos implementar el proceso factorial como una máquina de registros si podemos disponer el uso de los mismos componentes para cada instancia anidada de la máquina. Específicamente, la máquina que calcula *n* ! debe usar los mismos componentes para trabajar en el subproblema de calcular ( *n* - 1)!, en el subproblema para ( *n* - 2)!, y así sucesivamente. Esto es plausible porque, aunque el proceso factorial dicta que se necesita un número ilimitado de copias de la misma máquina para realizar un cálculo, solo una de estas copias necesita estar activa en un momento dado. Cuando la máquina encuentra un subproblema recursivo, puede suspender el trabajo en el problema principal, reutilizar las mismas partes físicas para trabajar en el subproblema y luego continuar el cálculo suspendido.

En el subproblema, el contenido de los registros será diferente al del problema principal (en este caso, el registro n se decrementa). Para poder continuar con el cálculo suspendido, la máquina debe guardar el contenido de los registros que se necesitarán después de que se resuelva el subproblema, de modo que se puedan restaurar para continuar con el cálculo suspendido. En el caso del factorial, guardaremos el valor anterior de n , para restaurarlo cuando terminemos de calcular el factorial del registro n decrementado . [2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_717)

Dado que no existe *un* límite a priori en la profundidad de las llamadas recursivas anidadas, es posible que necesitemos guardar una cantidad arbitraria de valores de registros. Estos valores deben restaurarse en el orden inverso al que se guardaron, ya que en un anidamiento de recursiones el último subproblema en ingresar es el primero en terminarse. Esto dicta el uso de una *pila* , o estructura de datos de "último en entrar, primero en salir", para guardar valores de registros. Podemos extender el lenguaje de máquina de registros para incluir una pila agregando dos tipos de instrucciones: Los valores se colocanen la pila mediante una instrucción de guardado y se recupera desde la pila mediante una instrucción de restauración . Después de que se haya guardado una secuencia de valores en la pila, una secuencia de restauraciones recuperará estos valores en orden inverso. [3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_718)

Con la ayuda de la pila, podemos reutilizar una única copia de las rutas de datos de la máquina factorial para cada subproblema factorial. Existe un problema de diseño similar al reutilizar la secuencia del controlador que opera las rutas de datos. Para volver a ejecutar el cálculo factorial, el controlador no puede simplemente volver al principio, como en un proceso iterativo, porque después de resolver el subproblema ( *n* - 1)! la máquina todavía debe multiplicar el resultado por *n* . El controlador debe suspender su cálculo de *n* !, resolver el subproblema ( *n* - 1)! y luego continuar su cálculo de *n* !. Esta visión del cálculo factorial sugiere el uso del mecanismo de subrutina descrito en la sección  [5.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1.3) , que hace que el controlador use unaregistro continue para pasar a la parte de la secuencia que resuelve un subproblema y luego continuar donde se quedó en el problema principal. De esta manera podemos hacer una subrutina factorial que regrese al punto de entrada almacenado en el registro continue . Alrededor de cada llamada de subrutina, guardamos y restauramos continue tal como lo hacemos con el registro n , ya que cada ``nivel'' del cálculo factorial usará el mismo registro continue . Es decir, la subrutina factorial debe poner un nuevo valor en continue cuando se llama a sí misma para un subproblema, pero necesitará el valor anterior para regresar al lugar que la llamó para resolver un subproblema.

La Figura  [5.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.11) muestra las rutas de datos y el controlador para una máquina que implementa el procedimiento factorial recursivo. La máquina tiene una pila y tres registros, llamados n , val y continue . Para simplificar el diagrama de la ruta de datos, no hemos nombrado los botones de asignación de registros, solo los botones de operación de la pila ( sc y sn para guardar registros, rc y rn para restaurar registros). Para operar la máquina, colocamos en el registro n el número cuyo factorial deseamos calcular e iniciamos la máquina. Cuando la máquina llega a fact-done , el cálculo finaliza y la respuesta se encontrará en el registro val . En la secuencia del controlador, n y continue se guardan antes de cada llamada recursiva y se restauran al regresar de la llamada. El regreso de una llamada se logra bifurcando a la ubicación almacenada en continue . Continue se inicializa cuando la máquina se inicia de modo que el último retorno irá a fact-done . El registro val , que contiene el resultado del cálculo factorial, no se guarda antes de la llamada recursiva, porque el contenido anterior de val no es útil después de que la subrutina retorna. Solo se necesita el nuevo valor, que es el valor producido por el subcálculo. Aunque en principio el cálculo factorial requiere una máquina infinita, la máquina en la figura  [5.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.11) es en realidad finita excepto por la pila, que es potencialmente ilimitada. Sin embargo, cualquier implementación física particular de una pila será de tamaño finito, y esto limitará la profundidad de las llamadas recursivas que puede manejar la máquina. Esta implementación de factorial ilustra la estrategia general para realizar algoritmos recursivos como máquinas de registros comunes aumentadas por pilas. Cuando se encuentra un subproblema recursivo, guardamos en la pila los registros cuyos valores actuales serán necesarios después de que se resuelva el subproblema, resolvemos el subproblema recursivo, luego restauramos los registros guardados y continuamos la ejecución del problema principal. El registro continue siempre debe guardarse. Si hay otros registros que se deben guardar depende de la máquina en particular, ya que no todos los cálculos recursivos necesitan los valores originales de los registros que se modifican durante la solución del subproblema (ver ejercicio  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.4) ).

**[Una doble recursión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_719)**

Examinemos un proceso recursivo más complejo, el cálculo recursivo en árbol de los números de Fibonacci, que presentamos en la sección  [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) :

(define (fib n)   
  (si (< n 2)   
      n   
      (+ (fib (- n 1)) (fib (- n 2)))))

Al igual que con factorial, podemos implementar el cálculo recursivo de Fibonacci como una máquina de registros con los registros n , val y continue . La máquina es más compleja que la de factorial, porque hay dos lugares en la secuencia del controlador donde necesitamos realizar llamadas recursivas: una para calcular Fib( *n* - 1) y otra para calcular Fib( *n* - 2). Para preparar cada una de estas llamadas, guardamos los registros cuyos valores se necesitarán más adelante, establecemos el registro n en el número cuyo Fib necesitamos calcular de forma recursiva ( *n* - 1 o *n* - 2 ), y asignamos a continue el punto de entrada en la secuencia principal al que regresar ( afterfib-n-1 o afterfib-n-2 , respectivamente). Luego vamos a fib-loop . Cuando regresamos de la llamada recursiva, la respuesta está en val . La Figura  [5.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.12) muestra la secuencia del controlador para esta máquina.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto.  (controlador     (asignar continuar (etiqueta fact-done))      *; establecer la dirección de retorno final*  fact-loop     (prueba (op =) (reg n) (const 1))     (rama (etiqueta base-case))     *;; Configurar para la llamada recursiva guardando  n  y  continue .*     *;; Configurar  continue  para que el cálculo continúe*     *;; en  after-fact  cuando la subrutina regrese.*    (guardar continue)     (guardar n)     (asignar n (op -) (reg n) (const 1))     (asignar continue (etiqueta after-fact))     (goto (etiqueta fact-loop))   after-fact     (restaurar n)     (restaurar continue)     (asignar val (op \*) (reg n) (reg val))    *;  ¡val  ahora contiene*  *n* ( *n* - 1)!     (goto (reg continue))                    *; regresar al*  caso base del llamador     (asignar val (const 1))                   *; caso base:* 1! = 1     (goto (reg continue))                    *; volver al*  hecho de que el llamador está hecho) |
|  |
| **Figura 5.11:**   Una máquina factorial recursiva. |

|  |
| --- |
| (controlador     (asignar continuar (etiqueta fib-hecho))   fib-loop     (prueba (op <) (reg n) (const 2))     (rama (etiqueta respuesta-inmediata))     *;; configurado para calcular  F i b ( n - 1)*    (guardar continuar)     (asignar continuar (etiqueta afterfib-n-1))     (guardar n)                            *; guardar el valor anterior de  n*    (asignar n (op -) (reg n) (const 1)) *; convertir  n  en  n - 1*    (goto (etiqueta fib-loop))             *; realizar llamada recursiva*  afterfib-n-1                          *; al regresar,  val  contiene  F i b ( n - 1)*    (restaurar n)     (restaurar continuar)     *;; configurado para calcular  F i b ( n - 2)*    (asignar n (op -) (reg n) (const 2)) (     guardar continuar)     (asignar continuar (etiqueta después de fib-n-2))     (guardar val)                          *; guardar  F i b ( n - 1)*    (goto (etiqueta fib-loop))   después de fib-n-2                          *; al regresar,  val  contiene  F i b ( n - 2)*    (asignar n (reg val))                *;  n  ahora contiene  F i b ( n - 2)*    (restaurar val)                       *;  val  ahora contiene  F i b ( n - 1)*    (restaurar continuar) *(*    asignar val                         *;   F i b ( n - 1) +   F i b ( n - 2)*            (op +) (reg val) (reg n))      (goto (reg continue))               *; regresar al llamador, la respuesta está en  val*  respuesta-inmediata     (asignar val (reg n))                *; caso base:   F i b ( n ) = n*    (ir a (reg continuar))   fib-hecho) |
|  |
| **Figura 5.12:**   Controlador de una máquina para calcular números de Fibonacci. |

**Ejercicio 5.4.**   Especifique las máquinas de registro que implementan cada uno de los siguientes procedimientos. Para cada máquina, escriba una secuencia de instrucciones del controlador y dibuje un diagrama que muestre las rutas de datos.

a. Exponenciación recursiva:

(define (expt b n)   
  (si (= n 0)   
      1   
      (\* b (expt b (- n 1)))))

b. Exponenciación iterativa:

(define (expt b n)   
  (define (expt-iter contador producto)   
    (si (= contador 0)   
        producto   
        (expt-iter (- contador 1) (\* b producto))))   
  (expt-iter n 1))

**Ejercicio 5.5.**   Simule manualmente las máquinas factorial y de Fibonacci, utilizando alguna entrada no trivial (requiriendo la ejecución de al menos una llamada recursiva). Muestre el contenido de la pila en cada punto significativo de la ejecución.

**Ejercicio 5.6.**   Ben Bitdiddle observa que la secuencia del controlador de la máquina de Fibonacci tiene una instrucción de guardado y una de restauración adicionales , que se pueden eliminar para hacer que la máquina sea más rápida. ¿Dónde están estas instrucciones?

**[5.1.5 Resumen de instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.1.5)**

Una instrucción de controlador en nuestro lenguaje de máquina de registros tiene una de las siguientes formas, donde cada < *entrada i* > es (reg < *nombre-de-registro* >) o (const < *valor-constante* >) .

Estas instrucciones se introdujeron en la sección  [5.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1.1) :

(asignar < *nombre-registro* > (reg < *nombre-registro* >))   
  
(asignar < *nombre-registro* > (const < *valor-constante* >))  
  
(asignar < *nombre-registro* > (op < *nombre-operación* >) < *entrada 1* >  ... < *entrada n* >)  
  
(realizar (op < *nombre-de-la-operación* >) < *entrada 1* >  ... < *entrada n* >)  
  
(prueba (op < *nombre-de-operación* >) < *entrada 1* >  ... < *entrada n* >)  
  
(rama (etiqueta < *nombre-etiqueta* >))  
  
(ir a (etiqueta < *nombre-etiqueta* >))

El uso de registros para almacenar etiquetas se introdujo en la sección  [5.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1.3) :

(asignar < *nombre-registro* > (etiqueta < *nombre-etiqueta* >))   
  
(ir a (reg < *nombre-registro* >))

Las instrucciones para utilizar la pila se introdujeron en la sección  [5.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1.4) :

(guardar *<nombre-registro>* )  
  
(restaurar < *nombre-registro* >)

El único tipo de < *valor-constante* > que hemos visto hasta ahora es un número, pero más adelante utilizaremos cadenas, símbolos y listas. Por ejemplo, (const "abc") es la cadena "abc" , (const abc) es el símbolo abc , (const (abc)) es la lista (abc) y (const ()) es la lista vacía.

[1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_715) Esta suposición oculta una gran parte de la complejidad. Normalmente, una gran parte de la implementación de un sistema Lisp se dedica a hacer que la lectura y la impresión funcionen.

[2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_717) Se podría argumentar que no necesitamos guardar elvalor n; después de disminuirlo y resolver el subproblema, simplemente podríamos incrementarlo para recuperar el valor anterior. Aunque esta estrategia funciona para factorial, no puede funcionar en general, ya que el valor anterior de un registro no siempre se puede calcular a partir del nuevo.

[3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_718) En la sección [5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.3)veremos cómo implementar una pila en términos de operaciones más primitivas.

[**5.2 Un simulador de máquina de registro**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_5.2)

Para comprender bien el diseño de las máquinas de registro, debemos probar las máquinas que diseñamos para ver si funcionan como se espera. Una forma de probar un diseño es simular manualmente el funcionamiento del controlador, como en el ejercicio  [5.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.5) . Pero esto es extremadamente tedioso para todas las máquinas, excepto para las más simples. En esta sección construimos un simulador para máquinas descritas en el lenguaje de máquina de registro. El simulador es un programa Scheme con cuatro procedimientos de interfaz. El primero utiliza una descripción de una máquina de registro para construir un modelo de la máquina (una estructura de datos cuyas partes corresponden a las partes de la máquina que se va a simular), y los otros tres nos permiten simular la máquina manipulando el modelo:

(make-machine < *register-names* > < *operations* > < *controller* >)  
construye y devuelve un modelo de la máquina con los registros, operaciones y controlador dados.

(set-register-contents! < *machine-model* > < *register-name* > < *value* >)  
almacena un valor en un registro simulado en la máquina dada.

(get-register-contents < *machine-model* > < *register-name* >)  
devuelve el contenido de un registro simulado en la máquina dada.

(start < *machine-model* >)  
simula la ejecución de la máquina dada, comenzando desde el principio de la secuencia del controlador y deteniéndose cuando llega al final de la secuencia.

Como ejemplo de cómo se utilizan estos procedimientos, podemos definir gcd-machine como un modelo de la máquina GCD de la sección  [5.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1.1) de la siguiente manera:

(define mcd-machine   
  (make-machine   
   '(a b t)   
   (list (list 'rem remainder) (list '= =))   
   '(test-b   
       (test (op =) (reg b) (const 0))   
       (branch (label mcd-done))   
       (assign t (op rem) (reg a) (reg b))   
       (assign a (reg b))   
       (assign b (reg t))   
       (goto (label test-b))   
     mcd-done)))

El primer argumento de make-machine es una lista de nombres de registros. El siguiente argumento es una tabla (una lista de listas de dos elementos) que empareja cada nombre de operación con un procedimiento de Scheme que implementa la operación (es decir, produce el mismo valor de salida dados los mismos valores de entrada). El último argumento especifica el controlador como una lista de etiquetas e instrucciones de máquina, como en la sección  [5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1) .

Para calcular MCD con esta máquina, configuramos los registros de entrada, iniciamos la máquina y examinamos el resultado cuando finaliza la simulación:

(establecer-contenido-registro! máquina-gcd 'a 206)   
*hecho*  
(establecer-contenido-registro! máquina-gcd 'b 40)   
*hecho*  
(iniciar máquina-gcd)   
*hecho*  
(obtener-contenido-registro máquina-gcd 'a)   
*2*

Este cálculo se ejecutará mucho más lentamente que un procedimiento gcd escrito en Scheme, porque simularemos instrucciones de máquina de bajo nivel, como asignar , mediante operaciones mucho más complejas.

**Ejercicio 5.7.**   Utilice el simulador para probar las máquinas que diseñó en el ejercicio  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.4) .

**[5.2.1 El modelo de máquina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.2.1)**

El modelo de máquina generado por make-machine se representa como un procedimiento con estado local que utiliza las técnicas de paso de mensajes desarrolladas en el capítulo 3. Para construir este modelo, make-machine comienza llamando al procedimiento make-new-machine para construir las partes del modelo de máquina que son comunes a todas las máquinas de registro. Este modelo de máquina básico construido por make-new-machine es esencialmente un contenedor para algunos registros y una pila, junto con un mecanismo de ejecución que procesa las instrucciones del controlador una por una.

Make-machine luego extiende este modelo básico (enviándole mensajes) para incluir los registros, las operaciones y el controlador de la máquina en particular que se está definiendo. Primero, asigna un registro en la nueva máquina para cada uno de los nombres de registro proporcionados e instala las operaciones designadas en la máquina. Luego, utiliza un*ensamblador* (descrito más adelante en la sección  [5.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2.2) ) para transformar la lista de controladores en instrucciones para la nueva máquina y las instala como secuencia de instrucciones de la máquina. Make-machine devuelve como valor el modelo de máquina modificado.

(define (make-machine nombres-registros ops texto-controlador)   
  (let ((machine (make-new-machine)))   
    (for-each (lambda (nombre-registro)   
                ((machine 'allocate-register) nombre-registro))   
              nombres-registro)   
    ((machine 'install-operations) ops)       
    ((machine 'install-instruction-secuencia)   
     (assemble texto-controlador machine))   
    machine))

**[Registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_724)**

Representaremos un registro como un procedimiento con estado local, como en el capítulo 3. El procedimiento make-register crea un registro que contiene un valor al que se puede acceder o modificar:

(define (make-register nombre)   
  (let ((contenido '\*sin asignar\*))   
    (define (enviar mensaje)   
      (cond ((eq? mensaje 'obtener) contenido)   
            ((eq? mensaje 'establecer)   
             (lambda (valor) (establecer! contenido valor)))   
            (else   
             (error "Solicitud desconocida -- REGISTER" mensaje))))   
    enviar))

Para acceder a los registros se utilizan los siguientes procedimientos:

(define (obtener-contenido registrar)   
  (registrar 'obtener))  
  
(define (set-contents! registro valor)   
  ((registro 'set) valor))

**[La pila](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_725)**

También podemos representar una pila como un procedimiento con estado local. El procedimiento make-stack crea una pila cuyo estado local consiste en una lista de los elementos que hay en ella. Una pila acepta solicitudes para insertar un elemento en ella, para sacar el elemento superior de la pila y devolverlo, y para inicializar la pila para que quede vacía.

(define (make-stack)   
  (let ((s '()))   
    (define (push x)   
      (set! s (cons x s)))   
    (define (pop)   
      (if (null? s)   
          (error "Pila vacía -- POP")   
          (let ((top (car s)))   
            (set! s (cdr s))   
            top)))   
    (define (inicializar)   
      (set! s '())   
      'done)   
    (define (enviar mensaje)   
      (cond ((eq? mensaje 'push) push)   
            ((eq? mensaje 'pop) (pop))   
            ((eq? mensaje 'inicializar) (inicializar))   
            (else (error "Solicitud desconocida -- STACK"   
                         mensaje))))   
    enviar))

Los siguientes procedimientos se utilizan para acceder a las pilas:

(define (pila emergente)   
  (pila 'explotar))  
  
(define (insertar valor de pila)   
  ((pila 'insertar valor))

**[La máquina básica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_726)**

El procedimiento de creación de una nueva máquina , que se muestra en la figura  [5.13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.13) , construye un objeto cuyo estado local consiste en una pila, una secuencia de instrucciones inicialmente vacía, una lista de operaciones que inicialmente contiene una operación parainicializar la pila y una*tabla de registros* que inicialmente contiene dosregistros, bandera nombrada y pc (para "contador de programa"). El procedimiento interno allocate-register agrega nuevas entradas a la tabla de registros, y el procedimiento interno lookup-register busca registros en la tabla.

El registro de bandera se utiliza para controlar la ramificación en la máquina simulada. Las instrucciones de prueba establecen el contenido de la bandera en el resultado de la prueba (verdadero o falso). Las instrucciones de ramificación deciden si se debe o no realizar una ramificación examinando el contenido de la bandera .

El registro pc determina la secuencia de instrucciones a medida que la máquina se ejecuta. Esta secuencia se implementa mediante el procedimiento interno de ejecución . En el modelo de simulación, cada instrucción de la máquina es una estructura de datos que incluye un procedimiento sin argumentos, llamado*procedimiento de ejecución de instrucciones* , de modo que al llamar a este procedimiento se simule la ejecución de la instrucción. A medida que se ejecuta la simulación, pc apunta al lugar en la secuencia de instrucciones que comienza con la siguiente instrucción que se ejecutará.Execute obtiene esa instrucción, la ejecuta llamando al procedimiento de ejecución de instrucciones y repite este ciclo hasta que no haya más instrucciones para ejecutar (es decir, hasta que pc apunte al final de la secuencia de instrucciones).

|  |
| --- |
| (define (make-new-machine)    (let ((pc (make-register 'pc))          (flag (make-register 'flag))          (stack (make-stack))          (the-instruction-secuence '()))      (let ((the-ops             (list (list 'initialize-stack                         (lambda () (stack 'initialize)))))            (register-table             (list (list 'pc pc) (list 'flag flag))))        (define (allocate-register nombre)          (if (assoc name tabla-registros)              (error "Multiplicar registro definido: " nombre)              (set! tabla-registros                    (cons (list name (make-register name))                          tabla-registros)))          'register-allocated)        (define (lookup-register nombre)          (let ((val (assoc name tabla-registros)))            (if val                (cadr val)                (error "Registro desconocido:" nombre))))        (define (ejecuta)          (let ((insts (get-contents pc)))            (if (null? insts)                'done                (begin                  ((instruction-execution-proc (car insts)))                  (ejecuta))))        (define (envía mensaje)          (cond ((eq? mensaje 'inicio)                 (set-contents! pc la-secuencia-de-instrucciones)                 (ejecuta))                ((eq? mensaje 'instala-secuencia-de-instrucciones)                 (lambda (seq) (set! la-secuencia-de-instrucciones seq)))                ((eq? mensaje 'allocate-register) allocate-register)                ((eq? mensaje 'get-register) lookup-register)                ((eq? mensaje 'install-operations)                 (lambda (ops) (set! the-ops (append the-ops ops))))                ((eq? mensaje 'stack) pila)                ((eq? mensaje 'operations) the-ops)                (else (error "Solicitud desconocida -- MACHINE" mensaje))))        envío))) |
|  |
| **Figura 5.13:**   El procedimiento de creación de una nueva máquina , que implementa el modelo básico de la máquina. |

Como parte de su funcionamiento, cada procedimiento de ejecución de instrucción modifica pc para indicar la siguiente instrucción que se ejecutará. Las instrucciones de bifurcación y goto cambian pc para apuntar al nuevo destino. Todas las demás instrucciones simplemente avanzan pc , haciendo que apunte a la siguiente instrucción en la secuencia. Observe que cada llamada aexecute llama aexecute nuevamente , pero esto no produce un bucle infinito porque ejecutar el procedimiento de ejecución de instrucción cambia el contenido de pc .

Make-new-machine devuelve un procedimiento de envío que implementa el acceso mediante el paso de mensajes al estado interno. Observe que el inicio de la máquina se logra configurando pc al comienzo de la secuencia de instrucciones y llamando aexecute .

Para mayor comodidad, proporcionamos una interfaz de procedimiento alternativa para la operación de inicio de una máquina , así como procedimientos para configurar y examinar el contenido de los registros, como se especifica al comienzo de la sección  [5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2) :

(define (iniciar máquina)   
  (máquina 'iniciar))  
(define (get-register-contents máquina nombre-registro)   
  (get-contents (get-register máquina nombre-registro)))  
(define (set-register-contents! machine register-name value)   
  (set-contents! (get-register machine register-name) value)   
  'hecho)

Estos procedimientos (y muchos procedimientos en las secciones  [5.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2.2) y [5.2.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2.3) ) utilizan lo siguiente para buscar el registro con un nombre determinado en una máquina determinada:

(define (get-register máquina nombre-reg)   
  ((máquina 'get-register) nombre-reg))

**[5.2.2 El ensamblador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.2.2)**

El ensamblador transforma la secuencia de expresiones del controlador de una máquina en una lista correspondiente de instrucciones de máquina, cada una con su procedimiento de ejecución. En general, el ensamblador es muy parecido a los evaluadores que estudiamos en el capítulo 4: hay un lenguaje de entrada (en este caso, el lenguaje de registro de la máquina) y debemos realizar una acción apropiada para cada tipo de expresión en el lenguaje.

La técnica de producir un procedimiento de ejecución para cada instrucción es exactamente lo que usamos en la sección  [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) para acelerar el evaluador al separar el análisis de la ejecución en tiempo de ejecución. Como vimos en el capítulo 4, se pueden realizar muchos análisis útiles de expresiones de Scheme sin conocer los valores reales de las variables. Aquí, de manera análoga, se pueden realizar muchos análisis útiles de expresiones de lenguaje de máquina de registros sin conocer el contenido real de los registros de la máquina. Por ejemplo, podemos reemplazar referencias a registros por punteros a los objetos de registro, y podemos reemplazar referencias a etiquetas por punteros al lugar en la secuencia de instrucciones que la etiqueta designa.

Antes de poder generar los procedimientos de ejecución de instrucciones, el ensamblador debe saber a qué se refieren todas las etiquetas, por lo que comienza escaneando el texto del controlador para separar las etiquetas de las instrucciones. A medida que escanea el texto, construye una lista de instrucciones y una tabla que asocia cada etiqueta con un puntero a esa lista. Luego, el ensamblador amplía la lista de instrucciones insertando el procedimiento de ejecución para cada instrucción.

El procedimiento de ensamblaje es la entrada principal al ensamblador. Toma el texto del controlador y el modelo de la máquina como argumentos y devuelve la secuencia de instrucciones que se almacenará en el modelo. Assemble llama a extract-labels para generar la lista de instrucciones inicial y la tabla de etiquetas a partir del texto del controlador proporcionado. El segundo argumento de extract-labels es un procedimiento que se llamará para procesar estos resultados: Este procedimiento usa update-insts! para generar los procedimientos de ejecución de instrucciones e insertarlos en la lista de instrucciones, y devuelve la lista modificada.

(define (ensamblar controlador-texto máquina)   
  (extraer-etiquetas controlador-texto   
    (lambda (insts etiquetas)   
      (actualizar-insts! insts etiquetas máquina)   
      insts)))

Extract-labels toma como argumentos una lista text (la secuencia de expresiones de instrucciones del controlador) y un procedimiento de recepción . Receive se llamará con dos valores: (1) una lista insts de estructuras de datos de instrucciones, cada una conteniendo una instrucción de text ; y (2) una tabla llamada labels , que asocia cada etiqueta de text con la posición en la lista insts que la etiqueta designa.

(define (extraer-etiquetas texto recibir)   
  (si (nulo? texto)   
      (recibir '() '())   
      (extraer-etiquetas (cdr texto)   
       (lambda (insts etiquetas)   
         (let ((next-inst (auto texto)))   
           (si (símbolo? next-inst)   
               (recibir insts   
                        (cons (hacer-etiqueta-entrada next-inst   
                                                insts)   
                              etiquetas))   
               (recibir (cons (hacer-instrucción next-inst)   
                              insts)   
                        etiquetas)))))))

Extract-labels funciona escaneando secuencialmente los elementos del texto y acumulando las insts y las labels . Si un elemento es un símbolo (y por lo tanto una etiqueta), se agrega una entrada apropiada a la tabla labels . De lo contrario, el elemento se acumula en la lista insts . [4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_727)

Update-insts! modifica la lista de instrucciones, que inicialmente contiene sólo el texto de las instrucciones, para incluir los procedimientos de ejecución correspondientes:

(define (update-insts! insts etiquetas máquina)   
  (let ((pc (get-register máquina 'pc))   
        (flag (get-register máquina 'flag))   
        (stack (máquina 'stack))   
        (ops (máquina 'operaciones)))   
    (for-each   
     (lambda (inst)   
       (set-instruction-execution-proc!    
        inst   
        (make-execution-procedure   
         (instruction-text inst) etiquetas máquina   
         pc flag pila ops)))   
     insts)))

La estructura de datos de la instrucción de la máquina simplemente empareja el texto de la instrucción con el procedimiento de ejecución correspondiente. El procedimiento de ejecución aún no está disponible cuando extract-labels construye la instrucción, y se inserta más tarde mediante update-insts !.

(define (make-instruction texto)   
  (cons texto '()))  
(define (texto de instrucción inst)   
  (auto inst))  
(define (proc-ejecucion-instrucción inst)   
  (cdr inst))  
(define (establecer-proc-ejecucion-instrucción! inst proc)   
  (establecer-cdr! inst proc))

Nuestro simulador no utiliza el texto de instrucciones, pero es útil tenerlo a mano para depurar (ver ejercicio  [5.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.16) ).

Los elementos de la tabla de etiquetas son pares:

(define (make-label-entry nombre-de-etiqueta insts)   
  (cons nombre-de-etiqueta insts))

Las entradas se buscarán en la tabla con

(define (lookup-label labels nombre-etiqueta)   
  (let ((val (assoc nombre-etiqueta etiquetas)))   
    (if val   
        (cdr val)   
        (error "Etiqueta no definida -- ASSEMBLE" nombre-etiqueta))))

**Ejercicio 5.8.**   El siguiente código de máquina de registro es ambiguo, porque la etiqueta aquí se define más de una vez:

inicio   
  (goto (etiqueta aquí))   
aquí   
  (asigne una (const 3))   
  (goto (etiqueta allí))   
aquí   
  (asigne una (const 4))   
  (goto (etiqueta allí))   
allí

Con el simulador tal como está escrito, ¿cuál será el contenido del registro a cuando el control llegue allí ? Modifique el procedimiento extract-labels para que el ensamblador indique un error si se utiliza el mismo nombre de etiqueta para indicar dos ubicaciones diferentes.

**[5.2.3 Generación de procedimientos de ejecución para instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.2.3)**

El ensamblador llama a make-execution-procedure para generar el procedimiento de ejecución de una instrucción. Al igual que el procedimiento de análisis del evaluador de la sección  [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) , este se despacha según el tipo de instrucción para generar el procedimiento de ejecución adecuado.

(define (make-ejecución-procedimiento inst etiquetas máquina   
                                  pc bandera pila ops)   
  (cond ((eq? (car inst) 'asignar)   
         (make-asignar inst máquina etiquetas ops pc))   
        ((eq? (car inst) 'probar)   
         (make-prueba inst máquina etiquetas ops bandera pc))   
        ((eq? (car inst) 'rama)   
         (make-rama inst máquina etiquetas bandera pc))   
        ((eq? (car inst) 'ir)   
         (make-goto inst máquina etiquetas pc))   
        ((eq? (car inst) 'guardar)   
         (make-guardar inst máquina pila pc))   
        ((eq? (car inst) 'restaurar)   
         (make-restaurar inst máquina pila pc))   
        ((eq? (car inst) 'ejecutar)   
         (make-ejecutar inst máquina etiquetas ops pc))   
        (else (error "Tipo de instrucción desconocido -- ASSEMBLE"   
                     inst))))

Para cada tipo de instrucción en el lenguaje de registro-máquina, existe un generador que construye un procedimiento de ejecución apropiado. Los detalles de estos procedimientos determinan tanto la sintaxis como el significado de las instrucciones individuales en el lenguaje de registro-máquina. Utilizamos la abstracción de datos para aislar la sintaxis detallada de las expresiones de registro-máquina del mecanismo de ejecución general, como hicimos para los evaluadores en la sección  [4.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.2) , utilizando procedimientos de sintaxis para extraer y clasificar las partes de una instrucción.

**[Asignar instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_729)**

El procedimiento make-assign maneja instrucciones de asignación :

(define (make-assign inst máquina etiquetas operaciones pc)   
  (let ((target   
         (get-register máquina (assign-reg-name inst)))   
        (value-exp (assign-value-exp inst)))   
    (let ((value-proc   
           (if (operation-exp? value-exp)   
               (make-operation-exp   
                value-exp máquina etiquetas operaciones)   
               (make-primitive-exp   
                (car value-exp) máquina etiquetas))))   
      (lambda ()                 *; procedimiento de ejecución para  assignment*  
        (set-contents! target (value-proc))   
        (advance-pc pc)))))

Make-assign extrae el nombre del registro de destino (el segundo elemento de la instrucción) y la expresión de valor (el resto de la lista que forma la instrucción) de la instrucción de asignación utilizando los selectores

(define (nombre-reg-asignar instrucción-asignar)   
  (cadr instrucción-asignar))  
(define (exp-asignar-valor instrucción-asignar)   
  (cddr instrucción-asignar))

El nombre del registro se busca con get-register para producir el objeto de registro de destino. La expresión de valor se pasa a make-operation-exp si el valor es el resultado de una operación y a make-primitive-exp en caso contrario. Estos procedimientos (que se muestran a continuación) analizan la expresión de valor y producen un procedimiento de ejecución para el valor. Este es un procedimiento sin argumentos, llamadovalue-proc , que se evaluará durante la simulación para producir el valor real que se asignará al registro. Observe que el trabajo de buscar el nombre del registro y analizar la expresión del valor se realiza solo una vez, en el momento del ensamblaje, no cada vez que se simula la instrucción. Este ahorro de trabajo es la razón por la que utilizamos la ejecuciónprocedimientos, y corresponde directamente al ahorro de trabajo que obtuvimos al separar el análisis del programa de su ejecución en el evaluador de la sección  [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) .

El resultado devuelto por make-assign es el procedimiento de ejecución de la instrucción de asignación . Cuando se llama a este procedimiento (por el procedimiento de ejecución del modelo de máquina ), establece el contenido del registro de destino en el resultado obtenido al ejecutar value-proc . Luego, avanza el pc a la siguiente instrucción ejecutando el procedimiento.

(define (advance-pc pc)   
  (establece-contenidos! pc (cdr (obtiene-contenidos pc))))

Advance-pc es la terminación normal para todas las instrucciones excepto branch y goto .

**[Instrucciones de prueba , rama y goto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_730)**

Make-test maneja las instrucciones de prueba de una manera similar. Extrae la expresión que especifica la condición que se va a probar y genera un procedimiento de ejecución para ella. En el momento de la simulación, se llama al procedimiento para la condición, el resultado se asigna al registro de indicadores y se avanza al pc :

(define (make-test inst máquina etiquetas operaciones flag pc)   
  (let ((condición (test-condition inst)))   
    (if (operation-exp? condición)   
        (let ((condition-proc   
               (make-operation-exp   
                condición máquina etiquetas operaciones)))   
          (lambda ()   
            (set-contents! flag (condition-proc))   
            (advance-pc pc)))   
        (error "Instrucción TEST incorrecta -- ASSEMBLE" inst))))  
(define (condición-de-prueba instrucción-de-prueba)   
  (cdr instrucción-de-prueba))

El procedimiento de ejecución de una instrucción de bifurcación verifica el contenido del registro de indicadores y establece el contenido del pc en el destino de la bifurcación (si se toma la bifurcación) o simplemente avanza el pc (si no se toma la bifurcación). Observe que el destino indicado en una instrucción de bifurcación debe ser una etiqueta, y el procedimiento make-branch lo hace cumplir. Observe también que la etiqueta se busca en el momento del ensamblaje, no cada vez que se simula la instrucción de bifurcación .

(define (make-branch inst machine labels flag pc)   
  (let ((dest (branch-dest inst)))   
    (if (label-exp? dest)   
        (let ((insts   
               (lookup-label labels (label-exp-label dest))))   
          (lambda ()   
            (if (get-contents flag)   
                (set-contents! pc insts)   
                (advance-pc pc))))   
        (error "Instrucción BRANCH incorrecta -- ASSEMBLE" inst))))  
(define (branch-dest instrucción-de-rama)   
  (cadr instrucción-de-rama))

Una instrucción goto es similar a una rama, excepto que el destino puede especificarse como una etiqueta o como un registro, y no hay ninguna condición que verificar: la pc siempre se establece en el nuevo destino.

(define (make-goto inst máquina etiquetas pc)   
  (let ((dest (goto-dest inst)))   
    (cond ((exp-etiqueta? dest)   
           (let ((insts   
                  (lookup-label etiquetas   
                                (exp-etiqueta-etiqueta dest))))   
             (lambda () (set-contents! pc insts))))   
          ((exp-registro? dest)   
           (let ((reg   
                  (get-register máquina   
                                (exp-registro-reg dest))))   
             (lambda ()   
               (set-contents! pc (get-contents reg)))))   
          (else (error "Instrucción GOTO incorrecta -- ASSEMBLE"   
                       inst)))))  
(define (goto-dest goto-instrucción)   
  (cadr goto-instrucción))

**[Otras instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_731)**

Las instrucciones de pila guardan y restauran simplemente usan la pila con el registro designado y avanzan la pc :

(define (make-save inst machine stack pc)   
  (let ((reg (get-register machine   
                           (stack-inst-reg-name inst))))   
    (lambda ()   
      (push stack (get-contents reg))   
      (advance-pc pc))))  
(define (make-restore inst machine stack pc)   
  (let ((reg (get-register machine   
                           (stack-inst-reg-name inst))))   
    (lambda ()   
      (set-contents! reg (pop stack))       
      (advance-pc pc))))  
(define (stack-inst-reg-name instrucción-pila)   
  (cadr instrucción-pila))

El tipo de instrucción final, manejado por make-perform , genera un procedimiento de ejecución para la acción que se va a realizar. En el momento de la simulación, se ejecuta el procedimiento de acción y se avanza a la computadora .

(define (make-perform inst máquina etiquetas operaciones pc)   
  (let ((action (perform-action inst)))   
    (if (operation-exp? action)   
        (let ((action-proc   
               (make-operation-exp   
                action máquina etiquetas operaciones)))   
          (lambda ()   
            (action-proc)   
            (advance-pc pc)))   
        (error "Instrucción PERFORM incorrecta -- ASSEMBLE" inst))))  
(define (ejecutar-acción inst) (cdr inst))

**[Procedimientos de ejecución para subexpresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_732)**

El valor de una expresión reg , label o const puede ser necesario para la asignación a un registro ( make-assign ) o para la entrada a una operación ( make-operation-exp , a continuación). El siguiente procedimiento genera procedimientos de ejecución para producir valores para estas expresiones durante la simulación:

(define (make-primitive-exp exp máquina etiquetas)   
  (cond ((constant-exp? exp)   
         (let ((c (constant-exp-value exp)))   
           (lambda () c)))   
        ((label-exp? exp)   
         (let ((insts   
                (lookup-label labels   
                              (label-exp-label exp))))   
           (lambda () insts)))   
        ((register-exp? exp)   
         (let ((r (get-register machine   
                                (register-exp-reg exp))))   
           (lambda () (get-contents r))))   
        (else   
         (error "Tipo de expresión desconocido -- ASSEMBLE" exp))))

La sintaxis de las expresiones reg , label y const está determinada por

(define (exp-registro? exp) (lista-etiquetada? exp 'reg))  
(define (registro-exp-reg exp) (cadr exp))  
(define (exp-constante? exp) (lista-etiquetada? exp 'const))  
(define (valor-exp-constante exp) (cadr exp))  
(define (etiqueta-exp? exp) (lista-etiquetada? exp 'etiqueta))  
(define (etiqueta-exp-etiqueta exp) (cadr exp))

Las instrucciones de asignación , ejecución y prueba pueden incluir la aplicación de una operación de máquina (especificada por una expresión op ) a algunos operandos (especificados por expresiones reg y const ). El siguiente procedimiento produce un procedimiento de ejecución para una ``expresión de operación'': una lista que contiene las expresiones de operación y operando de la instrucción:

(define (make-operacion-exp exp máquina etiquetas operaciones)   
  (let ((op (lookup-prim (operacion-exp-op exp) operaciones))   
        (aprocs   
         (map (lambda (e)   
                (make-primitive-exp e máquina etiquetas))   
              (operacion-exp-operandos exp))))   
    (lambda ()   
      (apply op (map (lambda (p) (p)) aprocs)))))

La sintaxis de las expresiones de operación está determinada por

(define (operación-exp? exp)   
  (y (par? exp) (lista-etiquetada? (auto exp) 'op)))  
(define (operación-exp-op operación-exp)   
  (cadr (coche operación-exp)))  
(define (operación-exp-operandos operación-exp)   
  (cdr operación-exp))

Observe que el tratamiento de las expresiones de operación es muy similar al tratamiento de las aplicaciones de procedimiento por parte del procedimiento  de análisis de aplicaciones en el evaluador de la sección [4.1.7,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) ya que generamos un procedimiento de ejecución para cada operando. En el momento de la simulación, llamamos a los procedimientos de operando y aplicamos el procedimiento Scheme que simula la operación a los valores resultantes. El procedimiento de simulación se encuentra buscando el nombre de la operación en la tabla de operaciones de la máquina:

(define (lookup-prim operaciones de símbolo)   
  (let ((val (assoc operaciones de símbolo)))   
    (if val   
        (cadr val)   
        (error "Operación desconocida -- símbolo ASSEMBLE"))))

**Ejercicio 5.9.**   El tratamiento de las operaciones de máquina descrito anteriormente permite que estas operen sobre etiquetas, así como sobre constantes y el contenido de los registros. Modifique los procedimientos de procesamiento de expresiones para aplicar la condición de que las operaciones solo se puedan utilizar con registros y constantes.

**Ejercicio 5.10.**   Diseñe una nueva sintaxis para las instrucciones de registro-máquina y modifique el simulador para que utilice su nueva sintaxis. ¿Puede implementar su nueva sintaxis sin cambiar ninguna parte del simulador excepto los procedimientos de sintaxis de esta sección?

**Ejercicio 5.11.**  Cuando presentamos guardar y restaurar en la sección  [5.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1.4) , no especificamos qué sucedería si intentara restaurar un registro que no fuera el último guardado, como en la secuencia

(guardar y)   
(guardar x)   
(restaurar y)

Existen varias posibilidades razonables para el significado de restaurar :

a.   (restaurar y) coloca en y el último valor guardado en la pila, sin importar de qué registro provenga ese valor. Así se comporta nuestro simulador. Muestre cómo aprovechar este comportamiento para eliminar una instrucción de la máquina de Fibonacci de la sección  [5.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1.4) (figura  [5.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.12) ).

b.   (restaurar y) coloca en y el último valor guardado en la pila, pero solo si ese valor se guardó desde y ; de lo contrario, indica un error. Modifique el simulador para que se comporte de esta manera. Deberá cambiar la función de guardar para colocar el nombre del registro en la pila junto con el valor.

c.   (restaurar y) coloca en y el último valor guardado de y sin importar qué otros registros se guardaron después de y y no se restauraron. Modifique el simulador para que se comporte de esta manera. Deberá asociar una pila separada con cada registro. Debe hacer que la operación initialize-stack inicialice todas las pilas de registros.

**Ejercicio 5.12.**   El simulador puede utilizarse para ayudar a determinar las rutas de datos necesarias para implementar una máquina con un controlador determinado. Extienda el ensamblador para almacenar la siguiente información en el modelo de la máquina:

* una lista de todas las instrucciones, con los duplicados eliminados, ordenadas por tipo de instrucción ( asignar , goto , etc.);
* una lista (sin duplicados) de los registros utilizados para contener puntos de entrada (estos son los registros a los que hacen referencia las instrucciones goto );
* una lista (sin duplicados) de los registros que se guardan o se restauran ;
* para cada registro, una lista (sin duplicados) de las fuentes desde las que se asigna (por ejemplo, las fuentes para el registro val en la máquina factorial de la figura  [5.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.11) son (const 1) y ((op \*) (reg n) (reg val)) ).

Amplíe la interfaz de paso de mensajes a la máquina para proporcionar acceso a esta nueva información. Para probar su analizador, defina la máquina de Fibonacci de la figura  [5.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.12) y examine las listas que construyó.

**Ejercicio 5.13.**   Modifique el simulador para que utilice la secuencia del controlador para determinar qué registros tiene la máquina en lugar de requerir una lista de registros como argumento para make-machine . En lugar de preasignar los registros en make-machine , puede asignarlos uno a la vez cuando se los ve por primera vez durante el ensamblaje de las instrucciones.

**[5.2.4 Monitoreo del rendimiento de la máquina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.2.4)**

La simulación es útil no sólo para verificar la corrección de un diseño de máquina propuesto, sino también para medir el rendimiento de la máquina. Por ejemplo, podemos instalar en nuestro programa de simulación un "medidor" que mida la cantidad de operaciones de pila utilizadas en un cálculo. Para ello, modificamos nuestra pila simulada para realizar un seguimiento de la cantidad de veces que se guardan los registros en la pila y la profundidad máxima alcanzada por la pila, y agregamos un mensaje a la interfaz de la pila que imprime las estadísticas, como se muestra a continuación. También agregamos una operación al modelo básico de la máquina para imprimir las estadísticas de la pila, inicializando las operaciones en make-new-machine para

(lista (lista 'inicializar-pila   
            (lambda () (pila 'inicializar)))   
      (lista 'imprimir-pila-estadísticas   
            (lambda () (pila 'imprimir-estadísticas))))

Aquí está la nueva versión de make-stack :

(define (make-stack)   
  (let ((s '())   
        (numero-empujes 0)   
        (max-depth 0)   
        (actual-depth 0))   
    (define (push x)   
      (set! s (cons x s))   
      (set! numero-empujes (+ 1 numero-empujes))   
      (set! profundidad-actual (+ 1 profundidad-actual))   
      (set! max-depth (max current-depth max-depth)))   
    (define (pop)   
      (if (null? s)   
          (error "Pila vacía -- POP")   
          (let ((top (car s)))   
            (set! s (cdr s))   
            (set! profundidad-actual (- profundidad-actual 1))   
            top)))       
    (define (inicializar)   
      (set! s '())   
      (set! numero-empujes 0)   
      (set! max-depth 0)   
      (set! profundidad-actual 0)   
      'hecho)   
    (define (print-statistics)   
      (newline)   
      (display (lista 'total-pushes '= number-pushes   
                     'maximum-depth '= max-depth)))   
    (define (enviar mensaje)   
      (cond ((eq? mensaje 'push) push)   
            ((eq? mensaje 'pop) (pop))   
            ((eq? mensaje 'inicializar) (inicializar))   
            ((eq? mensaje 'imprimir-estadísticas)   
             (imprimir-estadísticas))   
            (else   
             (error "Solicitud desconocida -- STACK" mensaje))))   
    enviar))

Los ejercicios  [5.15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.15) a  [5.19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.19) describen otras funciones útiles de monitoreo y depuración que se pueden agregar al simulador de máquina de registro.

**Ejercicio 5.14.**  Mida el número de empujes y la profundidad máxima de pila requerida para calcular *n* ! para varios valores pequeños de *n* utilizando la máquina factorial que se muestra en la figura  [5.11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.11) . A partir de sus datos, determine fórmulas en términos de *n* para el número total de operaciones de empuje y la profundidad máxima de pila utilizada para calcular *n* ! para cualquier *n* > 1. Tenga en cuenta que cada una de estas es una función lineal de *n* y, por lo tanto, está determinada por dos constantes. Para obtener las estadísticas impresas, tendrá que aumentar la máquina factorial con instrucciones para inicializar la pila e imprimir las estadísticas. Es posible que también desee modificar la máquina para que lea repetidamente un valor para *n* , calcule el factorial e imprima el resultado (como hicimos para la máquina MCD en la figura  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.4) ), de modo que no tenga que invocar repetidamente get-register-contents , set-register-contents! y start .

**Ejercicio 5.15.**   Sumar*Conteo de instrucciones* en la simulación de la máquina de registros. Es decir, hacer que el modelo de la máquina lleve un registro de la cantidad de instrucciones ejecutadas. Ampliar la interfaz del modelo de la máquina para aceptar un nuevo mensaje que imprima el valor del recuento de instrucciones y restablezca el recuento a cero.

**Ejercicio 5.16.**   Ampliar el simulador para que proporcione*Seguimiento de instrucciones* . Es decir, antes de ejecutar cada instrucción, el simulador debe imprimir el texto de la instrucción. Hacer que el modelo de la máquina acepte mensajes de seguimiento activado y desactivado para activar y desactivar el seguimiento.

**Ejercicio 5.17.**   Amplíe el seguimiento de instrucciones del ejercicio  [5.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.16) de modo que antes de imprimir una instrucción, el simulador imprima cualquier etiqueta que preceda inmediatamente a esa instrucción en la secuencia del controlador. Tenga cuidado de hacer esto de una manera que no interfiera con el conteo de instrucciones (ejercicio  [5.15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.15) ). Deberá hacer que el simulador retenga la información de etiqueta necesaria.

**Ejercicio 5.18.**  Modifique el procedimiento  de creación de registros de la sección [5.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2.1) para que se puedan rastrear los registros. Los registros deben aceptar mensajes que activen y desactiven el rastreo. Cuando se rastrea un registro, la asignación de un valor al registro debe imprimir el nombre del registro, el contenido anterior del registro y el nuevo contenido que se está asignando. Amplíe la interfaz al modelo de máquina para permitirle activar y desactivar el rastreo para los registros de máquina designados.

**Ejercicio 5.19.**   Alyssa P. Hacker quiere un*función de punto de interrupción* en el simulador para ayudarla a depurar los diseños de su máquina. Se le ha contratado para instalar esta función para ella. Ella quiere poder especificar un lugar en la secuencia del controlador donde el simulador se detendrá y le permitirá examinar el estado de la máquina. Debe implementar un procedimiento

(establecer punto de interrupción < *máquina* > < *etiqueta* > < *n* >)

que establece un punto de interrupción justo antes de la instrucción *n* después de la etiqueta dada. Por ejemplo,

(establecer punto de interrupción máquina-gcd 'prueba-b 4)

instala un punto de interrupción en gcd-machine justo antes de la asignación para registrar un . Cuando el simulador alcanza el punto de interrupción, debe imprimir la etiqueta y el desplazamiento del punto de interrupción y dejar de ejecutar instrucciones. Alyssa puede usar get-register-contents y set-register-contents! para manipular el estado de la máquina simulada. Luego debería poder continuar la ejecución diciendo

(proceder-maquina < *maquina* >)

También debería poder eliminar un punto de interrupción específico mediante

(cancelar-punto-de-interrupción *< máquina* > *<etiqueta>* <n> )

o eliminar todos los puntos de interrupción mediante

(cancelar-todos-los-puntos-de-interrupción < *máquina* >)

[4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_727) El uso del procedimiento de recepción aquí es una forma de lograr que extract-labels devuelva efectivamente dos valores ( etiquetas e insts ) sin crear explícitamente una estructura de datos compuesta para almacenarlos. Una implementación alternativa, que devuelve un par explícito de valores, es

(define (extraer-etiquetas texto)   
  (si (nulo? texto)   
      (cons '() '())   
      (let ((resultado (extraer-etiquetas (cdr texto))))   
        (let ((insts (auto resultado)) (etiquetas (cdr resultado)))   
          (let ((next-inst (auto texto)))   
            (si (símbolo? next-inst)   
                (cons insts   
                      (cons (hacer-etiqueta-entrada next-inst insts) etiquetas))   
                (cons (cons (hacer-instrucción next-inst) insts)   
                      etiquetas)))))))

que se llamaría por asamblea de la siguiente manera:

(define (ensamblar controlador-texto máquina)   
  (let ((resultado (extraer-etiquetas controlador-texto)))   
    (let ((insts (auto resultado)) (etiquetas (cdr resultado)))   
      (actualizar-insts! insts etiquetas máquina)   
      insts)))

Puede considerar nuestro uso de recibir como una demostración de una forma elegante de devolver múltiples valores, o simplemente una excusa para mostrar un truco de programación. Un argumento como recibir , que es el siguiente procedimiento que se invocará, se denomina "continuación". Recuerde que también usamos continuaciones para implementar la estructura de control de retroceso en el evaluador amb en la sección  [4.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.3.3) .

[**5.3 Asignación de almacenamiento y recolección de basura**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_5.3)

En la sección  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4) , mostraremos cómo implementar un evaluador de Scheme como una máquina de registros. Para simplificar la discusión, asumiremos que nuestras máquinas de registros pueden estar equipadas con una *memoria estructurada en forma de lista* , en la que las operaciones básicas para manipular datos estructurados en forma de lista son primitivas. Postular la existencia de dicha memoria es una abstracción útil cuando uno se centra en los mecanismos de control en un intérprete de Scheme, pero esto no refleja una visión realista de las operaciones de datos primitivos reales de las computadoras contemporáneas. Para obtener una imagen más completa de cómo funciona un sistema Lisp, debemos investigar cómo se puede representar la estructura de lista de una manera que sea compatible con las memorias de computadora convencionales.

Hay dos consideraciones a tener en cuenta a la hora de implementar una estructura de lista. La primera es puramente una cuestión de representación: cómo representar la estructura de "caja y puntero" de los pares Lisp, utilizando sólo las capacidades de almacenamiento y direccionamiento de las memorias de ordenador típicas. La segunda cuestión se refiere a la gestión de la memoria a medida que avanza un cálculo. El funcionamiento de un sistema Lisp depende fundamentalmente de la capacidad de crear continuamente nuevos objetos de datos. Entre ellos se incluyen los objetos creados explícitamente por los procedimientos Lisp que se interpretan, así como las estructuras creadas por el propio intérprete, como los entornos y las listas de argumentos. Aunque la creación constante de nuevos objetos de datos no plantearía ningún problema en un ordenador con una cantidad infinita de memoria rápidamente direccionable, las memorias de ordenador están disponibles sólo en tamaños finitos (es una lástima). Por tanto, los sistemas Lisp proporcionan una*Función de asignación automática de almacenamiento* para respaldar la ilusión de una memoria infinita. Cuando un objeto de datos ya no es necesario, la memoria asignada a él se recicla automáticamente y se utiliza para construir nuevos objetos de datos. Existen varias técnicas para proporcionar dicha asignación automática de almacenamiento. El método que analizaremos en esta sección se denomina *recolección de basura* .

**[5.3.1 La memoria como vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.3.1)**

Una memoria de computadora convencional puede considerarse como una serie de compartimentos, cada uno de los cuales puede contener un fragmento de información. Cada compartimento tiene un nombre único, llamado su*dirección* o*Ubicación* . Los sistemas de memoria típicos proporcionan dos operaciones primitivas: una que recupera los datos almacenados en una ubicación específica y otra que asigna nuevos datos a una ubicación específica. Las direcciones de memoria se pueden incrementar para admitir el acceso secuencial a algún conjunto de cubículos. De manera más general, muchas operaciones de datos importantes requieren que las direcciones de memoria se traten como datos, que se pueden almacenar en ubicaciones de memoria y manipular en registros de máquina. La representación de la estructura de lista es una aplicación de tales*dirección aritmética* .

Para modelar la memoria de la computadora, utilizamos un nuevo tipo de estructura de datos llamada*vector* . De manera abstracta, un vector es un objeto de datos compuesto cuyos elementos individuales pueden ser accedidos por medio de un índice entero en un período de tiempo que es independiente del índice. [5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_744) Para describir las operaciones de memoria, utilizamos dos procedimientos primitivos de Scheme para manipular vectores:

* (vector-ref < *vector* > < *n* >) devuelve el *n-* ésimo elemento del vector.
* (vector-set! < *vector* > < *n* > < *valor* >) establece el *n-* ésimo elemento del vector en el valor designado.

Por ejemplo, si v es un vector, entonces (vector-ref v 5) obtiene la quinta entrada en el vector v y (vector-set! v 5 7) cambia el valor de la quinta entrada del vector v a 7. [6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_745) Para la memoria de la computadora, este acceso se puede implementar mediante el uso de aritmética de direcciones para combinar una *dirección base* que especifica la ubicación inicial de un vector en la memoria con un *índice* que especifica el desplazamiento de un elemento particular del vector.

**[Representación de datos Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_746)**

Podemos utilizar vectores para implementar las estructuras de pares básicas necesarias para una memoria estructurada en forma de lista. Imaginemos que la memoria de la computadora está dividida en dos vectores:los-coches ythe-cdrs . Representaremos la estructura de la lista de la siguiente manera: un puntero a un par es un índice en los dos vectores. El car del par es la entrada en the-cars con el índice designado, y el cdr del par es la entrada en the-cdrs con el índice designado. También necesitamos una representación para objetos distintos de pares (como números y símbolos) y una forma de distinguir un tipo de datos de otro. Hay muchos métodos para lograr esto, pero todos se reducen a usar*punteros tipificados* , es decir, extender la noción de ``puntero'' para incluir información sobre el tipo de datos. [7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_747) El tipo de datos permite al sistema distinguir un puntero a un par (que consiste en el tipo de datos ``par'' y un índice en los vectores de memoria) de punteros a otros tipos de datos (que consisten en algún otro tipo de datos y lo que sea que se esté utilizando para representar datos de ese tipo). Dos objetos de datos sonse considera que son iguales ( eq? ) si sus punteros son idénticos. [8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_748) La figura  [5.14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.14) ilustra el uso de este método para representar la lista ((1 2) 3 4) , cuyo diagrama de caja y puntero también se muestra. Usamos prefijos de letras para denotar la información del tipo de datos. Así, un puntero al par con índice 5 se denota p5 , la lista vacía se denota por el puntero e0 y un puntero al número 4 se denota n4 . En el diagrama de caja y puntero, hemos indicado en la parte inferior izquierda de cada par el índice vectorial que especifica dónde se almacenan el car y el cdr del par. Las ubicaciones en blanco en the-cars y the-cdrs pueden contener partes de otras estructuras de lista (no son de interés aquí).

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 5.14:**   Representaciones de caja y puntero y vector de memoria de la lista ((1 2) 3 4) . |

Un puntero a un número, como n4 , podría consistir en un tipo que indica datos numéricos junto con la representación real del número 4. [9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_749) Para tratar con números que son demasiado grandes para ser representados en la cantidad fija de espacio asignado para un solo puntero, podríamos usar un*tipo de datos bignum* , para el cual el puntero designa una lista en la que se almacenan las partes del número. [10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_750)

Un símbolo puede representarse como un puntero tipificado que designa una secuencia de caracteres que forman la representación impresa del símbolo. El lector Lisp construye esta secuencia cuando la cadena de caracteres se encuentra inicialmente en la entrada. Dado que queremos que dos instancias de un símbolo se reconozcan como el ``mismo'' símbolo mediante eq? ySi queremos que eq? sea una prueba sencilla de igualdad de punteros, debemos asegurarnos de que si el lector ve la misma cadena de caracteres dos veces, utilizará el mismo puntero (a la misma secuencia de caracteres) para representar ambas apariciones. Para lograr esto, el lector mantiene una tabla, tradicionalmente llamada*obarray* , de todos los símbolos que ha encontrado. Cuando el lector encuentra una cadena de caracteres y está a punto de construir un símbolo, comprueba el obarray para ver si ha visto antes la misma cadena de caracteres. Si no es así, utiliza los caracteres para construir un nuevo símbolo (un puntero tipeado a una nueva secuencia de caracteres) e introduce este puntero en el obarray. Si el lector ha visto la cadena antes, devuelve el puntero de símbolo almacenado en el obarray. Este proceso de sustitución de cadenas de caracteres por punteros únicos se denominasímbolos *de internación .*

**[Implementación de las operaciones de lista primitivas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_751)**

Dado el esquema de representación anterior, podemos reemplazar cada operación de lista "primitiva" de una máquina de registros con una o más operaciones vectoriales primitivas. Usaremos dos registros, the-cars y the-cdrs , para identificar los vectores de memoria, y asumiremos que vector-ref y vector-set! están disponibles como operaciones primitivas. También asumimos que las operaciones numéricas sobre punteros (como incrementar un puntero, usar un puntero de pares para indexar un vector o sumar dos números) usan solo la porción de índice del puntero tipificado.

Por ejemplo, podemos hacer que una máquina de registro admita las instrucciones.

(asignar < *reg 1* > (op car) (reg < *reg 2* >))   
  
(asignar < *reg 1* > (op cdr) (reg < *reg 2* >))

Si implementamos estos, respectivamente, como

(asignar < *reg 1* > (op vector-ref) (reg los-coches) (reg < *reg 2* >))   
  
(asignar < *reg 1* > (op vector-ref) (reg los-cdrs) (reg < *reg 2* >))

Las instrucciones

(realizar (op set-car!) (reg < *reg 1* >) (reg < *reg 2* >))   
  
(realizar (op set-cdr!) (reg < *reg 1* >) (reg < *reg 2* >))

se implementan como

(realizar   
 (op vector-set!) (reg los-coches) (reg < *reg 1* >) (reg < *reg 2* >))   
  
(realizar   
 (op vector-set!) (reg los-cdrs) (reg < *reg 1* >) (reg < *reg 2* >))

La función cons se ejecuta asignando un índice no utilizado y almacenando los argumentos de cons en the-cars y the-cdrs en esa posición del vector indexado. Suponemos que hay un registro especial,libre , que siempre contiene un puntero de par que contiene el siguiente índice disponible, y que podemos incrementar la parte de índice de ese puntero para encontrar la siguiente ubicación libre. [11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_752) Por ejemplo, la instrucción

(asignar < *reg 1* > (op cons) (reg < *reg 2* >) (reg < *reg 3* >))

se implementa como la siguiente secuencia de operaciones vectoriales: [12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_753)

(ejecutar   
 (op vector-set!) (reg los-coches) (reg libre) (reg < *reg 2* >))   
(ejecutar   
 (op vector-set!) (reg los-cdrs) (reg libre) (reg < *reg 3* >))   
(asignar < *reg 1* > (reg libre))   
(asignar libre (op +) (reg libre) (const 1))

La operación eq?

(op eq?) (reg < *reg 1* >) (reg < *reg 2* >)

simplemente prueba la igualdad de todos los campos en los registros, yLos predicados como par?, nulo ?, símbolo ? y número? solo necesitan verificar el campo de tipo.

**[Implementando pilas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_754)**

Aunque nuestras máquinas de registro utilizan pilas, no necesitamos hacer nada especial aquí, ya que las pilas se pueden modelar en términos de listas. La pila puede ser una lista de los valores guardados, apuntada por un registro especial the-stack . Por lo tanto, (save < *reg* >) se puede implementar como

(asignar la pila (op cons) (reg < *reg* >) (reg la pila))

De manera similar, (restore < *reg* >) se puede implementar como

(asignar < *reg* > (op car) (reg la-pila))   
(asignar la-pila (op cdr) (reg la-pila))

y (perform (op initialize-stack)) se puede implementar como

(asignar la pila (const ()))

Estas operaciones se pueden ampliar aún más en términos de las operaciones vectoriales indicadas anteriormente. Sin embargo, en las arquitecturas informáticas convencionales, suele ser ventajoso asignar la pila como un vector independiente. Luego, se puede empujar y sacar la pila incrementando o decrementando un índice en ese vector.

**Ejercicio 5.20.**   Dibuje la representación de caja y puntero y la representación de vector de memoria (como en la figura  [5.14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.14) ) de la estructura de lista producida por

(define x (cons 1 2))   
(define y (lista x x))

con el puntero libre inicialmente p1 . ¿Cuál es el valor final de libre ? ¿Qué punteros representan los valores de x e y ?

**Ejercicio 5.21.**  Implemente máquinas de registro para los siguientes procedimientos. Suponga que las operaciones de memoria de estructura de lista están disponibles como primitivas de máquina.

a. Cuenta-hojas recursiva :

(define (count-leaves tree)   
  (cond ((null? tree) 0)   
        ((not (pair? tree)) 1)   
        (else (+ (count-leaves (car tree))   
                 (count-leaves (cdr tree))))))

b. Cuenta-hojas recursiva con contador explícito:

(define (árbol de contar-hojas)   
  (define (árbol de contar-iteradores n)   
    (cond ((árbol? nulo) n)   
          ((no (árbol? par)) (+ n 1))   
          (sino (árbol de contar-iteradores (árbol cdr)   
                            (árbol de contar-iteradores (árbol de automóviles) n)))))   
  (árbol de contar-iteradores 0))

**Ejercicio 5.22.**  El ejercicio  [3.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.12) de la sección  [3.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.1) presentó un procedimiento append que agrega dos listas para formar una nueva lista y un procedimiento append! que une dos listas. Diseñe una máquina de registros para implementar cada uno de estos procedimientos. Suponga que las operaciones de memoria de la estructura de lista están disponibles como operaciones primitivas.

**[5.3.2 Manteniendo la ilusión de la memoria infinita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.3.2)**

El método de representación descrito en la sección  [5.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.3.1) resuelve el problema de implementar la estructura de lista, siempre que tengamos una cantidad infinita de memoria. Con una computadora real, eventualmente nos quedaremos sin espacio libre en el que construir nuevos pares. [13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_758) Sin embargo, la mayoría de los pares generados en un cálculo típico se utilizan solo para almacenar resultados intermedios. Después de acceder a estos resultados, los pares ya no son necesarios: son *basura* . Por ejemplo, el cálculo

(acumular + 0 (filtrar impar? (enumerar-intervalo 0 n)))

construye dos listas: la enumeración y el resultado de filtrar la enumeración. Cuando se completa la acumulación, estas listas ya no son necesarias y se puede recuperar la memoria asignada. Si podemos organizarnos para recolectar toda la basura periódicamente y si esto resulta en reciclar la memoria aproximadamente al mismo ritmo al que construimos nuevos pares, habremos conservado la ilusión de que hay una cantidad infinita de memoria.

Para reciclar pares, debemos tener una manera de determinar qué pares asignados no son necesarios (en el sentido de que su contenido ya no puede influir en el futuro del cálculo). El método que examinaremos para lograr esto se conoce como *recolección de basura* . La recolección de basura se basa en la observación de que, en cualquier momento en una interpretación de Lisp, los únicos objetos que pueden afectar el futuro del cálculo son aquellos a los que se puede llegar mediante alguna sucesión de operaciones car y cdr a partir de los punteros que se encuentran actualmente en los registros de la máquina. [14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_759) Cualquier celda de memoria que no sea tan accesible puede reciclarse.

Existen muchas formas de realizar la recolección de basura. El método que examinaremos aquí se denomina*stop-and-copy* . La idea básica es dividir la memoria en dos mitades: "memoria de trabajo" y "memoria libre". Cuando cons construye pares, los asigna en la memoria de trabajo. Cuando la memoria de trabajo está llena, realizamos la recolección de basura ubicando todos los pares útiles en la memoria de trabajo y copiándolos en ubicaciones consecutivas en la memoria libre. (Los pares útiles se ubican rastreando todos los punteros car y cdr , comenzando con los registros de la máquina). Como no copiamos la basura, presumiblemente habrá memoria libre adicional que podemos usar para asignar nuevos pares. Además, no se necesita nada en la memoria de trabajo, ya que todos los pares útiles en ella han sido copiados. Por lo tanto, si intercambiamos los roles de memoria de trabajo y memoria libre, podemos continuar el procesamiento; los nuevos pares se asignarán en la nueva memoria de trabajo (que era la antigua memoria libre). Cuando está llena, podemos copiar los pares útiles en la nueva memoria libre (que era la antigua memoria de trabajo). [15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_760)

**[Implementación de un recolector de basura de tipo stop-and-copy](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_761)**

Ahora utilizaremos nuestro lenguaje de máquina de registros para describir el algoritmo de detención y copia con más detalle. Supondremos que hay un registro llamadoraíz que contiene un puntero a una estructura que eventualmente apunta a todos los datos accesibles. Esto se puede organizar almacenando el contenido de todos los registros de la máquina en una lista preasignada a la que apunta la raíz justo antes de comenzar la recolección de basura. [16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_762) También asumimos que, además de la memoria de trabajo actual, hay memoria libre disponible en la que podemos copiar los datos útiles. La memoria de trabajo actual consta de vectores cuyas direcciones base están enregistros llamados the-cars y the-cdrs , y la memoria libre está en registros llamadoscoches nuevos y cdrs nuevos .

La recolección de basura se activa cuando se agotan las celdas libres en la memoria de trabajo actual, es decir, cuando una operación cons intenta incrementar el puntero libre más allá del final del vector de memoria. Cuando se completa el proceso de recolección de basura, el puntero raíz apuntará a la nueva memoria, todos los objetos accesibles desde la raíz se habrán movido a la nueva memoria y el puntero libre indicará el siguiente lugar en la nueva memoria donde se puede asignar un nuevo par. Además, los roles de la memoria de trabajo y la nueva memoria se habrán intercambiado: se construirán nuevos pares en la nueva memoria, comenzando en el lugar indicado por free , y la memoria de trabajo (anterior) estará disponible como la nueva memoria para la próxima recolección de basura. La Figura  [5.15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.15) muestra la disposición de la memoria justo antes y justo después de la recolección de basura.

|  |
| --- |
| Forma  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 5.15:**   Reconfiguración de la memoria por el proceso de recolección de basura. |

El estado del proceso de recolección de basura se controla manteniendo dos punteros: free y scan . Estos se inicializan para apuntar al comienzo de la nueva memoria. El algoritmo comienza reubicando el par apuntado por root al comienzo de la nueva memoria. El par se copia, el puntero root se ajusta para apuntar a la nueva ubicación y el puntero free se incrementa. Además, la ubicación anterior del par se marca para mostrar que su contenido se ha movido. Este marcado se realiza de la siguiente manera: En la posición del automóvil , colocamos una etiqueta especial que indica que este es un objeto ya movido. (Un objeto de este tipo se denomina tradicionalmente*corazón roto* .) [17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_763) En la posición cdr colocamos un*dirección de reenvío* que apunta a la ubicación a la que se ha movido el objeto.

Después de reubicar la raíz, el recolector de basura entra en su ciclo básico. En cada paso del algoritmo, el puntero de escaneo (que inicialmente apunta a la raíz reubicada) apunta a un par que se ha movido a la nueva memoria pero cuyos punteros car y cdr aún hacen referencia a objetos en la memoria anterior. Cada uno de estos objetos se reubica y el puntero de escaneo se incrementa. Para reubicar un objeto (por ejemplo, el objeto indicado por el puntero car del par que estamos escaneando) verificamos si el objeto ya se ha movido (como lo indica la presencia de una etiqueta de corazón roto en la posición car del objeto). Si el objeto aún no se ha movido, lo copiamos al lugar indicado por free , actualizamos free , configuramos un corazón roto en la ubicación anterior del objeto y actualizamos el puntero al objeto (en este ejemplo, el puntero car del par que estamos escaneando) para que apunte a la nueva ubicación. Si el objeto ya se ha movido, su dirección de reenvío (que se encuentra en la posición cdr del corazón roto) sustituye al puntero del par que se está escaneando. Finalmente, todos los objetos accesibles se habrán movido y escaneado, momento en el que el puntero de escaneo superará al puntero libre y el proceso finalizará.

Podemos especificar el algoritmo de parada y copia como una secuencia de instrucciones para una máquina de registros. El paso básico de reubicar un objeto se logra mediante una subrutina llamada relocate-old-result-in-new . Esta subrutina obtiene su argumento, un puntero al objeto que se va a reubicar, de un registro llamadoantiguo . Reubica el objeto designado (incrementando la libertad en el proceso), coloca un puntero al objeto reubicado en un registro llamadonew , y retorna al punto de entrada almacenado en el registro relocate-continue . Para comenzar la recolección de basura, invocamos esta subrutina para reubicar el puntero raíz , después de inicializar free y scan . Cuando se ha logrado la reubicación de la raíz , instalamos el nuevo puntero como la nueva raíz e ingresamos al bucle principal del recolector de basura.

comenzar-recolección-de-basura   
  (asignar libre (const 0))   
  (asignar escaneo (const 0))   
  (asignar antiguo (reg raíz))   
  (asignar reubicar-continuar (etiqueta reasignar-raíz))   
  (goto (etiqueta reubicar-antiguo-resultado-en-nuevo))   
reasignar-raíz   
  (asignar raíz (reg nuevo))   
  (goto (etiqueta gc-loop))

En el bucle principal del recolector de basura debemos determinar si hay más objetos para escanear. Para ello, probamos si el puntero de escaneo coincide con el puntero libre . Si los punteros son iguales, entonces todos los objetos accesibles han sido reubicados y pasamos a gc-flip , que limpia todo para que podamos continuar con el cálculo interrumpido. Si todavía hay pares para escanear, llamamos a la subrutina relocate para reubicar el automóvil del siguiente par (colocando el puntero car en old ). El registro relocate-continue está configurado para que la subrutina regrese para actualizar el puntero car .

gc-loop   
  (prueba (op =) (reg scan) (reg free))   
  (rama (etiqueta gc-flip))   
  (asignar antiguo (op vector-ref) (reg nuevos-autos) (reg scan))   
  (asignar reubicar-continuar (etiqueta actualizar-auto))   
  (goto (etiqueta reubicar-antiguo-resultado-en-nuevo))

En update-car , modificamos el puntero del coche del par que se está escaneando y luego procedemos a reubicar el cdr del par. Regresamos a update-cdr cuando se ha completado esa reubicación. Después de reubicar y actualizar el cdr , terminamos de escanear ese par, por lo que continuamos con el bucle principal.

update-car   
  (ejecutar   
   (op vector-set!) (reg nuevos-cars) (reg scan) (reg nuevo))   
  (asignar antiguo (op vector-ref) (reg nuevos-cdrs) (reg scan))   
  (asignar reubicar-continuar (etiqueta update-cdr))   
  (goto (etiqueta reubicar-antiguo-resultado-en-nuevo))   
  
update-cdr   
  (ejecutar   
   (op vector-set!) (reg nuevos-cdrs) (reg scan) (reg nuevo))   
  (asignar scan (op +) (reg scan) (const 1))   
  (goto (etiqueta gc-loop))

La subrutina relocate-old-result-in-new reubica objetos de la siguiente manera: Si el objeto a ser reubicado (apuntado por old ) no es un par, entonces devolvemos el mismo puntero al objeto sin cambios (en new ). (Por ejemplo, podemos estar escaneando un par cuyo auto es el número 4. Si representamos el auto por n4 , como se describe en la sección  [5.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.3.1) , entonces queremos que el puntero del auto ``reubicado'' siga siendo n4 .) De lo contrario, debemos realizar la reubicación. Si la posición del auto del par a ser reubicado contiene una etiqueta de corazón roto, entonces el par de hecho ya ha sido movido, por lo que recuperamos la dirección de reenvío (de la posición cdr del corazón roto) y la devolvemos en new . Si el puntero en old apunta a un par aún no movido, entonces movemos el par a la primera celda libre en la memoria new (apuntada por free ) y configuramos el corazón roto almacenando una etiqueta de corazón roto y una dirección de reenvío en la ubicación anterior. Reubicar-antiguo-resultado-en-nuevo utiliza un registrooldcr para sujetar el carro o el cdr del objeto apuntado por old . [18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_764)

relocate-old-result-in-new   
  (test (op pointer-to-pair?) (reg old))   
  (branch (label pair))   
  (assign new (reg old))   
  (goto (reg relocate-continue))   
pair   
  (assign oldcr (op vector-ref) (reg the-cars) (reg old))   
  (test (op broken-heart?) (reg oldcr))   
  (branch (label Already-moved))   
  (assign new (reg free))  *; nueva ubicación para el par*   
  *;; Actualizar   puntero libre*  
.   (assign free (op +) (reg free) (const 1))   
  *;; Copiar el  auto  y  el cdr  a la nueva memoria.*  
  (perform (op vector-set!)   
           (reg new-cars) (reg new) (reg oldcr))   
  (assign oldcr (op vector-ref) (reg the-cdrs) (reg old))   
  (perform (op vector-set!)   
           (reg new-cdrs) (reg new) (reg oldcr))   
  *;; Construye el corazón roto.*  
  (perform (op vector-set!)   
           (reg the-cars) (reg old) (const broken-heart))   
  (perform   
   (op vector-set!) (reg the-cdrs) (reg old) (reg new))   
  (goto (reg relocate-continue))   
ya movido   
  (assign new (op vector-ref) (reg the-cdrs) (reg old))   
  (goto (reg relocate-continue))

Al final del proceso de recolección de basura, intercambiamos el papel de las memorias antiguas y nuevas intercambiando punteros: intercambiamos the-cars con new-cars y the-cdrs con new-cdrs . Entonces estaremos listos para realizar otra recolección de basura la próxima vez que se agote la memoria.

gc-flip   
  (asignar temp (reg los-cdrs))   
  (asignar los-cdrs (reg nuevos-cdrs))   
  (asignar nuevos-cdrs (reg temp))   
  (asignar temp (reg los-autos))   
  (asignar los-autos (reg nuevos-autos))   
  (asignar nuevos-autos (reg temp))

[5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_744) Podríamos representar la memoria como listas de elementos. Sin embargo, el tiempo de acceso no sería independiente del índice, ya que acceder alelemento*nn*- 1operacionescdr

[6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_745) Para completar, deberíamos especificar unade creación de vectoresque construya vectores. Sin embargo, en la presente aplicación utilizaremos vectores únicamente para modelar divisiones fijas de la memoria de la computadora.

[7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_747) Esto es exactamente lo mismoLa idea de los "datos etiquetados" que introdujimos en el capítulo 2 para tratar con operaciones genéricas, sin embargo, aquí los tipos de datos se incluyen en el nivel de la máquina primitiva en lugar de construirse mediante el uso de listas.

[8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_748) La información de tipo se puede codificar de diversas maneras, dependiendo de los detalles de la máquina en la que se va a implementar el sistema Lisp. La eficiencia de ejecución de los programas Lisp dependerá en gran medida de la inteligencia con la que se haga esta elección, pero es difícil formular reglas de diseño generales para tomar buenas decisiones. La forma más directa de implementar punteros tipados es asignar un conjunto fijo de bits en cada puntero para que sea un*campo de tipo* que codifica el tipo de datos. Las preguntas importantes que se deben abordar al diseñar una representación de este tipo incluyen las siguientes: ¿Cuántos bits de tipo se requieren? ¿Qué tan grandes deben ser los índices vectoriales? ¿Con qué eficiencia se pueden usar las instrucciones de máquina primitivas para manipular los campos de tipo de los punteros? Se dice que las máquinas que incluyen hardware especial para el manejo eficiente de los campos de tipo tienen*arquitecturas etiquetadas* .

[9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_749) Esta decisión sobre laLa representación de números determina si eq? , que prueba la igualdad de punteros, se puede utilizar para probar la igualdad de números. Si el puntero contiene el número mismo, entonces los números iguales tendrán el mismo puntero. Pero si el puntero contiene el índice de una ubicación donde se almacena el número, se garantizará que los números iguales tengan punteros iguales solo si tenemos cuidado de nunca almacenar el mismo número en más de una ubicación.

[10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_750) Esto es como escribir un número como una secuencia de dígitos, excepto que cada «dígito» es un número entre 0 y el número más grande que se puede almacenar en un solo puntero.

[11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_752) Hay otras formas de encontrar espacio de almacenamiento libre. Por ejemplo, podríamos unir todos los pares no utilizados en un*lista libre* . Nuestras ubicaciones libres son consecutivas (y por lo tanto se puede acceder a ellas incrementando un puntero) porque estamos usando un recolector de basura compactador, como veremos en la sección  [5.3.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.3.2) .

[12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_753) Esta es esencialmente la implementación deconsen términos deset-car!yset-cdr!,como se describe en la sección [3.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_3.3.1). La operaciónget-new-pairutilizada en esa implementación se realiza aquí mediante ellibre.

[13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_758) Puede que esto no sea cierto en el futuro, porque las memorias pueden llegar a ser lo suficientemente grandes como para que sea imposible quedarse sin memoria libre durante la vida útil de la computadora. Por ejemplo, hay alrededor de 3 × 1013microsegundos en un año, por lo que si tuviéramos quegastaruna vez por microsegundo, necesitaríamos alrededor de 1015celdas de memoria para construir una máquina que pudiera funcionar durante 30 años sin quedarse sin memoria. Esa cantidad de memoria parece absurdamente grande para los estándares actuales, pero no es físicamente imposible. Por otro lado, los procesadores son cada vez más rápidos y una computadora futura puede tener una gran cantidad de procesadores operando en paralelo en una sola memoria, por lo que puede ser posible utilizar la memoria mucho más rápido de lo que hemos postulado.

Suponemos aquí que la pila se representa como una lista como se describe en la sección  [5.3.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_759) [,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_5.3.1)de modo que los elementos de la pila son accesibles a través del puntero en el registro de la pila.

[15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_760) Esta idea fue inventada y se implementó por primera vezpor Minsky, como parte de la implementación deLisp para el PDP-1 en elLaboratorio de Investigación de Electrónica del MIT. Fue desarrollado posteriormente porFenichel y Yochelson (1969) para su uso en la implementación de Lisp paraEl sistema de tiempo compartido Multics. Más tarde,Baker (1978) desarrolló una versión del método en "tiempo real", que no requiere que el cálculo se detenga durante la recolección de basura. La idea de Baker fue ampliada porHewitt, Lieberman y Moon (véase Lieberman y Hewitt 1983) para aprovechar el hecho de que algunas estructuras son más volátiles y otras son más permanentes.

Una técnica alternativa de recolección de basura comúnmente utilizada es la*método de barrido de marcas* . Consiste en rastrear toda la estructura accesible desde los registros de la máquina y marcar cada par al que llegamos. Luego escaneamos toda la memoria y cualquier ubicación que no esté marcada se "barre" como basura y se pone a disposición para su reutilización.Se puede encontrar una discusión del método de barrido de marcas en Allen 1978.

El algoritmo Minsky-Fenichel-Yochelson es el algoritmo dominante en uso para sistemas con gran memoria porque examina solo la parte útil de la memoria. Esto contrasta con el método de barrido de marcas, en el que la fase de barrido debe revisar toda la memoria. Una segunda ventaja del método de parada y copia es que es un*Compactación* del recolector de basura. Es decir, al final de la fase de recolección de basura, los datos útiles se habrán movido a ubicaciones de memoria consecutivas, y se habrán comprimido todos los pares de basura. Esto puede ser una consideración de rendimiento extremadamente importante en máquinas con memoria virtual, en las que los accesos a direcciones de memoria muy separadas pueden requerir operaciones de paginación adicionales.

[16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_762) Esta lista de registros no incluye los registros utilizados por el sistema de asignación de almacenamiento:root,the-cars,the-cdrsy los demás registros que se presentarán en esta sección.

[17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_763) El término*El término corazón roto* fue acuñado por David Cressey, quien escribió un recolector de basura paraMDL, un dialecto de Lisp desarrollado en el MIT a principios de la década de 1970.

[18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_764) El recolector de basura utiliza el predicado de bajo nivelpointer-to-pair?en lugar de la operación de estructura de listapair?porque en un sistema real puede haber varias cosas que se tratan como pares para fines de recolección de basura. Por ejemplo, en un sistema Scheme que cumple con el estándar IEEE, un objeto de procedimiento puede implementarse como un tipo especial de ``pair'' que no satisface elpredicadopairpointer-to-pair?puede implementarse comopair?.

[**5.4 El evaluador de control explícito**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_5.4)

En la sección  [5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1) vimos cómo transformar programas simples de Scheme en descripciones de máquinas de registros. Ahora realizaremos esta transformación en un programa más complejo, el evaluador metacircular de las secciones [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) - [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4) , que muestra cómo se puede describir el comportamiento de un intérprete de Scheme en términos de los procedimientos eval y apply . El *evaluador de control explícito* que desarrollamos en esta sección muestra cómo se pueden describir los mecanismos subyacentes de llamada a procedimientos y paso de argumentos utilizados en el proceso de evaluación en términos de operaciones sobre registros y pilas. Además, el evaluador de control explícito puede servir como una implementación de un intérprete de Scheme, escrito en un lenguaje que es muy similar al lenguaje de máquina nativo de las computadoras convencionales. El evaluador puede ser ejecutado por el simulador de máquina de registros de la sección  [5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2) . Alternativamente, se puede utilizar como punto de partida para construir una implementación en lenguaje de máquina de un evaluador de Scheme, o incluso unMáquina de propósito especial para evaluar expresiones de Scheme. La figura  [5.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.16) muestra una implementación de hardware de este tipo: un chip de silicio que actúa como evaluador de Scheme. Los diseñadores del chip comenzaron con las especificaciones de la ruta de datos y del controlador para una máquina de registro similar al evaluador descrito en esta sección y utilizaron programas de automatización del diseño para construir el diseño del circuito integrado. [19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_765)

**[Registros y operaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_766)**

Al diseñar el evaluador de control explícito, debemos especificar las operaciones que se utilizarán en nuestra máquina de registros. Describimos el evaluador metacircular en términos de sintaxis abstracta, utilizando procedimientos como quoted? y make-procedure . Al implementar la máquina de registros, podríamos expandir estos procedimientos en secuencias de operaciones de memoria de estructura de lista elementales e implementar estas operaciones en nuestra máquina de registros. Sin embargo, esto haría que nuestro evaluador fuera muy largo, oscureciendo la estructura básica con detalles. Para aclarar la presentación, incluiremos como operaciones primitivas de la máquina de registros los procedimientos de sintaxis dados en la sección  [4.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.2) y los procedimientos para representar entornos y otros datos de tiempo de ejecución dados en las secciones  [4.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.3) y  [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4) . Para especificar completamente un evaluador que pudiera programarse en un lenguaje de máquina de bajo nivel o implementarse en hardware, reemplazaríamos estas operaciones por operaciones más elementales, utilizando la implementación de estructura de lista que describimos en la sección  [5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.3) .

|  |
| --- |
| Imagen digital de un edificio  El contenido generado por IA puede ser incorrecto. |
|  |
| **Figura 5.16:**   Una implementación de chip de silicio de un evaluador para Scheme. |

Nuestra máquina de registros evaluadores de Scheme incluye una pila y siete registros: exp , env , val , continue , proc , argl y unev . Exp se utiliza para almacenar la expresión que se va a evaluar y env contiene el entorno en el que se realizará la evaluación. Al final de una evaluación, val contiene el valor obtenido al evaluar la expresión en el entorno designado. El registro continue se utiliza para implementar la recursión, como se explica en la sección  [5.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1.4) . (El evaluador debe llamarse a sí mismo de forma recursiva, ya que evaluar una expresión requiere evaluar sus subexpresiones). Los registros proc , argl y unev se utilizan para evaluar combinaciones.

No proporcionaremos un diagrama de ruta de datos para mostrar cómo se conectan los registros y las operaciones del evaluador, ni daremos la lista completa de operaciones de la máquina. Estas están implícitas en el controlador del evaluador, que se presentará en detalle.

**[5.4.1 El núcleo del evaluador de control explícito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.4.1)**

El elemento central del evaluador es la secuencia de instrucciones que comienza en eval-dispatch . Esto corresponde al procedimiento eval del evaluador metacircular descrito en la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) . Cuando el controlador comienza en eval-dispatch , evalúa la expresión especificada por exp en el entorno especificado por env . Cuando se completa la evaluación, el controlador irá al punto de entrada almacenado en continue , y el registro val contendrá el valor de la expresión. Al igual que con el metacircular eval , la estructura de eval-dispatch es un análisis de caso sobre el tipo sintáctico de la expresión que se va a evaluar. [20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_767)

eval-dispatch   
  (prueba (op ¿autoevaluación?) (reg exp))   
  (rama (etiqueta ev-autoevaluación))   
  (prueba (op ¿variable?) (reg exp))   
  (rama (etiqueta ev-variable))   
  (prueba (op ¿entrecomillado?) (reg exp))   
  (rama (etiqueta ev-entrecomillado))   
  (prueba (op ¿asignación?) (reg exp))   
  (rama (etiqueta ev-asignación))   
  (prueba (op ¿definición?) (reg exp))   
  (rama (etiqueta ev-definición))   
  (prueba (op ¿si?) (reg exp))   
  (rama (etiqueta ev-si))   
  (prueba (op ¿lambda?) (reg exp))   
  (rama (etiqueta ev-lambda))   
  (prueba (op ¿comenzar?) (reg exp))   
  (rama (etiqueta ev-comenzar))   
  (prueba (op ¿aplicación?) (reg exp))   
  (rama (etiqueta ev-application))   
  (goto (etiqueta tipo-de-expresión-desconocido))

**[Evaluación de expresiones simples](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_768)**

Los números y cadenas (que se autoevalúan), las variables, las comillas y las expresiones lambda no tienen subexpresiones para evaluar. Para estos, el evaluador simplemente coloca el valor correcto en el registro val y continúa la ejecución en el punto de entrada especificado por continue . La evaluación de expresiones simples se realiza mediante el siguiente código de controlador:

ev-self-eval   
  (asignar val (reg exp))   
  (ir a (reg continuar))  
variable-ev   
  (asignar val (op valor-variable-buscada) (reg exp) (reg env))   
  (goto (reg continue))  
ev-quoted   
  (assign val (op texto-de-la-cita) (reg exp))   
  (goto (reg continue))  
ev-lambda   
  (assign unev (op parámetros-lambda) (reg exp))   
  (assign exp (op cuerpo-lambda) (reg exp))   
  (assign val (op procedimiento-make)   
              (reg unev) (reg exp) (reg env))   
  (goto (reg continue))

Observe cómo ev-lambda usa los registros unev y exp para contener los parámetros y el cuerpo de la expresión lambda para que puedan pasarse a la operación make-procedure , junto con el entorno en env .

**[Evaluación de aplicaciones de procedimientos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_769)**

Una aplicación de procedimiento se especifica mediante una combinación que contiene un operador y operandos. El operador es una subexpresión cuyo valor es un procedimiento y los operandos son subexpresiones cuyos valores son los argumentos a los que se debe aplicar el procedimiento. El metacircular eval maneja las aplicaciones llamándose a sí mismo recursivamente para evaluar cada elemento de la combinación y luego pasando los resultados a apply , que realiza la aplicación del procedimiento real. El evaluador de control explícito hace lo mismo; estas llamadas recursivas se implementan mediante instrucciones goto , junto conUso de la pila para guardar registros que se restaurarán después de que la llamada recursiva regrese. Antes de cada llamada, tendremos cuidado de identificar qué registros deben guardarse (porque sus valores serán necesarios más adelante). [21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_770)

Comenzamos la evaluación de una aplicación evaluando el operador para producir un procedimiento, que luego se aplicará a los operandos evaluados. Para evaluar el operador, lo movemos al registro exp y vamos a eval-dispatch . El entorno en el registro env ya es el correcto en el que evaluar el operador. Sin embargo, guardamos env porque lo necesitaremos más adelante para evaluar los operandos. También extraemos los operandos en unev y lo guardamos en la pila. Configuramos continue para que eval-dispatch se reanude en ev-appl-did-operator después de que se haya evaluado el operador. Primero, sin embargo, guardamos el valor anterior de continue , que le dice al controlador dónde continuar después de la aplicación.

ev-application   
  (guardar continuar)   
  (guardar env)   
  (asignar unev (operandos op) (reg exp))   
  (guardar unev)   
  (asignar exp (operador op) (reg exp))   
  (asignar continuar (etiqueta ev-appl-did-operator))   
  (goto (etiqueta eval-dispatch))

Al regresar de evaluar la subexpresión del operador, procedemos a evaluar los operandos de la combinación y a acumular los argumentos resultantes en una lista, contenida en argl . Primero restauramos los operandos no evaluados y el entorno. Inicializamos argl a una lista vacía. Luego asignamos al registro proc el procedimiento que se produjo al evaluar el operador. Si no hay operandos, vamos directamente a apply-dispatch . De lo contrario, guardamos proc en la pila e iniciamos el bucle de evaluación de argumentos: [22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_771)

ev-appl-did-operator   
  (restaurar unev)                   *; los operandos*  
  (restaurar env)   
  (asignar argl (op lista-de-argumentos-vacía))   
  (asignar proc (reg val))          *; el operador*  
  (prueba (op ¿sin-operandos?) (reg unev))   
  (rama (etiqueta aplicar-enviar))   
  (guardar proc)

Cada ciclo del bucle de evaluación de argumentos evalúa un operando de la lista en unev y acumula el resultado en argl . Para evaluar un operando, lo colocamos en el registro exp y vamos a eval-dispatch , después de configurar continue para que la ejecución se reanude con la fase de acumulación de argumentos. Pero primero guardamos los argumentos acumulados hasta ahora (guardados en argl ), el entorno (guardado en env ) y los operandos restantes a evaluar (guardados en unev ). Se crea un caso especial para la evaluación del último operando, que se maneja en ev-appl-last-arg .

ev-appl-operand-loop   
  (guardar argl)   
  (asignar exp (op primer operando) (reg unev))   
  (prueba (op último operando?) (reg unev))   
  (rama (etiqueta ev-appl-último-arg))   
  (guardar env)   
  (guardar unev)   
  (asignar continuar (etiqueta ev-appl-acumular-arg))   
  (goto (etiqueta eval-dispatch))

Cuando se ha evaluado un operando, el valor se acumula en la lista contenida en argl . Luego, el operando se elimina de la lista de operandos no evaluados en unev , y la evaluación de argumentos continúa.

ev-appl-accumulate-arg   
  (restaurar unev)   
  (restaurar env)   
  (restaurar argl)   
  (asignar argl (op argumento-adjunto) (reg val) (reg argl))   
  (asignar unev (op operandos-resto) (reg unev))   
  (goto (etiqueta bucle-de-operandos-ev-appl))

La evaluación del último argumento se maneja de manera diferente. No es necesario guardar el entorno ni la lista de operandos no evaluados antes de ir a eval-dispatch , ya que no serán necesarios después de que se evalúe el último operando. Por lo tanto, regresamos de la evaluación a un punto de entrada especial ev-appl-accum-last-arg , que restaura la lista de argumentos, acumula el nuevo argumento, restaura el procedimiento guardado y continúa para ejecutar la aplicación. [23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_772)

ev-appl-last-arg   
  (asignar continuar (etiqueta ev-appl-accum-last-arg))   
  (goto (etiqueta eval-dispatch))   
ev-appl-accum-last-arg   
  (restaurar argl)   
  (asignar argl (op adjoin-arg) (reg val) (reg argl))   
  (restaurar proc)   
  (goto (etiqueta apply-dispatch))

Los detalles del bucle de evaluación de argumentos determinan el orden en el que el intérprete evalúa los operandos de una combinación (por ejemplo, de izquierda a derecha o de derecha a izquierda; consulte el ejercicio  [3.8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.8) ). Este orden no lo determina el evaluador metacircular, que hereda su estructura de control del Scheme subyacente en el que se implementa. [24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_773) Debido a que el selector del primer operando (usado en ev-appl-operand-loop para extraer operandos sucesivos de unev ) se implementa como car y el selector de operandos restantes se implementa como cdr , el evaluador de control explícito evaluará los operandos de una combinación en orden de izquierda a derecha.

**[Solicitud de procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_774)**

El punto de entrada apply-dispatch corresponde al procedimiento apply del evaluador metacircular. Cuando llegamos a apply-dispatch , el registro proc contiene el procedimiento a aplicar y argl contiene la lista de argumentos evaluados a los que debe aplicarse. El valor guardado de continue (originalmente pasado a eval-dispatch y guardado en ev-application ), que indica dónde regresar con el resultado de la aplicación del procedimiento, está en la pila. Cuando se completa la aplicación, el controlador transfiere al punto de entrada especificado por el continue guardado , con el resultado de la aplicación en val . Al igual que con el apply metacircular , hay dos casos a considerar. O bien el procedimiento a aplicar es un primitivo o es un procedimiento compuesto.

aplicar-enviar   
  (prueba (op procedimiento-primitivo?) (reg proc))   
  (rama (etiqueta aplicación-primitiva))   
  (prueba (op procedimiento-compuesto?) (reg proc))     
  (rama (etiqueta aplicación-compuesto))   
  (goto (etiqueta tipo-procedimiento-desconocido))

Suponemos que cada primitivo se implementa de manera que obtenga sus argumentos de argl y coloque su resultado en val . Para especificar cómo maneja la máquina los primitivos, tendríamos que proporcionar una secuencia de instrucciones del controlador para implementar cada primitivo y hacer que primitive-apply envíe las instrucciones para el primitivo identificado por el contenido de proc . Dado que estamos interesados ​​en la estructura del proceso de evaluación en lugar de los detalles de los primitivos, en su lugar utilizaremos una operación apply-primitive-procedure que aplica el procedimiento en proc a los argumentos en argl . Con el propósito de simular el evaluador con el simulador de la sección  [5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2) utilizamos el procedimiento apply-primitive-procedure , que llama al sistema Scheme subyacente para realizar la aplicación, tal como lo hicimos para el evaluador metacircular en la sección  [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4) . Después de calcular el valor de la aplicación primitiva, restauramos continue y vamos al punto de entrada designado.

aplicación-primitiva   
  (asignar val (op aplicar-procedimiento-primitivo)   
              (reg proc)   
              (reg argl))   
  (restaurar continuar)   
  (ir a (reg continuar))

Para aplicar un procedimiento compuesto, procedemos igual que con el evaluador metacircular. Construimos un marco que vincula los parámetros del procedimiento a los argumentos, utilizamos este marco para extender el entorno que lleva el procedimiento y evaluamos en este entorno extendido la secuencia de expresiones que forma el cuerpo del procedimiento. Ev-sequence , descrita más adelante en la sección  [5.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.2) , maneja la evaluación de la secuencia.

aplicación-compuesta   
  (asignar unev (op parámetros-procedimiento) (reg proc))   
  (asignar env (op entorno-procedimiento) (reg proc))   
  (asignar env (op entorno-extensión)   
              (reg unev) (reg argl) (reg env))   
  (asignar unev (op cuerpo-procedimiento) (reg proc))   
  (goto (etiqueta secuencia-ev))

Compound-apply es el único lugar en el intérprete donde se asigna un nuevo valor al registro env . Al igual que en el evaluador metacircular, el nuevo entorno se construye a partir del entorno que lleva el procedimiento, junto con la lista de argumentos y la lista correspondiente de variables que se deben vincular.

**[5.4.2 Evaluación de secuencias y recursión de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.4.2)**

La parte del evaluador de control explícito en ev-sequence es análoga al procedimiento eval-sequence del evaluador metacircular . Maneja secuencias de expresiones en cuerpos de procedimientos o en expresiones de inicio explícitas .

Las expresiones de inicio explícitas se evalúan colocando la secuencia de expresiones a evaluar en unev , guardando continue en la pila y saltando a ev-sequence .

ev-begin   
  (asignar unev (op begin-actions) (reg exp))   
  (guardar continuar)   
  (goto (etiqueta ev-sequence))

Las secuencias implícitas en los cuerpos de los procedimientos se manejan saltando a ev-sequence desde composite-apply , momento en el cual continue ya está en la pila, habiéndose guardado en ev-application .

Las entradas en ev-sequence y ev-sequence-continue forman un bucle que evalúa sucesivamente cada expresión en una secuencia. La lista de expresiones no evaluadas se guarda en unev . Antes de evaluar cada expresión, verificamos si hay expresiones adicionales para evaluar en la secuencia. Si es así, guardamos el resto de las expresiones no evaluadas (guardadas en unev ) y el entorno en el que deben evaluarse (guardado en env ) y llamamos a eval-dispatch para evaluar la expresión. Los dos registros guardados se restauran al regresar de esta evaluación, en ev-sequence-continue .

La expresión final en la secuencia se maneja de manera diferente, en el punto de entrada ev-sequence-last-exp . Dado que no hay más expresiones para evaluar después de esta, no necesitamos guardar unev o env antes de ir a eval-dispatch . El valor de toda la secuencia es el valor de la última expresión, por lo que después de la evaluación de la última expresión no queda nada por hacer excepto continuar en el punto de entrada que se encuentra actualmente en la pila (que fue guardado por ev-application o ev-begin ). En lugar de configurar continue para organizar que eval-dispatch regrese aquí y luego restaurar continue desde la pila y continuar en ese punto de entrada, restauramos continue desde la pila antes de ir a eval-dispatch , de modo que eval-dispatch continúe en ese punto de entrada después de evaluar la expresión.

ev-sequence   
  (assign exp (op first-exp) (reg unev))   
  (test (op last-exp?) (reg unev))   
  (branch (label ev-sequence-last-exp))   
  (save unev)   
  (save env)   
  (assign continue (label ev-sequence-continue))   
  (goto (label eval-dispatch))   
ev-sequence-continue   
  (restore env)   
  (restore unev)   
  (assign unev (op rest-exps) (reg unev))   
  (goto (label ev-sequence))   
ev-sequence-last-exp   
  (restore continue)   
  (goto (label eval-dispatch))

**[Recursión de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_775)**

En el capítulo 1 dijimos que el proceso descrito por un procedimiento como

(define (sqrt-iter suposición x)   
  (si (¿suficientemente bueno? suposición x)   
      suposición   
      (sqrt-iter (mejora suposición x)   
                 x)))

es un proceso iterativo. Aunque el procedimiento es sintácticamente recursivo (definido en términos de sí mismo), no es lógicamente necesario que un evaluador guarde información al pasar de una llamada a sqrt-iter a la siguiente. [25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_776) Un evaluador que puede ejecutar un procedimiento como sqrt-iter sin requerir un aumento de almacenamiento a medida que el procedimiento continúa llamándose a sí mismo se denomina*evaluador recursivo de cola* .La implementación metacircular del evaluador en el capítulo 4 no especifica si el evaluador es recursivo en la cola, porque ese evaluador hereda su mecanismo para guardar el estado del Scheme subyacente. Sin embargo, con el evaluador de control explícito, podemos rastrear el proceso de evaluación para ver cuándo las llamadas a procedimientos causan una acumulación neta de información en la pila.

Nuestro evaluador es recursivo de cola, porque para evaluar la expresión final de una secuencia, pasamos directamente a eval-dispatch sin guardar ninguna información en la pila. Por lo tanto, evaluar la expresión final de una secuencia (incluso si se trata de una llamada a un procedimiento, como en sqrt-iter , donde la expresión if , que es la última expresión en el cuerpo del procedimiento, se reduce a una llamada a sqrt-iter ) no hará que se acumule ninguna información en la pila. [26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_777)

Si no pensáramos en aprovechar el hecho de que no era necesario guardar información en este caso, podríamos haber implementado eval-sequence tratando todas las expresiones en una secuencia de la misma manera: guardando los registros, evaluando la expresión, volviendo a restaurar los registros y repitiendo esto hasta que se hayan evaluado todas las expresiones: [27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_778)

ev-sequence   
  (prueba (op no-more-exps?) (reg unev))   
  (rama (etiqueta ev-sequence-end))   
  (asignar exp (op first-exp) (reg unev))   
  (guardar unev)   
  (guardar entorno)   
  (asignar continuar (etiqueta ev-sequence-continue))   
  (goto (etiqueta eval-dispatch))   
ev-sequence-continue   
  (restaurar entorno)   
  (restaurar unev)   
  (asignar unev (op rest-exps) (reg unev))   
  (goto (etiqueta ev-sequence))   
ev-sequence-end   
  (restaurar continuar)   
  (goto (reg continue))

Esto puede parecer un cambio menor en nuestro código anterior para la evaluación de una secuencia: la única diferencia es que pasamos por el ciclo de guardar-restaurar para la última expresión en una secuencia así como para las demás. El intérprete seguirá dando el mismo valor para cualquier expresión. Pero este cambio es fatal para la implementación recursiva de cola, porque ahora debemos regresar después de evaluar la expresión final en una secuencia para deshacer los guardados de registros (inútiles). Estos guardados adicionales se acumularán durante un anidamiento de llamadas a procedimientos. En consecuencia, los procesos como sqrt-iter requerirán espacio proporcional al número de iteraciones en lugar de requerir espacio constante. Esta diferencia puede ser significativa. Por ejemplo,Con la recursión de cola, se puede expresar un bucle infinito utilizando únicamente el mecanismo de llamada a procedimiento:

(definir (contar n)   
  (nueva línea)   
  (mostrar n)   
  (contar (+ n 1)))

Sin recursión de cola, dicho procedimiento eventualmente se quedaría sin espacio en la pila y expresar una verdadera iteración requeriría algún mecanismo de control distinto a la llamada al procedimiento.

**[5.4.3 Condicionales, asignaciones y definiciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.4.3)**

Al igual que con el evaluador metacircular, las formas especiales se manejan evaluando selectivamente fragmentos de la expresión. Para una expresión if , debemos evaluar el predicado y decidir, en función del valor del predicado, si evaluar el consecuente o la alternativa.

Antes de evaluar el predicado, guardamos la propia expresión if para poder extraer después el consecuente o la alternativa. También guardamos el entorno, que necesitaremos más adelante para evaluar el consecuente o la alternativa, y guardamos continue , que necesitaremos más adelante para volver a la evaluación de la expresión que está esperando el valor del if .

ev-if   
  (guardar exp)                     *; guardar expresión para más tarde*  
  (guardar env)   
  (guardar continue)   
  (asignar continue (etiqueta ev-if-decide))   
  (asignar exp (op if-predicate) (reg exp))   
  (goto (etiqueta eval-dispatch))   *; evaluar el predicado*

Cuando volvemos de evaluar el predicado, comprobamos si era verdadero o falso y, según el resultado, colocamos el consecuente o la alternativa en exp antes de pasar a eval-dispatch . Observe que restaurar env y continue aquí configura eval-dispatch para que tenga el entorno correcto y continúe en el lugar correcto para recibir el valor de la expresión if .

ev-if-decide   
  (restaurar continuar)   
  (restaurar entorno)   
  (restaurar exp)   
  (prueba (op ¿verdadero?) (reg val))   
  (rama (etiqueta ev-if-consequent))   
  
ev-if-alternative   
  (asignar exp (op if-alternative) (reg exp))   
  (goto (etiqueta eval-dispatch))   
ev-if-consequent   
  (asignar exp (op if-consequent) (reg exp))   
  (goto (etiqueta eval-dispatch))

**[Tareas y definiciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_779)**

Las asignaciones se manejan mediante ev-assignment , al que se accede desde eval-dispatch con la expresión de asignación en exp . El código en ev-assignment primero evalúa la parte de valor de la expresión y luego instala el nuevo valor en el entorno. Se supone que set-variable-value! está disponible como una operación de máquina.

ev-assignment   
  (assign unev (op variable-asignación) (reg exp))   
  (save unev)                    *; guardar variable para más tarde*  
  (assign exp (op valor-asignación) (reg exp))   
  (save env)   
  (save continue)   
  (assign continue (label ev-assignment-1))   
  (goto (label eval-dispatch))   *; evaluar el valor de la asignación*  
ev-assignment-1   
  (restore continue)   
  (restore env)   
  (restore unev)   
  (perform   
   (op set-variable-value!) (reg unev) (reg val) (reg env))   
  (assign val (const ok))   
  (goto (reg continue))

Las definiciones se manejan de manera similar:

ev-definition   
  (assign unev (op variable-definición) (reg exp))   
  (save unev)                    *; guardar variable para más tarde*  
  (assign exp (op valor-definición) (reg exp))   
  (save env)   
  (save continue)   
  (assign continue (label ev-definition-1))   
  (goto (label eval-dispatch))   *; evaluar el valor de la definición*  
ev-definition-1   
  (restore continue)   
  (restore env)   
  (restore unev)   
  (perform   
   (op variable-definición!) (reg unev) (reg val) (reg env))   
  (assign val (const ok))   
  (goto (reg continue))

**Ejercicio 5.23.**  Amplíe el evaluador para que pueda manejar expresiones derivadas como cond , let , etc. (sección  [4.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.2) ). Puede hacer trampa y suponer que los transformadores de sintaxis como cond->if están disponibles como operaciones de máquina. [28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_781)

**Ejercicio 5.24.**  Implemente cond como una nueva forma especial básica sin reducirla a if . Deberá construir un bucle que pruebe los predicados de cláusulas cond sucesivas hasta encontrar una que sea verdadera y luego usar ev-sequence para evaluar las acciones de la cláusula.

**Ejercicio 5.25.**  Modifique el evaluador para que utilice la evaluación de orden normal, basándose en el evaluador perezoso de la sección  [4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.2) .

**[5.4.4 Ejecución del evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.4.4)**

Con la implementación del evaluador de control explícito llegamos al final de un desarrollo, iniciado en el capítulo 1, en el que hemos explorado sucesivamente modelos más precisos del proceso de evaluación. Empezamos con el modelo de sustitución relativamente informal, y luego lo ampliamos en el capítulo 3 al modelo de entorno, que nos permitió tratar con el estado y el cambio. En el evaluador metacircular del capítulo 4, utilizamos el propio Scheme como lenguaje para hacer más explícita la estructura del entorno construida durante la evaluación de una expresión. Ahora, con las máquinas de registro, hemos examinado en profundidad los mecanismos del evaluador para la gestión del almacenamiento, el paso de argumentos y el control. En cada nuevo nivel de descripción, hemos tenido que plantear cuestiones y resolver ambigüedades que no eran evidentes en el tratamiento anterior, menos preciso, de la evaluación. Para comprender el comportamiento del evaluador de control explícito, podemos simularlo y monitorear su rendimiento.

Instalaremos un bucle de controlador en nuestra máquina evaluadora. Este cumple la función del procedimiento  de bucle de controlador de la sección [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4) . El evaluador imprimirá repetidamente un mensaje, leerá una expresión, evaluará la expresión yendo a eval-dispatch e imprimirá el resultado. Las siguientes instrucciones forman el comienzo de la secuencia del controlador del evaluador de control explícito: [29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_784)

leer-evaluar-imprimir-bucle   
  (ejecutar (op inicializar-pila))   
  (ejecutar   
   (op solicitar-entrada) (const ";;; Entrada EC-Eval:"))   
  (asignar exp (op leer))   
  (asignar env (op obtener-entorno-global))   
  (asignar continuar (etiqueta imprimir-resultado))   
  (goto (etiqueta enviar-evaluación))  
imprimir-resultado   
  (ejecutar   
   (op anunciar-salida) (const ";;; valor EC-Eval:"))   
  (ejecutar (op usuario-imprimir) (reg val))   
  (ir a (etiqueta leer-eval-imprimir-bucle))

Cuando encontramos un error en un procedimiento (como el «error de tipo de procedimiento desconocido» indicado en apply-dispatch ), imprimimos un mensaje de error y regresamos al bucle del controlador. [30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_785)

tipo-de-expresión-desconocido   
  (asignar val (const error-de-tipo-de-expresión-desconocido))   
  (ir a (etiqueta error-de-señal))  
tipo-procedimiento-desconocido   
  (restaurar continuar)     *; limpiar pila (desde  aplicar-enviar )*  
  (asignar val (const error-tipo-procedimiento-desconocido))   
  (ir a (etiqueta error-señal))  
señal-error   
  (ejecutar (op usuario-imprimir) (reg val))   
  (ir a (etiqueta leer-evaluar-imprimir-bucle))

Para los fines de la simulación, inicializamos la pila cada vez que pasamos por el bucle del controlador, ya que podría no estar vacía después de que un error (como una variable no definida) interrumpa una evaluación. [31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_786)

Si combinamos todos los fragmentos de código presentados en las secciones [5.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.1) - [5.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.4) , podemos crear un modelo de máquina evaluadora que podemos ejecutar utilizando el simulador de máquina de registro de la sección  [5.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2) .

(define eceval   
  (make-machine   
   '(exp env val proc argl continue unev)   
   eceval-operations   
  '(   
    read-eval-print-loop   
      < *controlador de la máquina completa como se indica arriba* >   
   )))

Debemos definir procedimientos de Scheme para simular las operaciones utilizadas como primitivas por el evaluador. Son los mismos procedimientos que utilizamos para el evaluador metacircular en la sección  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) , junto con algunos procedimientos adicionales definidos en notas a pie de página a lo largo de la sección  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4) .

(define eceval-operations   
  (list (list 'autoevaluating? autoevaluating)   
        *<lista completa de operaciones para la máquina eceval>* ))

Finalmente, podemos inicializar el entorno global y ejecutar el evaluador:

(define el entorno global (configuración del entorno))   
  
(inicia eceval)   
*;;; Entrada de EC-Eval:*  
(define (append x y)   
  (if (null? x)   
      y   
      (cons (car x)   
            (append (cdr x) y))))   
*;;; Valor de EC-Eval:*   
*ok*   
*;;; Entrada de EC-Eval:*  
(append '(a b c) '(d e f))   
*;;; Valor de EC-Eval:*   
*(a b c d e f)*

Por supuesto, evaluar expresiones de esta manera llevará mucho más tiempo que si las hubiéramos escrito directamente en Scheme, debido a los múltiples niveles de simulación involucrados. Nuestras expresiones son evaluadas por la máquina evaluadora de control explícito, que está siendo simulada por un programa Scheme, que a su vez está siendo evaluado por el intérprete Scheme.

**[Seguimiento del desempeño del evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_787)**

La simulación puede ser una herramienta poderosa para guiar la implementación de evaluadores. Las simulaciones facilitan no sólo la exploración de variaciones del diseño de la máquina de registro, sino también el monitoreo del desempeño del evaluador simulado. Por ejemplo, un factor importante en el desempeño es la eficiencia con la que el evaluador utiliza la pila. Podemos observar la cantidad de operaciones de pila requeridas para evaluar varias expresiones definiendo la máquina de registro del evaluador con la versión del simulador que recopila estadísticas sobre el uso de la pila (sección  [5.2.4 ) y agregando una instrucción en el punto de entrada de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2.4) resultados de impresión del evaluador para imprimir las estadísticas:

print-result   
  (perform (op print-stack-statistics)) *; instrucción agregada*  
  (perform   
   (op Announce-output) (const ";;; EC-Eval value:"))   
  ...  *; igual que antes*

Las interacciones con el evaluador ahora se ven así:

*;;; Entrada de EC-Eval:*  
(define (factorial n)   
  (si (= n 1)   
      1   
      (\* (factorial (- n 1)) n)))   
*(total-pushes = 3 maximum-depth = 3)*   
*;;; Valor de EC-Eval:*   
*ok*   
*;;; Entrada de EC-Eval:*  
(factorial 5)   
*(total-pushes = 144 maximum-depth = 28)*   
*;;; Valor de EC-Eval:*   
*120*

Tenga en cuenta que el bucle del controlador del evaluador reinicializa la pila al inicio de cada interacción, de modo que las estadísticas impresas se referirán únicamente a las operaciones de pila utilizadas para evaluar la expresión anterior.

**Ejercicio 5.26.**  Utilice la pila monitoreada para explorar la propiedad recursiva de cola del evaluador (sección  [5.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.2) ). Inicie el evaluador y defina el procedimiento  factorial iterativo de la sección [1.2.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.1) :

(define (factorial n)   
  (define (iter producto contador)   
    (si (> contador n)   
        producto   
        (iter (\* contador producto)   
              (+ contador 1))))   
  (iter 1 1))

Ejecute el procedimiento con algunos valores pequeños de *n* . Registre la profundidad máxima de la pila y la cantidad de empujes necesarios para calcular *n* ! para cada uno de estos valores.

a. Descubrirás que la profundidad máxima requerida para evaluar *n* ! es independiente de *n* . ¿Cuál es esa profundidad?

b. Determine a partir de sus datos una fórmula en términos de *n* para el número total de operaciones de inserción utilizadas para evaluar *n* ! para cualquier *n* > 1. Tenga en cuenta que el número de operaciones utilizadas es una función lineal de *n* y, por lo tanto, está determinado por dos constantes.

**Ejercicio 5.27.**  A modo de comparación con el ejercicio  [5.26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.26) , explore el comportamiento del siguiente procedimiento para calcular factoriales de forma recursiva:

(define (factorial n)   
  (si (= n 1)   
      1   
      (\* (factorial (- n 1)) n)))

Al ejecutar este procedimiento con la pila monitoreada, determine, como una función de *n* , la profundidad máxima de la pila y la cantidad total de empujes utilizados para evaluar *n* ! para *n* > 1. (Nuevamente, estas funciones serán lineales). Resuma sus experimentos completando la siguiente tabla con las expresiones apropiadas en términos de *n* :

|  |  |  |
| --- | --- | --- |
|  | Profundidad máxima | Número de pulsaciones |
| Recursivo |  |  |
| factorial |  |  |
| Iterativo |  |  |
| factorial |  |  |
|  |  |  |

La profundidad máxima es una medida de la cantidad de espacio utilizado por el evaluador para realizar el cálculo, y el número de impulsos se correlaciona bien con el tiempo requerido.

**Ejercicio 5.28.**  Modifique la definición del evaluador modificando eval-sequence como se describe en la sección  [5.4.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.2) para que el evaluador ya no sea recursivo en la cola. Vuelva a ejecutar los experimentos de los ejercicios  [5.26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.26) y  [5.27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.27) para demostrar que ambas versiones del procedimiento factorial ahora requieren un espacio que crece linealmente con su entrada.

**Ejercicio 5.29.**  Supervisar las operaciones de pila en el cálculo recursivo de Fibonacci en árbol:

(define (fib n)   
  (si (< n 2)   
      n   
      (+ (fib (- n 1)) (fib (- n 2)))))

a. Dé una fórmula en términos de *n* para la profundidad máxima de la pila requerida para calcular *F i b* ( *n* ) para *n* > 2. Sugerencia: En la sección  [1.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_1.2.2) argumentamos que el espacio utilizado por este proceso crece linealmente con *n* .

b. Dé una fórmula para el número total de pulsaciones utilizadas para calcular *F i b* ( *n* ) para *n* > 2. Debería encontrar que la cantidad de pulsaciones (que se correlaciona bien con el tiempo utilizado) crece exponencialmente con *n* . Pista: Sea *S* ( *n* ) la cantidad de pulsaciones utilizadas para calcular *F i b* ( *n* ). Debería poder argumentar que existe una fórmula que expresa *S* ( *n* ) en términos de *S* ( *n* - 1), *S* ( *n* - 2) y una constante fija de ``sobrecarga'' *k* que es independiente de *n* . Dé la fórmula y diga cuál es *k* . Luego demuestre que *S* ( *n* ) se puede expresar como *a F i b* ( *n* + 1) + *b* y dé los valores de *a* y *b* .

**Ejercicio 5.30.**  Actualmente, nuestro evaluador detecta y señala sólo dos tipos de errores: tipos de expresión desconocidos y tipos de procedimiento desconocidos. Otros errores nos sacarán del bucle de lectura-evaluación-impresión del evaluador. Cuando ejecutamos el evaluador utilizando el simulador de máquina de registro, estos errores son detectados por el sistema Scheme subyacente. Esto es análogo a la falla de la computadora cuando un programa de usuario comete un error. [32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_793) Hacer que funcione un sistema de errores real es un proyecto grande, pero vale la pena el esfuerzo para entender lo que está involucrado aquí.

a. Los errores que ocurren en el proceso de evaluación, como un intento de acceder a una variable no vinculada, se pueden detectar modificando la operación de búsqueda para que devuelva un código de condición distinguido, que no puede ser un valor posible de ninguna variable de usuario. El evaluador puede probar este código de condición y luego hacer lo necesario para ir a signal-error . Busque todos los lugares en el evaluador donde sea necesario un cambio de este tipo y corríjalos. Esto es mucho trabajo.

b. Mucho peor es el problema de manejar errores que se señalan al aplicar procedimientos primitivos, como un intento de dividir por cero o un intento de extraer el car de un símbolo. En un sistema de alta calidad escrito profesionalmente, cada aplicación primitiva se verifica por seguridad como parte de la primitiva. Por ejemplo, cada llamada a car podría verificar primero que el argumento sea un par. Si el argumento no es un par, la aplicación devolvería un código de condición distinguido al evaluador, que luego informaría del fallo. Podríamos organizar esto en nuestro simulador de máquina de registro haciendo que cada procedimiento primitivo verifique su aplicabilidad y devuelva un código de condición distinguido apropiado en caso de fallo. Luego, el código de aplicación primitiva en el evaluador puede verificar el código de condición y pasar a signal-error si es necesario. Construya esta estructura y hágala funcionar. Este es un proyecto importante.

[19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_765) Véase Batali et al. 1982 para más información.Información sobre el chip y el método por el cual fue diseñado.

[20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_767) En nuestro controlador, el envío se escribe como una secuencia dede pruebaybifurcación. Alternativamente, podría haberse escrito en un estilo dirigido por datos (y en un sistema real probablemente lo hubiera sido) para evitar la necesidad de realizar pruebas secuenciales y facilitar la definición de nuevos tipos de expresión. Una máquina diseñada para ejecutar Lisp probablemente incluiría unade envío por tipoque ejecutaría de manera eficiente dichos envíos dirigidos por datos.

[21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_770) Este es un punto importante pero sutil en la traducción de algoritmos de un lenguaje procedimental, como Lisp, a un lenguaje de máquina de registros. Como alternativa a guardar solo lo que se necesita, podríamos guardar todos los registros (exceptoval) antes de cada llamada recursiva. Esto se llama una*Disciplina de pila enmarcada* . Esto funcionaría, pero podría ahorrar más registros de los necesarios; esto podría ser una consideración importante en un sistema donde las operaciones de pila son costosas. Guardar registros cuyo contenido no se necesitará más adelante también puede retener datos inútiles que de otro modo podrían ser recolectados por basura, liberando espacio para ser reutilizado.

Agregamos a los procedimientos de estructura de datos del evaluador en la sección  [4.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_771) [los](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.3)dos procedimientos siguientes para manipular listas de argumentos:

(define (lista de argumentos vacía) '())  
  
(definir (adjoin-arg arg arglist)   
  (añadir arglist (lista arg)))

También utilizamos un procedimiento de sintaxis adicional para probar el último operando en una combinación:

(define (¿último operando? operaciones)   
  (¿nulo? (cdr operaciones)))

[23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_772) La optimización del tratamiento del último operandoSe le conoce especialmente como *recursión de cola de Evlis* (verPodríamos ser un poco más eficientes en el bucle de evaluación de argumentos si también convirtiéramos la evaluación del primer operando en un caso especial. Esto nos permitiría posponer la inicialización de argl hasta después de evaluar el primer operando, para evitar guardar argl en este caso. El compilador de la sección  [5.5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5) realiza esta optimización. (Compare el procedimiento  construct-arglist de la sección [5.5.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.3) ).

[El orden de evaluación de los operandos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_773) en el evaluador metacircular está determinado por el orden de evaluación de los argumentos deconsen el procedimientolista-de-valoresde la sección [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1)(ver ejercicio [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.1)).

Vimos en la sección [5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.1) [cómo implementar un proceso de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_776) este tipo con una máquina de registros que no tenía pila; el estado del proceso se almacenaba en un conjunto fijo de registros.

[26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_777) Esta implementación de la recursión de cola enev-sequencees una variedad de una técnica de optimización bien conocida que utilizan muchos compiladores. Al compilar un procedimiento que termina con una llamada a procedimiento, se puede reemplazar la llamada por un salto al punto de entrada del procedimiento llamado. Al incorporar esta estrategia en el intérprete, como hemos hecho en esta sección, se logra una optimización uniforme en todo el lenguaje.

Podemos definirno-more-exps? [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_778) la siguiente manera:

(define (no-más-exps? seq) (nulo? seq))

[En realidad,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_781) esto no es trampa. En una implementación real creada desde cero, utilizaríamos nuestro evaluador de control explícito para interpretar un programa Scheme que realiza transformaciones a nivel de fuente comocond->ifen una fase de sintaxis que se ejecuta antes de la ejecución.

[29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_784) Suponemos aquí quede lecturae impresión están disponibles como operaciones primitivas de máquina, lo que resulta útil para nuestra simulación, pero completamente irreal en la práctica. En realidad, se trata de operaciones extremadamente complejas. En la práctica, se implementarían utilizando operaciones de entrada y salida de bajo nivel, como la transferencia de caracteres individuales hacia y desde un dispositivo.

Para apoyar la operación de obtención de un entorno global definimos

(definir el entorno global (configurar el entorno))  
  
(define (obtiene-el-entorno-global)   
  el-entorno-global)

Hay otros errores que nos gustaría que el intérprete corrigiese, pero no son tan sencillos. Véase el [ejercicio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.30) [5.30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_785) .

[Podríamos realizar la inicialización de la pila solo después de errores ,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_786) pero hacerlo en el bucle del controlador será conveniente para monitorear el rendimiento del evaluador, como se describe a continuación.

[32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_793) Lamentablemente, esta es la situación normal ensistemas de lenguaje convencionales basados ​​en compiladores, como C.En UNIX *TM el* sistema "vuelca el núcleo" y en DOS/Windows *TM se* vuelve catatónico. El Macintosh *TM muestra* una imagen de una bomba explotando y le ofrece la oportunidad de reiniciar la computadora, si tiene suerte.

[**5.5 Compilación**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_sec_5.5)

El evaluador de control explícito de la sección  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4) es una máquina de registro cuyo controlador interpreta programas de Scheme. En esta sección veremos cómo ejecutar programas de Scheme en una máquina de registro cuyo controlador no es un intérprete de Scheme.

La máquina evaluadora de control explícito es universal: puede llevar a cabo cualquier proceso computacional que pueda describirse en Scheme. El controlador del evaluador organiza el uso de sus rutas de datos para realizar el cálculo deseado. Por lo tanto, las rutas de datos del evaluador son universales: son suficientes para realizar cualquier cálculo que deseemos, dado un controlador apropiado. [33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_794)

Las computadoras comerciales de propósito general son máquinas de registro organizadas en torno a una colección de registros y operaciones que constituyen un conjunto universal eficiente y conveniente de rutas de datos. El controlador de una máquina de propósito general es un intérprete de un lenguaje de máquina de registro como el que hemos estado usando. Este lenguaje se llama*idioma nativo* de la máquina, o simplemente*lenguaje de máquina* . Los programas escritos en lenguaje de máquina son secuencias de instrucciones que utilizan las rutas de datos de la máquina. Por ejemplo, elLa secuencia de instrucciones del evaluador de control explícito puede considerarse como un programa en lenguaje de máquina para una computadora de propósito general en lugar de como el controlador de una máquina intérprete especializada.

Existen dos estrategias comunes para salvar la brecha entre los lenguajes de alto nivel y los lenguajes de máquina de registro. El evaluador de control explícito ilustra la estrategia de interpretación. Un intérprete escrito en el lenguaje nativo de una máquina configura la máquina para ejecutar programas escritos en un lenguaje (llamado*lenguaje fuente* ) que puede diferir del lenguaje nativo de la máquina que realiza la evaluación. Los procedimientos primitivos del lenguaje fuente se implementan como una biblioteca de subrutinas escritas en el lenguaje nativo de la máquina dada. Un programa a interpretar (llamado*El programa fuente* se representa como una estructura de datos. El intérprete recorre esta estructura de datos y analiza el programa fuente. Al hacerlo, simula el comportamiento previsto del programa fuente llamando a subrutinas primitivas apropiadas de la biblioteca.

En esta sección, exploramos la estrategia alternativa de *compilación* . Un compilador para un lenguaje fuente y una máquina determinados traduce un programa fuente a un programa equivalente (llamado*programa objeto* ) escrito en el lenguaje nativo de la máquina. El compilador que implementamos en esta sección traduce los programas escritos en Scheme en secuencias de instrucciones que se ejecutarán utilizando las rutas de datos de la máquina del evaluador de control explícito. [34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_795)

En comparación con la interpretación, la compilación puede proporcionar un gran aumento en la eficiencia de la ejecución del programa, como explicaremos a continuación en la descripción general del compilador. Por otro lado, un intérprete proporciona un entorno más potente para el desarrollo y la depuración de programas interactivos, porque el programa fuente que se está ejecutando está disponible en tiempo de ejecución para ser examinado y modificado. Además, debido a que está presente toda la biblioteca de primitivas, se pueden construir nuevos programas y agregarlos al sistema durante la depuración.

En vista de las ventajas complementarias de la compilación y la interpretación, los entornos de desarrollo de programas modernos siguen una estrategia mixta. Los intérpretes de Lisp están generalmente organizados de modo que los procedimientos interpretados y los procedimientos compilados puedan llamarse entre sí. Esto permite a un programador compilar aquellas partes de un programa que se supone que están depuradas, obteniendo así la ventaja de eficiencia de la compilación, mientras se conserva el modo de ejecución interpretativo para aquellas partes del programa que están en el flujo de desarrollo y depuración interactivos. En la sección  [5.5.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.7) , después de haber implementado el compilador, mostraremos cómo interconectarlo con nuestro intérprete para producir un sistema de desarrollo de compilador e intérprete integrado.

**[Una descripción general del compilador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_796)**

Nuestro compilador es muy parecido a nuestro intérprete, tanto en su estructura como en la función que realiza. En consecuencia, los mecanismos que utiliza el compilador para analizar expresiones serán similares a los que utiliza el intérprete. Además, para facilitar la interconexión del código compilado con el interpretado, diseñaremos el compilador para que genere código que obedezca las mismas convenciones deUso del registro como intérprete: el entorno se mantendrá en el registro env , las listas de argumentos se acumularán en argl , un procedimiento a aplicar estará en proc , los procedimientos devolverán sus respuestas en val y la ubicación a la que debe regresar un procedimiento se mantendrá en continue . En general, el compilador traduce un programa fuente en un programa objeto que realiza esencialmente las mismas operaciones de registro que realizaría el intérprete al evaluar el mismo programa fuente.

Esta descripción sugiere una estrategia para implementar un compilador rudimentario: recorremos la expresión de la misma manera que lo hace el intérprete. Cuando encontramos una instrucción de registro que el intérprete ejecutaría al evaluar la expresión, no ejecutamos la instrucción sino que la acumulamos en una secuencia. La secuencia de instrucciones resultante será el código objeto. Observe laventaja de eficiencia de la compilación sobre la interpretación. Cada vez que el intérprete evalúa una expresión (por ejemplo, (f 84 96) ), realiza el trabajo de clasificar la expresión (descubriendo que se trata de una aplicación de procedimiento) y comprobar el final de la lista de operandos (descubriendo que hay dos operandos). Con un compilador, la expresión se analiza solo una vez, cuando se genera la secuencia de instrucciones en el momento de la compilación. El código objeto producido por el compilador contiene solo las instrucciones que evalúan el operador y los dos operandos, ensamblan la lista de argumentos y aplican el procedimiento (en proc ) a los argumentos (en argl ).

Este es el mismo tipo de optimización que implementamos en el evaluador de análisis de la sección  [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) . Pero hay más oportunidades para ganar eficiencia en el código compilado. A medida que se ejecuta el intérprete, sigue un proceso que debe ser aplicable a cualquier expresión en el lenguaje. En contraste, un segmento dado de código compilado está destinado a ejecutar alguna expresión particular. Esto puede hacer una gran diferencia, por ejemplo en el uso de la pila para guardar registros. Cuando el intérprete evalúa una expresión, debe estar preparado para cualquier contingencia. Antes de evaluar una subexpresión, el intérprete guarda todos los registros que se necesitarán más adelante, porque la subexpresión puede requerir una evaluación arbitraria. Un compilador, por otro lado, puede explotar la estructura de la expresión particular que está procesando para generar código que evite operaciones de pila innecesarias.

Como ejemplo, considere la combinación (f 84 96) . Antes de que el intérprete evalúe el operador de la combinación, se prepara para esta evaluación guardando los registros que contienen los operandos y el entorno, cuyos valores se necesitarán más adelante. Luego, el intérprete evalúa el operador para obtener el resultado en val , restaura los registros guardados y, finalmente, mueve el resultado de val a proc . Sin embargo, en la expresión particular con la que estamos tratando, el operador es el símbolo f , cuya evaluación se logra mediante la operación de máquina lookup-variable-value , que no altera ningún registro. El compilador que implementamos en esta sección aprovechará este hecho y generará código que evalúa el operador utilizando la instrucción

(asignar proc (op valor-variable-de-búsqueda) (const f) (reg env))

Este código no solo evita los guardados y restauraciones innecesarios, sino que también asigna el valor de la búsqueda directamente a proc , mientras que el intérprete obtendría el resultado en val y luego lo movería a proc .

Un compilador también puede optimizar el acceso al entorno. Una vez analizado el código, el compilador puede, en muchos casos, saber en qué marco se ubicará una variable en particular y acceder a ese marco directamente, en lugar de realizar la búsqueda de valores de variables . Analizaremos cómo implementar dicho acceso a variables en la sección  [5.5.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.6) . Sin embargo, hasta entonces, nos centraremos en el tipo de optimizaciones de registros y pilas descritas anteriormente. Hay muchas otras optimizaciones que puede realizar un compilador, como codificar operaciones primitivas "en línea" en lugar de utilizar un mecanismo  de aplicación general (consulte el ejercicio [5.38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.38) ); pero no las enfatizaremos aquí. Nuestro objetivo principal en esta sección es ilustrar el proceso de compilación en un contexto simplificado (pero aún así interesante).

**[5.5.1 Estructura del compilador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.5.1)**

En la sección  [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) modificamos nuestro intérprete metacircular original para separar el análisis de la ejecución. Analizamos cada expresión para producir un procedimiento de ejecución que tomara un entorno como argumento y realizara las operaciones requeridas. En nuestro compilador, haremos esencialmente el mismo análisis. Sin embargo, en lugar de producir procedimientos de ejecución, generaremos secuencias de instrucciones que ejecutará nuestra máquina de registros.

El procedimiento compile es el envío de nivel superior en el compilador. Corresponde al procedimiento  eval de la sección [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) , al procedimiento  analyze de la sección [4.1.7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.7) y al punto de entrada eval-dispatch del explicit-control-evaluator de la sección  [5.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.1) . El compilador, al igual que los intérpretes, utiliza elprocedimientos de sintaxis de expresión definidos en la sección  [4.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.2) . [35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_797) Compile realiza un análisis de caso sobre el tipo sintáctico de la expresión que se va a compilar. Para cada tipo de expresión, envía a un especialista*generador de código* :

(define (compila exp enlace objetivo)   
  (cond ((autoevaluación? exp)   
         (compilación-autoevaluación exp enlace objetivo))   
        ((entrecomillado? exp) (compilación-entrecomillado exp enlace objetivo))   
        ((variable? exp)   
         (compilación-variable exp enlace objetivo))   
        ((asignación? exp)   
         (compilación-asignación exp enlace objetivo))   
        ((definición? exp)   
         (compilación-definición exp enlace objetivo))   
        ((si? exp) (compilación-si exp enlace objetivo))   
        ((lambda? exp) (compilación-lambda exp enlace objetivo))   
        ((comienzo? exp)   
         (compilación-secuencia (comienzo-acciones exp) enlace   
                           objetivo   
                           ))   
        ((cond? exp) (compilación (cond->if exp) enlace objetivo))   
        ((aplicación? exp)   
         (compilación-aplicación exp enlace de destino))   
        (else   
         (error "Tipo de expresión desconocido -- COMPILE" exp))))

**[Objetivos y vínculos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_798)**

Compilar y los generadores de código que llama toman dos argumentos además de la expresión para compilar. Hay una*target* , que especifica el registro en el que el código compilado debe devolver el valor de la expresión. También hay un*descriptor de enlace* , que describe cómo debe proceder el código resultante de la compilación de la expresión cuando haya terminado su ejecución. El descriptor de enlace puede exigir que el código realice una de las tres cosas siguientes:

* continuar en la siguiente instrucción en secuencia (esto esespecificado por el descriptor de enlace siguiente ),
* Regresar del procedimiento que se está compilando (esto se especifica)por el descriptor de enlace return ), o
* saltar a un punto de entrada nombrado (esto se especifica utilizando la etiqueta designada como descriptor de enlace).

Por ejemplo, compilar la expresión 5 (que se autoevalúa) con un objetivo del registro val y un enlace de next debería producir la instrucción

(asignar val (const 5))

Compilar la misma expresión con un enlace de retorno debería producir las instrucciones

(asignar val (const 5))   
(ir a (reg continuar))

En el primer caso, la ejecución continuará con la siguiente instrucción de la secuencia. En el segundo caso, regresaremos de una llamada a un procedimiento. En ambos casos, el valor de la expresión se colocará en el registro val de destino.

**[Secuencias de instrucciones y uso de la pila](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_799)**

Cada generador de código devuelve una *secuencia de instrucciones* que contiene el código objeto que ha generado para la expresión. La generación de código para una expresión compuesta se logra combinando la salida de generadores de código más simples para expresiones componentes, de la misma manera que la evaluación de una expresión compuesta se logra evaluando las expresiones componentes.

El método más simple para combinar secuencias de instrucciones es un procedimientollamada append-instruction-sequences . Toma como argumentos cualquier número de secuencias de instrucciones que se ejecutarán secuencialmente; las agrega y devuelve la secuencia combinada. Es decir, si < *s e q 1* > y < *s e q 2* > son secuencias de instrucciones, entonces evaluar

(secuencias-de-instrucciones-anexas < *s e q 1* > < *s e q 2* >)

produce la secuencia

< *s e q 1* >   
< *s e q 2* >

Siempre que sea necesario guardar registros, los generadores de código del compilador utilizanpreserving , que es un método más sutil para combinar secuencias de instrucciones. Preserving toma tres argumentos: un conjunto de registros y dos secuencias de instrucciones que se ejecutarán secuencialmente. Agrega las secuencias de tal manera que el contenido de cada registro en el conjunto se conserva durante la ejecución de la primera secuencia, si esto es necesario para la ejecución de la segunda secuencia. Es decir, si la primera secuencia modifica el registro y la segunda secuencia realmente necesita el contenido original del registro, entonces preserving envuelve un guardado y una restauración del registro alrededor de la primera secuencia antes de agregar las secuencias. De lo contrario, preserving simplemente devuelve las secuencias de instrucciones agregadas. Así, por ejemplo,

(preservando (lista < *r e g 1* > < *r e g 2* >) < *s e q 1* > < *s e q 2* >)

produce una de las siguientes cuatro secuencias de instrucciones, dependiendo de cómo < *s e q 1* > y < *s e q 2* > utilizan < *r e g 1* > y < *r e g 2* >:

Forma

El contenido generado por IA puede ser incorrecto.

Al utilizar la conservación para combinar secuencias de instrucciones, el compilador evita operaciones de pila innecesarias. Esto también aísla los detalles de si se deben generar o no instrucciones de guardado y restauración dentro del procedimiento de conservación , separándolos de las preocupaciones que surgen al escribir cada uno de los generadores de código individuales. De hecho, los generadores de código no producen explícitamente instrucciones de guardado o restauración .

En principio, podríamos representar una secuencia de instrucciones simplemente como una lista de instrucciones. Append-instruction-sequences podría entonces combinar secuencias de instrucciones realizando un append de lista ordinario . Sin embargo, preserving sería entonces una operación compleja, porque tendría que analizar cada secuencia de instrucciones para determinar cómo la secuencia utiliza sus registros. Preserving sería ineficiente así como complejo, porque tendría que analizar cada uno de los argumentos de su secuencia de instrucciones, aunque estas secuencias podrían haber sido construidas por llamadas a preserving , en cuyo caso sus partes ya habrían sido analizadas. Para evitar un análisis tan repetitivo asociaremos con cada secuencia de instrucciones alguna información sobre su uso de registros. Cuando construyamos una secuencia de instrucciones básica proporcionaremos esta información explícitamente, y los procedimientos que combinen secuencias de instrucciones derivarán información de uso de registros para la secuencia combinada a partir de la información asociada con las secuencias componentes.

Una secuencia de instrucciones contendrá tres piezas de información:

* el conjunto de registros que deben inicializarse antes de que se ejecuten las instrucciones de la secuencia (se dice que la secuencia *necesita estos registros),*
* el conjunto de registros cuyos valores son modificados por las instrucciones en la secuencia, y
* las instrucciones reales (también llamadas *declaraciones* ) en la secuencia.

Representaremos una secuencia de instrucciones como una lista de sus tres partes. El constructor de secuencias de instrucciones es, por tanto,

(define (make-instruction-secuence necesita declaraciones modifica)   
  (list necesita declaraciones modifica))

Por ejemplo, la secuencia de dos instrucciones que busca el valor de la variable x en el entorno actual, asigna el resultado a val y luego retorna, requiere que los registros env y continue se hayan inicializado y modifica el registro val . Por lo tanto, esta secuencia se construiría como

(hacer-secuencia-de-instrucción '(env continue) '(val)   
 '((asignar val   
           (op valor-de-variable-de-búsqueda) (const x) (reg env))   
   (goto (reg continue))))

A veces necesitamos construir una secuencia de instrucciones sin declaraciones:

(define (secuencia-de-instrucciones-vacía)   
  (crea-secuencia-de-instrucciones '() '() '()))

Los procedimientos para combinar secuencias de instrucciones se muestran en la sección  [5.5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.4) .

**Ejercicio 5.31.**  Al evaluar una aplicación de procedimiento, el evaluador de control explícito siempre guarda y restaura el registro env alrededor de la evaluación del operador, guarda y restaura env alrededor de la evaluación de cada operando (excepto el último), guarda y restaura argl alrededor de la evaluación de cada operando, y guarda y restaura proc alrededor de la evaluación de la secuencia de operandos. Para cada una de las siguientes combinaciones, diga cuáles de estas operaciones de guardar y restaurar son superfluas y, por lo tanto, podrían eliminarse mediante el mecanismo de conservación del compilador :

(f'x'y)   
  
((f)'x'y)   
  
(f(g'x)y)   
  
(f(g'x)'y)

**Ejercicio 5.32.**  Al utilizar el mecanismo de conservación , el compilador evitará guardar y restaurar el entorno en torno a la evaluación del operador de una combinación en el caso en que el operador sea un símbolo. También podríamos incorporar dichas optimizaciones en el evaluador. De hecho, el evaluador de control explícito de la sección  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4) ya realiza una optimización similar, al tratar las combinaciones sin operandos como un caso especial.

a. Ampliar el evaluador de control explícito para reconocer como una clase separada de expresiones las combinaciones cuyo operador es un símbolo y aprovechar este hecho al evaluar dichas expresiones.

b. Alyssa P. Hacker sugiere que, al ampliar el evaluador para que reconozca cada vez más casos especiales, podríamos incorporar todas las optimizaciones del compilador, lo que eliminaría por completo la ventaja de la compilación. ¿Qué opinas de esta idea?

**[5.5.2 Compilación de expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.5.2)**

En esta sección y la siguiente implementamos los generadores de código a los que se envía el procedimiento de compilación .

**[Compilación de código de enlace](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_802)**

En general, la salida de cada generador de código terminará con instrucciones (generadas por el procedimiento compile-linkage ) que implementan el enlace requerido. Si el enlace es return , entonces debemos generar la instrucción (goto (reg continue)) . Esto necesita el registro continue y no modifica ningún registro. Si el enlace es next , entonces no necesitamos incluir ninguna instrucción adicional. De lo contrario, el enlace es una etiqueta y generamos un goto a esa etiqueta, una instrucción que no necesita ni modifica ningún registro. [36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_803)

(define (compilación-enlace enlace)   
  (cond ((eq? enlace 'return)   
         (hacer-secuencia-de-instrucciones '(continuar) '()   
          '((goto (reg continuar)))))   
        ((eq? enlace 'next)   
         (secuencia-de-instrucciones-vacía))   
        (else   
         (hacer-secuencia-de-instrucciones '() '()   
          `((goto (etiqueta ,enlace)))))))

El código de enlace se agrega a una secuencia de instrucciones preservando el registro de continuación , ya que un enlace de retorno requerirá el registro de continuación : si la secuencia de instrucciones dada modifica la continuación y el código de enlace lo necesita, se guardará y restaurará la continuación .

(define (fin-con-enlace enlace secuencia-de-instrucciones)   
  (preservando '(continuar)   
   secuencia-de-instrucciones   
   (compilar-enlace enlace)))

**[Compilar expresiones simples](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_804)**

Los generadores de código para expresiones, citas y variables autoevaluadas construyen secuencias de instrucciones que asignan el valor requerido al registro de destino y luego proceden según lo especificado por el descriptor de enlace.

(define (compila-autoevaluación exp objetivo enlace)   
  (fin-con-enlace enlace   
   (hacer-secuencia-de-instrucciones '() (lista objetivo)   
    `((asignar ,objetivo (const ,exp))))))  
(define (compilación-entre-citas exp objetivo enlace)   
  (fin-con-enlace enlace   
   (hacer-secuencia-de-instrucciones '() (lista objetivo)   
    `((asignar ,objetivo (const ,(texto-de-cita exp)))))))  
(define (compilar-variable exp objetivo enlace)   
  (fin-con-enlace enlace   
   (hacer-secuencia-de-instrucción '(env) (lista objetivo)   
    `((asignar ,objetivo   
              (op valor-variable-búsqueda)   
              (const ,exp)   
              (reg env))))))

Todas estas instrucciones de asignación modifican el registro de destino, y la que busca una variable necesita el registro env .

Las asignaciones y definiciones se manejan de manera muy similar a como se hacen en el intérprete. Generamos de manera recursiva código que calcula el valor que se asignará a la variable y le agregamos una secuencia de dos instrucciones que realmente establece o define la variable y asigna el valor de toda la expresión (el símbolo ok ) al registro de destino. La compilación recursiva tiene el destino val y el enlace a continuación, de modo que el código colocará su resultado en val y continuará con el código que se agrega después de él. La adición se realiza preservando env , ya que el entorno es necesario para establecer o definir la variable y el código para el valor de la variable podría ser la compilación de una expresión compleja que podría modificar los registros de formas arbitrarias.

(define (compilación-asignación exp objetivo enlace)   
  (let ((var (asignación-variable exp))   
        (obtener-valor-código   
         (compilación (asignación-valor exp) 'val 'next)))   
    (fin-con-enlace enlace   
     (preservando '(env)   
      obtener-valor-código   
      (hacer-secuencia-de-instrucción '(env val) (lista objetivo)   
       `((ejecutar (op ¡establecer-valor-variable!)   
                  (const ,var)   
                  (reg val)   
                  (reg env))   
         (asignar ,objetivo (const ok))))))))  
(define (compila-definición exp objetivo enlace)   
  (let ((var (variable-definición exp))   
        (obtener-valor-código   
         (compila (valor-definición exp) 'val 'next)))   
    (fin-con-enlace enlace   
     (preservando '(env)   
      obtener-valor-código   
      (hacer-secuencia-de-instrucción '(env val) (lista objetivo)   
       `((realizar (op variable-definida!)   
                  (const ,var)   
                  (reg val)   
                  (reg env))   
         (asignar ,objetivo (const ok))))))))

La secuencia de dos instrucciones adjunta requiere env y val y modifica el objetivo. Tenga en cuenta que, aunque conservamos env para esta secuencia, no conservamos val , porque el código de obtención de valor está diseñado para colocar explícitamente su resultado en val para que lo use esta secuencia. (De hecho, si conserváramos val , tendríamos un error, porque esto haría que el contenido anterior de val se restableciera justo después de que se ejecute el código de obtención de valor ).

**[Compilación de expresiones condicionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_805)**

El código para una expresión if compilada con un destino y un enlace determinados tiene la forma

 < *compilación de predicado, objetivo  val , enlace  siguiente* >   
 (prueba (op falso?) (reg val))   
 (rama (etiqueta rama-falsa))   
rama-verdadera   
 < *compilación de consecuente con objetivo dado y enlace dado o  después-si* >   
rama-falsa   
 < *compilación de alternativo con objetivo dado y enlace* >   
después-si

Para generar este código, compilamos el predicado, el consecuente y la alternativa, y combinamos el código resultante con instrucciones para probar el resultado del predicado y con etiquetas recién generadas para marcar las ramas verdaderas y falsas y el final del condicional. [37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_806) En esta disposición del código, debemos ramificar alrededor de la rama verdadera si la prueba es falsa. La única pequeña complicación está en cómo se debe manejar el enlace para la rama verdadera. Si el enlace para el condicional es return o una etiqueta, entonces las ramas verdaderas y falsas usarán este mismo enlace. Si el enlace es next , la rama verdadera termina con un salto alrededor del código para la rama falsa hasta la etiqueta al final del condicional.

(define (compila-si exp enlace-objetivo)   
  (let ((rama-t (etiqueta-hacer 'rama-verdadera))   
        (rama-f (etiqueta-hacer 'rama-falsa))                       
        (después-si (etiqueta-hacer 'después-si)))   
    (let ((enlace-consecuente   
           (si (eq? enlace 'siguiente) enlace-después-si)))   
      (let ((código-p (compila-si-predicado exp) 'val 'siguiente))   
            (código-c   
             (compila   
              -si-consecuente exp) enlace-consecuente-objetivo))   
            (código-a   
             (compila-si-alternativa exp) enlace-objetivo)))   
        (preservando '(env continue)   
         código-p   
         (secuencias-de-instrucciones-agregadas   
          (secuencia-de-instrucciones-hacer '(val) '()   
           `((prueba (op falso?) (reg val))   
             (rama (etiqueta ,f-rama))))   
          (secuencias-de-instrucciones-paralelas   
           (agregar-secuencias-de-instrucciones rama-t código-c)   
           (agregar-secuencias-de-instrucciones rama-f código-a))   
          después-de-si))))))

Env se conserva alrededor del código de predicado porque podría ser necesario para las ramas verdadera y falsa, y continue se conserva porque podría ser necesario para el código de enlace en esas ramas. El código para las ramas verdadera y falsa (que no se ejecutan secuencialmente) se agrega utilizando un combinador especial parallel-instruction-sequences descrito en la sección  [5.5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.4) .

Tenga en cuenta que cond es una expresión derivada, por lo que todo lo que el compilador debe hacer es aplicar el transformador cond->if (de la sección  [4.1.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.2) ) y compilar la expresión if resultante.

**[Compilación de secuencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_807)**

La compilación de secuencias (a partir de cuerpos de procedimientos o expresiones de inicio explícitas ) es paralela a su evaluación. Se compila cada expresión de la secuencia: la última expresión con el enlace especificado para la secuencia y las demás expresiones con el enlace next (para ejecutar el resto de la secuencia). Las secuencias de instrucciones para las expresiones individuales se agregan para formar una única secuencia de instrucciones, de modo que env (necesaria para el resto de la secuencia) y continue (posiblemente necesaria para el enlace al final de la secuencia) se conservan.

(define (compilar-secuencia seq objetivo enlace)   
  (si (última-exp? seq)   
      (compilar (primera-exp seq) objetivo enlace)   
      (preservando '(env continue)   
       (compilar (primera-exp seq) objetivo 'siguiente)   
       (compilar-secuencia (resto-exps seq) objetivo enlace))))

**[Compilación de expresiones lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_808)**

Las expresiones lambda construyen procedimientos. El código objeto de una expresión lambda debe tener la forma

< *construir objeto de procedimiento y asignarlo al registro de destino* >   
< *enlace* >

Cuando compilamos la expresión lambda , también generamos el código para el cuerpo del procedimiento. Aunque el cuerpo no se ejecutará en el momento de la construcción del procedimiento, es conveniente insertarlo en el código objeto justo después del código para la expresión lambda . Si el enlace para la expresión lambda es una etiqueta o return , esto está bien. Pero si el enlace es next , necesitaremos omitir el código para el cuerpo del procedimiento utilizando un enlace que salte a una etiqueta que se inserta después del cuerpo. El código objeto tiene, por lo tanto, la forma

 < *construir objeto de procedimiento y asignarlo al registro de destino* >   
 < *código para el enlace dado* > *o*  (goto (etiqueta after-lambda))  
 < *compilación del cuerpo del procedimiento* >   
after-lambda

Compile-lambda genera el código para construir el objeto de procedimiento seguido del código para el cuerpo del procedimiento. El objeto de procedimiento se construirá en tiempo de ejecución combinando el entorno actual (el entorno en el punto de definición) con el punto de entrada al cuerpo del procedimiento compilado (una etiqueta recién generada). [38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_809)

(define (compila-lambda exp objetivo enlace)   
  (let ((entrada-proc (etiqueta-hacer 'entrada))   
        (después-de-lambda (etiqueta-hacer 'después-de-lambda)))   
    (let ((enlace-lambda   
           (if (eq? enlace 'next) enlace-después-de-lambda)))   
      (anexas-secuencias-de-instrucciones   
       (tack-on-secuencia-de-instrucciones   
        (end-with-linkage enlace-lambda   
         (hacer-secuencia-de-instrucciones '(env) (lista objetivo)   
          `((asignar ,objetivo   
                    (op hacer-procedimiento-compilado)   
                    (etiqueta ,entrada-proc)   
                    (reg env)))))   
        (compila-cuerpo-lambda exp entrada-proc))   
       después-de-lambda))))

Compile-lambda utiliza el combinador especial tack-on-instruction-sequence (sección  [5.5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.4) ) en lugar de append-instruction-sequences para agregar el cuerpo del procedimiento al código de expresión lambda , porque el cuerpo no es parte de la secuencia de instrucciones que se ejecutará cuando se ingresa la secuencia combinada; más bien, está en la secuencia solo porque ese era un lugar conveniente para colocarlo.

Compile-lambda-body construye el código para el cuerpo del procedimiento. Este código comienza con una etiqueta para el punto de entrada. A continuación vienen las instrucciones que harán que el entorno de evaluación en tiempo de ejecución cambie al entorno correcto para evaluar el cuerpo del procedimiento, es decir, el entorno de definición del procedimiento, ampliado para incluir los enlaces de los parámetros formales a los argumentos con los que se llama al procedimiento. Después de esto viene el código para la secuencia de expresiones que compone el cuerpo del procedimiento. La secuencia se compila con el enlace return y el objetivo val de modo que terminará retornando del procedimiento con el resultado del procedimiento en val .

(define (compila-cuerpo-lambda exp entrada-proc)   
  (let ((formales (parámetros-lambda exp)))   
    (agrega-secuencias-de-instrucciones   
     (haz-secuencia-de-instrucciones '(env proc argl) '(env)   
      `(,entrada-proc   
        (asigna env (op entorno-procedimiento-compilado) (reg proc))   
        (asigna env   
                (op entorno-extendido)   
                (const ,formales)   
                (reg argl)   
                (reg env))))   
     (compila-secuencia (cuerpo-lambda exp) 'val 'return))))

**[5.5.3 Compilación de combinaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.5.3)**

La esencia del proceso de compilación es la compilación de aplicaciones de procedimiento. El código para una combinación compilada con un objetivo y un enlace determinados tiene la forma

< *compilación de operador, objetivo  proc , enlace  siguiente* >   
< *evaluar operandos y construir lista de argumentos en  argl* >   
< *compilación de llamada a procedimiento con objetivo y enlace dados* >

Es posible que sea necesario guardar y restaurar los registros env , proc y argl durante la evaluación del operador y los operandos. Tenga en cuenta que este es el único lugar en el compilador donde se especifica un objetivo distinto de val .

El código requerido es generado por compile-application . Esto compila recursivamente el operador, para producir código que pone el procedimiento a ser aplicado en proc , y compila los operandos, para producir código que evalúa los operandos individuales de la aplicación. Las secuencias de instrucciones para los operandos son combinadas (por construct-arglist ) con código que construye la lista de argumentos en argl , y el código de lista de argumentos resultante es combinado con el código de procedimiento y el código que realiza la llamada de procedimiento (producido por compile-procedure-call ). Al agregar las secuencias de código, el registro env debe ser preservado alrededor de la evaluación del operador (ya que evaluar el operador podría modificar env , que será necesario para evaluar los operandos), y el registro proc debe ser preservado alrededor de la construcción de la lista de argumentos (ya que evaluar los operandos podría modificar proc , que será necesario para la aplicación del procedimiento real). Continue también debe ser preservado en todo momento, ya que es necesario para el enlace en la llamada de procedimiento.

(define (compilación-aplicación exp objetivo enlace)   
  (let ((código-proc (compilación (operador exp) 'proc 'next))   
        (códigos-operando   
         (mapa (lambda (operando) (compilación operando 'val 'next))   
              (operandos exp))))   
    (preservando '(env continue)   
     código-proc   
     (preservando '(proc continue)   
      (construcción-lista-argumentos códigos-operando)   
      (compilación-procedimiento-llamada objetivo enlace)))))

El código para construir la lista de argumentos evaluará cada operando en val y luego aplicará la conversión de ese valor a la lista de argumentos que se acumula en argl . Dado que aplicamos la conversión de los argumentos a argl en secuencia, debemos comenzar con el último argumento y terminar con el primero, de modo que los argumentos aparezcan en orden del primero al último en la lista resultante. En lugar de desperdiciar una instrucción inicializando argl en la lista vacía para preparar esta secuencia de evaluaciones, hacemos que la primera secuencia de código construya el argl inicial . La forma general de la construcción de la lista de argumentos es, por lo tanto, la siguiente:

< *compilación del último operando, destinado a  val* >   
(assign argl (op list) (reg val))   
< *compilación del siguiente operando, destinado a  val* >   
(assign argl (op cons) (reg val) (reg argl))   
... < *compilación del primer operando, destinado a  val* >   
(assign argl (op cons) (reg val) (reg argl))

Argl debe conservarse alrededor de cada evaluación de operando excepto la primera (para que los argumentos acumulados hasta el momento no se pierdan), y env debe conservarse alrededor de cada evaluación de operando excepto la última (para su uso en evaluaciones de operando posteriores).

Compilar este código de argumentos es un poco complicado, debido al tratamiento especial del primer operando que se va a evaluar y la necesidad de conservar argl y env en lugares diferentes. El procedimiento construct-arglist toma como argumentos el código que evalúa los operandos individuales. Si no hay operandos en absoluto, simplemente emite la instrucción

(asignar argl(const()))

De lo contrario, construct-arglist crea código que inicializa argl con el último argumento y agrega código que evalúa el resto de los argumentos y los adjunta a argl en sucesión. Para procesar los argumentos desde el último hasta el primero, debemos invertir la lista de secuencias de códigos de operandos del orden proporcionado por compile-application .

(define (construir-lista-argumentos códigos-de-operando)   
  (let ((códigos-de-operando (invertir códigos-de-operando)))   
    (if (null? códigos-de-operando)   
        (hacer-secuencia-de-instrucciones '() '(argl)   
         '((asignar argl (const ()))))   
        (let ((código-para-obtener-el-último-argumento   
               (agregar-secuencias-de-instrucciones   
                (car códigos-de-operando)   
                (hacer-secuencia-de-instrucciones '(val) '(argl)   
                 '((asignar argl (op list) (reg val)))))))   
          (if (null? (cdr códigos-de-operando))   
              código-para-obtener-el-último-argumento   
              (preservando '(env)   
               código-para-obtener-el-último-argumento   
               (código-para-obtener-los-argumentos-del-resto   
                (cdr códigos-de-operando)))))))   
(define (código-para-obtener-argumentos-de-resto códigos-de-operando)   
  (let ((código-para-siguiente-argumento   
         (preservando '(argl)   
          (car códigos-de-operando)   
          (make-secuencia-de-instrucciones '(val argl) '(argl)   
           '((assign argl   
              (op cons) (reg val) (reg argl)))))))   
    (if (null? (cdr códigos-de-operando))   
        código-para-siguiente-argumento   
        (preservando '(env)   
         código-para-siguiente-argumento   
         (código-para-obtener-argumentos-de-resto (cdr códigos-de-operando))))))

**[Procedimientos de aplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_810)**

Después de evaluar los elementos de una combinación, el código compilado debe aplicar el procedimiento en proc a los argumentos en argl . El código realiza esencialmente el mismo envío que el procedimiento apply en el evaluador metacircular de la sección  [4.1.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.1) o el punto de entrada apply-dispatch en el evaluador explicit-control de la sección  [5.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.1) . Verifica si el procedimiento que se va a aplicar es un procedimiento primitivo o un procedimiento compilado. Para un procedimiento primitivo, utiliza apply-primitive-procedure ; veremos en breve cómo maneja los procedimientos compilados. El código procedure-application tiene la siguiente forma:

 (prueba (op procedimiento-primitivo?) (reg proc))   
 (rama (etiqueta rama-primitiva))   
rama-compilada   
 < *código para aplicar el procedimiento compilado con el objetivo dado y el enlace apropiado* >   
rama-primitiva   
 (asignar < *objetivo* >   
         (op aplicar-procedimiento-primitivo)   
         (reg proc)   
         (reg argl))   
 < *enlace* >   
después de la llamada

Observe que la rama compilada debe saltarse la rama primitiva. Por lo tanto, si el enlace para la llamada al procedimiento original era next , la rama compuesta debe utilizar un enlace que salte a una etiqueta que se inserta después de la rama primitiva. (Esto es similar al enlace utilizado para la rama verdadera en compile-if ).

(define (compilar-procedimiento-llamada objetivo enlace)   
  (let ((rama-primitiva (make-label 'rama-primitiva))   
        (rama-compilada (make-label 'rama-compilada))   
        (llamada-posterior (make-label 'llamada-posterior)))   
    (let ((enlace-compilado   
           (if (eq? enlace 'next) llamada-posterior enlace)))   
      (añadir-secuencias-de-instrucciones   
       (make-secuencia-de-instrucciones '(proc) '()   
        `((prueba (op procedimiento-primitivo?) (reg proc))   
          (rama (etiqueta ,rama-primitiva))))   
       (secuencias-de-instrucciones-paralelas   
        (añadir-secuencias-de-instrucciones   
         rama-compilada   
         (aplicación-proc-compilación objetivo enlace-compilado))   
        (añadir-secuencias-de-instrucciones   
         rama-primitiva   
         (fin-con-enlace enlace   
          (make-instruction-secuencia '(proc argl)   
                                     (lista objetivo)   
           `((assign ,target   
                     (op apply-primitive-procedure)   
                     (reg proc)   
                     (reg argl)))))))   
       después de la llamada))))

Las ramas primitivas y compuestas, al igual que las ramas verdaderas y falsas en compile-if , se agregan utilizando parallel-instruction-sequences en lugar de las append-instruction-sequences ordinarias , porque no se ejecutarán secuencialmente.

**[Aplicación de procedimientos compilados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_811)**

El código que maneja la aplicación de procedimientos es la parte más sutil del compilador, aunque las secuencias de instrucciones que genera son muy cortas. Un procedimiento compilado (como el construido por compile-lambda ) tiene un punto de entrada, que es una etiqueta que designa dónde comienza el código para el procedimiento. El código en este punto de entrada calcula un resultado en val y regresa ejecutando la instrucción (goto (reg continue)) . Por lo tanto, podríamos esperar que el código para una aplicación de procedimiento compilado (que se generará mediante compile-proc-appl ) con un objetivo y un enlace dados se vea así si el enlace es una etiqueta

 (asignar continuar (etiqueta proc-retorno))   
 (asignar val (op entrada-procedimiento-compilado) (reg proc))   
 (goto (reg val))   
proc-retorno   
 (asignar < *objetivo* > (reg val))    *; incluido si el objetivo no es  val*  
 (goto (etiqueta < *enlace* >))    *; código de enlace*

o así si el enlace es de retorno .

 (guardar continuar)   
 (asignar continuar (etiqueta proc-return))   
 (asignar val (op entrada-procedimiento-compilado) (reg proc))   
 (goto (reg val))   
proc-return   
 (asignar < *target* > (reg val))    *; incluido si target no es  val*  
 (restaurar continuar)   
 (goto (reg continue))    *; código de enlace*

Este código configura continue para que el procedimiento regrese a una etiqueta proc-return y salte al punto de entrada del procedimiento. El código en proc-return transfiere el resultado del procedimiento desde val al registro de destino (si es necesario) y luego salta a la ubicación especificada por el enlace. (El enlace siempre es return o una etiqueta, porque compile-procedure-call reemplaza un enlace siguiente para la rama de procedimiento compuesto por una etiqueta posterior a la llamada ).

De hecho, si el objetivo no es val , ese es exactamente el código que nuestro compilador generará. [39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_812) Sin embargo, normalmente el objetivo es val (la única vez que el compilador especifica un registro diferente es cuando apunta la evaluación de un operador a proc ), por lo que el resultado del procedimiento se coloca directamente en el registro de destino y no hay necesidad de regresar a una ubicación especial que lo copie. En cambio, simplificamos el código configurando continue para que el procedimiento ``regrese'' directamente al lugar especificado por el enlace del llamador:

< *configurar  continuar  para vinculación* >   
(asignar val (op entrada-procedimiento-compilado) (reg proc))   
(goto (reg val))

Si el enlace es una etiqueta, configuramos continue para que el procedimiento regrese a esa etiqueta. (Es decir, el (goto (reg continue)) con el que finaliza el procedimiento se vuelve equivalente al (goto (label < *linkage* >)) en proc-return anterior).

(asignar continuar (etiqueta < *enlace* >))   
(asignar val (op entrada-procedimiento-compilado) (reg proc))   
(goto (reg val))

Si el enlace es return , no necesitamos configurar continue en absoluto: ya contiene la ubicación deseada. (Es decir, el (goto (reg continue)) con el que finaliza el procedimiento va directamente al lugar donde habría ido el (goto (reg continue)) en proc-return ).

(asignar val (op entrada-procedimiento-compilado) (reg proc))   
(goto (reg val))

Con esta implementación del enlace de retorno , el compilador genera código recursivo de cola. Al llamar a un procedimiento como paso final en el cuerpo de un procedimiento, se realiza una transferencia directa, sin guardar ninguna información en la pila.

Supongamos, en cambio, que hubiéramos manejado el caso de una llamada a un procedimiento con un enlace de retorno y un objetivo de val como se muestra arriba para un objetivo que no es val . Esto destruiría la recursión de cola. Nuestro sistema seguiría dando el mismo valor para cualquier expresión. Pero cada vez que llamáramos a un procedimiento, guardaríamos continue y return después de la llamada para deshacer el guardado (inútil). Estos guardados adicionales se acumularían durante un anidamiento de llamadas a procedimientos. [40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_813)

Compile-proc-appl genera el código de aplicación de procedimiento anterior considerando cuatro casos, dependiendo de si el objetivo de la llamada es val y si el enlace es return . Observe que las secuencias de instrucciones se declaran para modificar todos los registros, ya que la ejecución del cuerpo del procedimiento puede cambiar los registros de formas arbitrarias. [41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_814) Observe también que la secuencia de código para el caso con el objetivo val y el enlace return se declara para necesitar continue : Aunque continue no se utiliza explícitamente en la secuencia de dos instrucciones, debemos estar seguros de que continue tendrá el valor correcto cuando ingresemos al procedimiento compilado.

(define (compilar-proc-aplicación-objetivo enlace)   
  (cond ((y (eq? objetivo 'val) (no (eq? enlace 'retorno)))   
         (hacer-secuencia-de-instrucciones '(proc) todos-regs   
           `((asignar continuar (etiqueta ,enlace))   
             (asignar val (op entrada-procedimiento-compilado)   
                         (reg proc))   
             (goto (reg val))))   
        ((y (no (eq? objetivo 'val))   
              (no (eq? enlace 'retorno)))   
         (dejar ((proc-retorno (hacer-etiqueta 'proc-retorno)))   
           (hacer-secuencia-de-instrucciones '(proc) todos-regs   
            `((asignar continuar (etiqueta ,proc-retorno))   
              (asignar val (op entrada-procedimiento-compilado)   
                          (reg proc))   
              (goto (reg val))   
              ,proc-retorno   
              (asignar ,target (reg val))   
              (goto (etiqueta ,enlace))))))   
        ((y (eq? target 'val) (eq? enlace 'retorno))   
         (make-instrucción-secuencia '(proc continue) all-regs   
          '((assign val (op compilado-proc-entrada)   
                        (reg proc))   
            (goto (reg val)))))   
        ((y (no (eq? target 'val)) (eq? enlace 'retorno))   
         (error "devolver enlace, target no val -- COMPILE"   
                target))))

**[5.5.4 Combinación de secuencias de instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.5.4)**

Esta sección describe los detalles sobre cómo se representan y combinan las secuencias de instrucciones. Recuerde de la sección  [5.5.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.5.1) que una secuencia de instrucciones se representa como una lista de los registros necesarios, los registros modificados y las instrucciones reales. También consideraremos que una etiqueta (símbolo) es un caso degenerado de una secuencia de instrucciones, que no necesita ni modifica ningún registro. Por lo tanto, para determinar los registros necesarios y modificados por las secuencias de instrucciones, utilizamos los selectores

(define (registros necesarios)   
  (si (símbolo? s) '() (auto s)))  
(define (registros modificados s)   
  (si (símbolo? s) '() (cadr s)))  
(define (declaraciones s)   
  (si (símbolo? s) (lista s) (caddr s)))

y para determinar si una secuencia dada necesita o modifica un registro dado usamos los predicados

(define (necesita-registro? seq reg)   
  (memq reg (registros-necesarios seq)))  
(define (modifica-registro? seq reg)   
  (memq reg (registros-modificados seq)))

En términos de estos predicados y selectores, podemos implementar los diversos combinadores de secuencias de instrucciones utilizados en todo el compilador.

El combinador básico es append-instruction-sequences . Este toma como argumentos un número arbitrario de secuencias de instrucciones que se deben ejecutar secuencialmente y devuelve una secuencia de instrucciones cuyas instrucciones son las instrucciones de todas las secuencias agregadas juntas. El punto sutil es determinar los registros que se necesitan y modifican por la secuencia resultante. Modifica aquellos registros que son modificados por cualquiera de las secuencias; necesita aquellos registros que deben inicializarse antes de que se pueda ejecutar la primera secuencia (los registros necesarios para la primera secuencia), junto con aquellos registros necesarios para cualquiera de las otras secuencias que no son inicializadas (modificadas) por secuencias que la preceden.

Las secuencias se añaden de dos en dos mediante append-2-sequences . Esto toma dos secuencias de instrucciones seq1 y seq2 y devuelve la secuencia de instrucciones cuyas instrucciones son las instrucciones de seq1 seguidas de las instrucciones de seq2 , cuyos registros modificados son aquellos registros que son modificados por seq1 o seq2 , y cuyos registros necesarios son los registros que necesita seq1 junto con aquellos registros que necesita seq2 que no son modificados por seq1 . (En términos de operaciones de conjunto, el nuevo conjunto de registros necesarios es la unión del conjunto de registros que necesita seq1 con la diferencia de conjunto de los registros que necesita seq2 y los registros modificados por seq1 .) Por lo tanto, append-instruction-sequences se implementa de la siguiente manera:

(define (append-instruction-secuences . seqs)   
  (define (append-2-secuences seq1 seq2)   
    (make-instruction-secuence   
     (list-union (registers-needed seq1)   
                 (list-difference (registers-needed seq2)   
                                  (registers-modified seq1)))   
     (list-union (registers-modified seq1)   
                 (registers-modified seq2))   
     (append (sentencias seq1) (sentencias seq2))))   
  (define (append-seq-list seqs)   
    (if (null? seqs)   
        (secuencia-instrucciones-vacias)   
        (append-2-secuencias (car seqs)   
                            (append-seq-list (cdr seqs))))   
  (append-seq-list seqs))

Este procedimiento utiliza algunas operaciones simples para manipular conjuntos representados como listas, similares a la representación de conjuntos (no ordenados) descrita en la sección  [2.3.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_2.3.3) :

(define (unión-lista s1 s2)   
  (cond ((null? s1) s2)   
        ((memq (car s1) s2) (unión-lista (cdr s1) s2))   
        (else (cons (car s1) (unión-lista (cdr s1) s2)))))  
(define (diferencia-de-lista s1 s2)   
  (cond ((null? s1) '())   
        ((memq (car s1) s2) (diferencia-de-lista (cdr s1) s2))   
        (else (cons (car s1)   
                    (diferencia-de-lista (cdr s1) s2)))))

Preserving , el segundo combinador de secuencias de instrucciones principal, toma una lista de registros regs y dos secuencias de instrucciones seq1 y seq2 que se deben ejecutar secuencialmente. Devuelve una secuencia de instrucciones cuyas instrucciones son las instrucciones de seq1 seguidas de las instrucciones de seq2 , con instrucciones de guardado y restauración apropiadas alrededor de seq1 para proteger los registros en regs que son modificados por seq1 pero necesarios para seq2 . Para lograr esto, preserving primero crea una secuencia que tiene las instrucciones de guardado requeridas seguidas de las instrucciones de seq1 seguidas de las instrucciones de restauración requeridas . Esta secuencia necesita que los registros se guarden y restauren además de los registros necesarios para seq1 , y modifica los registros modificados por seq1 excepto los que se guardan y restauran. Esta secuencia aumentada y seq2 se agregan luego de la manera habitual. El siguiente procedimiento implementa esta estrategia de forma recursiva, recorriendo la lista de registros que se deben preservar: [42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_815)

(define (preservando regs seq1 seq2)   
  (if (null? regs)   
      (append-instruction-secuences seq1 seq2)   
      (let ((first-reg (car regs)))   
        (if (and (necesita-registro? seq2 first-reg)   
                 (modifica-registro? seq1 first-reg))   
            (preservando (cdr regs)   
             (make-instruction-secuence   
              (list-union (list first-reg)   
                          (registros-necesarios seq1))   
              (list-difference (registros-modificados seq1)   
                               (list first-reg))   
              (append `((save ,first-reg))   
                      (sentencias seq1)   
                      `((restore ,first-reg))))   
             seq2)   
            (preservando (cdr regs) seq1 seq2)))))

Otro combinador de secuencias, tack-on-instruction-sequence , es utilizado por compile-lambda para agregar un cuerpo de procedimiento a otra secuencia. Debido a que el cuerpo del procedimiento no está ``en línea'' para ser ejecutado como parte de la secuencia combinada, su uso de registros no tiene impacto en el uso de registros de la secuencia en la que está incrustado. Por lo tanto, ignoramos los conjuntos de registros necesarios y modificados del cuerpo del procedimiento cuando lo agregamos a la otra secuencia.

(define (tack-on-instrucción-secuencia cuerpo-secuencia)   
  (make-instrucción-secuencia   
   (registros-necesarios-secuencia)   
   (registros-modificados-secuencia)   
   (append (declaraciones-secuencia) (declaraciones-secuencia cuerpo))))

Compile-if y compile-procedure-call utilizan un combinador especial llamado parallel-instruction-sequences para agregar las dos ramas alternativas que siguen a una prueba. Las dos ramas nunca se ejecutarán secuencialmente; para cualquier evaluación particular de la prueba, se ingresará una rama u otra. Debido a esto, los registros que necesita la segunda rama todavía son necesarios para la secuencia combinada, incluso si estos son modificados por la primera rama.

(define (secuencias-de-instrucciones-paralelas seq1 seq2)   
  (make-secuencia-de-instrucciones   
   (unión-de-lista (registros-necesarios seq1)   
               (registros-necesarios seq2))   
   (unión-de-lista (registros-modificados seq1)   
               (registros-modificados seq2))   
   (append (declaraciones seq1) (declaraciones seq2))))

**[5.5.5 Un ejemplo de código compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.5.5)**

Ahora que hemos visto todos los elementos del compilador, examinemos un ejemplo de código compilado para ver cómo encajan las cosas. Compilaremos la definición de un procedimiento factorial recursivo llamando a compile :

(compilar   
 '(define (factorial n)   
    (si (= n 1)   
        1   
        (\* (factorial (- n 1)) n)))   
 'val   
 'siguiente)

Hemos especificado que el valor de la expresión define debe colocarse en el registro val . No nos importa lo que haga el código compilado después de ejecutar define , por lo que nuestra elección de next como descriptor de enlace es arbitraria.

Compile determines that the expression is a definition, so it calls compile-definition to compile code to compute the value to be assigned (targeted to val), followed by code to install the definition, followed by code to put the value of the define (which is the symbol ok) into the target register, followed finally by the linkage code. Env is preserved around the computation of the value, because it is needed in order to install the definition. Because the linkage is next, there is no linkage code in this case. The skeleton of the compiled code is thus

  <*save env if modified by code to compute value*>  
  <*compilation of definition value, target val, linkage next*>  
  <*restore env if saved above*>  
  (perform (op define-variable!)  
           (const factorial)  
           (reg val)  
           (reg env))  
  (assign val (const ok))

The expression that is to be compiled to produce the value for the variable factorial is a lambda expression whose value is the procedure that computes factorials. Compile handles this by calling compile-lambda, which compiles the procedure body, labels it as a new entry point, and generates the instruction that will combine the procedure body at the new entry point with the run-time environment and assign the result to val. The sequence then skips around the compiled procedure code, which is inserted at this point. The procedure code itself begins by extending the procedure's definition environment by a frame that binds the formal parameter n to the procedure argument. Then comes the actual procedure body. Since this code for the value of the variable doesn't modify the env register, the optional save and restore shown above aren't generated. (The procedure code at entry2 isn't executed at this point, so its use of env is irrelevant.) Therefore, the skeleton for the compiled code becomes

  (assign val (op make-compiled-procedure)   
              (label entry2)   
              (reg env))   
  (goto (label after-lambda1))   
entry2   
  (assign env (op compiled-procedure-env) (reg proc))   
  (assign env (op extend-environment)   
              (const (n))   
              (reg argl)   
              (reg env))   
  < *compilación del cuerpo del procedimiento* >   
after-lambda1   
  (perform (op define-variable!)   
           (const factorial)   
           (reg val)   
           (reg env))   
  (assign val (const ok))

El cuerpo de un procedimiento siempre se compila (mediante compile-lambda-body ) como una secuencia con un valor de destino y un enlace return . La secuencia en este caso consta de una única expresión if :

(si (= n 1)   
    1   
    (\* (factorial (- n 1)) n))

Compile-if genera código que primero calcula el predicado (apunta a val ), luego verifica el resultado y se ramifica alrededor de la rama verdadera si el predicado es falso. Env y continue se conservan alrededor del código del predicado, ya que pueden ser necesarios para el resto de la expresión if . Dado que la expresión if es la expresión final (y la única expresión) en la secuencia que compone el cuerpo del procedimiento, su objetivo es val y su enlace es return , por lo que las ramas verdadera y falsa se compilan con el objetivo val y el enlace return . (Es decir, el valor del condicional, que es el valor calculado por cualquiera de sus ramas, es el valor del procedimiento).

  < *guardar  continuar ,  env  si fue modificado por predicado y es necesario para las ramas* >   
  < *compilación de predicado,  valor objetivo , enlace  siguiente* >   
  < *restaurar  continuar ,  env  si fue guardado arriba* >   
  (test (op false?) (reg val))   
  (branch (label false-branch4))   
true-branch5   
  < *compilación de rama verdadera,  valor objetivo , enlace  retorno* >   
false-branch4   
  < *compilación de rama falsa,  valor objetivo , enlace  retorno* >   
after-if3

El predicado (= n 1) es una llamada a un procedimiento. Busca el operador (el símbolo = ) y coloca este valor en proc . Luego ensambla los argumentos 1 y el valor de n en argl . Luego prueba si proc contiene un procedimiento primitivo o compuesto y envía a una rama primitiva o a una rama compuesta según corresponda. Ambas ramas se reanudan en la etiqueta posterior a la llamada . Los requisitos para preservar los registros en torno a la evaluación del operador y los operandos no dan como resultado ningún ahorro de registros, porque en este caso esas evaluaciones no modifican los registros en cuestión.

  (asignar proc   
          (op valor-variable-buscada) (const =) (reg env))   
  (asignar val (const 1))   
  (asignar argl (op lista) (reg val))   
  (asignar val (op valor-variable-buscada) (const n) (reg env))   
  (asignar argl (op cons) (reg val) (reg argl))   
  (prueba (op procedimiento-primitivo?) (reg proc))   
  (rama (etiqueta rama-primitiva17))   
rama-compilada16   
  (asignar continuar (etiqueta después-de-llamada15))   
  (asignar val (op entrada-procedimiento-compilado) (reg proc))   
  (goto (reg val))   
rama-primitiva17   
  (asignar val (op aplicar-procedimiento-primitivo)   
              (reg proc)   
              (reg argl))   
después-de-llamada15

La rama verdadera, que es la constante 1, se compila (con el objetivo val y el enlace de retorno ) a

  (asignar val (const 1))   
  (ir a (reg continuar))

El código para la rama falsa es otra llamada a un procedimiento, donde el procedimiento es el valor del símbolo \* y los argumentos son n y el resultado de otra llamada a un procedimiento (una llamada a factorial ). Cada una de estas llamadas configura proc y argl y sus propias ramas primitivas y compuestas. La Figura  [5.17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.17) muestra la compilación completa de la definición del procedimiento factorial . Observe que el posible guardado y restauración de continue y env alrededor del predicado, que se muestra arriba, de hecho se generan, porque estos registros son modificados por la llamada al procedimiento en el predicado y son necesarios para la llamada al procedimiento y el enlace de retorno en las ramas.

**Ejercicio 5.33.**   Consideremos la siguiente definición de procedimiento factorial, que es ligeramente diferente de la dada anteriormente:

(define (factorial-alt n)   
  (si (= n 1)   
      1   
      (\* n (factorial-alt (- n 1)))))

Compila este procedimiento y compara el código resultante con el generado para factorial . Explica las diferencias que encuentres. ¿Alguno de los programas se ejecuta con mayor eficiencia que el otro?

**Ejercicio 5.34.**  Compilar el procedimiento factorial iterativo

(define (factorial n)   
  (define (iter producto contador)   
    (si (> contador n)   
        producto   
        (iter (\* contador producto)   
              (+ contador 1))))   
  (iter 1 1))

Anote el código resultante, mostrando la diferencia esencial entre el código para las versiones iterativas y recursivas de factorial que hace que un proceso acumule espacio de pila y el otro se ejecute en un espacio de pila constante.

|  |
| --- |
| *;; construye el procedimiento y omite el código para el cuerpo del procedimiento*   (assign val            (op make-compiled-procedure) (label entry2) (reg env))    (goto (label after-lambda1))   entry2      *; las llamadas a  factorial  entrarán aquí*   (assign env (op compiled-procedure-env) (reg proc))    (assign env            (op extend-environment) (const (n)) (reg argl) (reg env))  *;; comienza el cuerpo del procedimiento actual*   (save continue)    (save env)   *;; calcular  (= n 1)*   (asignar proc (op valor-variable-buscada) (const =) (reg env))    (asignar val (const 1))    (asignar argl (op lista) (reg val))    (asignar val (op valor-variable-buscada) (const n) (reg env))    (asignar argl (op cons) (reg val) (reg argl))    (prueba (op procedimiento-primitivo?) (reg proc))    (rama (etiqueta rama-primitiva17))  rama-compilada16    (asignar continuar (etiqueta después-llamada15))    (asignar val (op entrada-procedimiento-compilado) (reg proc))    (goto (reg val))  rama-primitiva17    (asignar val (op aplicar-procedimiento-primitivo) (reg proc) (reg argl))   después-llamada15    *;  val  ahora contiene el resultado de  (= n 1)*   (restaurar env)    (restaurar continue)    (test (op false?) (reg val))    (branch (label false-branch4))  true-branch5   *; return 1*   (assign val (const 1))    (goto (reg continue))   false-branch4  *;; calcula y devuelve  (\* (factorial (- n 1)) n)*   (assign proc (op lookup-variable-value) (const \*) (reg env))    (save continue)    (save proc)    *; save  \**  procedure    (assign val (op lookup-variable-value) (const n) (reg env))    (assign argl (op list) (reg val))    (save argl)    *; save parcial lista de argumentos para  \**  *;; calcular  (factorial (- n 1)) , que es el otro argumento para  \**   (asignar proc            (op valor-variable-buscado) (const factorial) (reg env))    (guardar proc)   *; guardar   procedimiento factorial* |
|  |
| **Figura 5.17:**   Compilación de la definición del procedimiento factorial (continúa en la página siguiente). |

|  |
| --- |
| *;; computar  (- n 1) , que es el argumento para  factorial*   (asignar proc (op valor-variable-buscada) (const -) (reg env))    (asignar val (const 1))    (asignar argl (op lista) (reg val))    (asignar val (op valor-variable-buscada) (const n) (reg env))    (asignar argl (op cons) (reg val) (reg argl))    (prueba (op procedimiento-primitivo?) (reg proc))    (rama (etiqueta rama-primitiva8))  rama-compilada7    (asignar continuar (etiqueta llamada-posterior6))    (asignar val (op entrada-procedimiento-compilado) (reg proc))    (goto (reg val))  rama-primitiva8    (asignar val (op aplicar-procedimiento-primitivo) (reg proc) (reg argl))   after-call6    *;  val  ahora contiene el resultado de  (- n 1)*   (assign argl (op list) (reg val))    (restore proc)  *; restaurar  factorial* *;; aplicar  factorial*   (test (op primitive-procedure?) (reg proc))    (branch (label primitive-branch11))  compiled-branch10    (assign continue (label after-call9))    (assign val (op compiled-procedure-entry) (reg proc))    (goto (reg val))  primitive-branch11    (assign val (op apply-primitive-procedure) (reg proc) (reg argl))   after-call9       *;  val  ahora contiene el resultado de  (factorial (- n 1))*   (restore argl)  *; restaurar lista parcial de argumentos para  \**   (assign argl (op cons) (reg val) (reg argl))    (restore proc)  *; restaurar  \**   (restaurar continuar)  *;; aplicar  \**  y devolver su valor    (test (op primitive-procedure?) (reg proc))    (branch (label primitive-branch14))  compiled-branch13  *;; tenga en cuenta que un procedimiento compuesto aquí se llama recursivamente de cola*   (assign val (op compiled-procedure-entry) (reg proc))    (goto (reg val))  primitive-branch14    (assign val (op apply-primitive-procedure) (reg proc) (reg argl))    (goto (reg continue))  after-call12  after-if3  after-lambda1  *;; asignar el procedimiento a la variable  factorial*   (ejecutar     (op define-variable!) (const factorial) (reg val) (reg env))    (asignar val (const ok)) |
|  |
| **Figura 5.17:**   (continuación) |

**Ejercicio 5.35.**   ¿Qué expresión se compiló para producir el código que se muestra en la figura  [5.18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.18) ?

|  |
| --- |
| (assign val (op make-compiled-procedure) (label entry16)                                             (reg env))    (goto (label after-lambda15))  entry16    (assign env (op compiled-procedure-env) (reg proc))    (assign env            (op extend-environment) (const (x)) (reg argl) (reg env))    (assign proc (op lookup-variable-value) (const +) (reg env))    (save continue)    (save proc)    (save env)    (assign proc (op lookup-variable-value) (const g) (reg env))    (save proc)    (assign proc (op lookup-variable-value) (const +) (reg env))    (assign val (const 2))    (assign argl (op list) (reg val))    (assign val (op valor-variable-buscada) (const x) (reg env))    (assign argl (op cons) (reg val) (reg argl))    (test (op procedimiento-primitivo?) (reg proc))    (branch (label rama-primitiva19))  rama-compilada18    (assign continue (label after-call17))    (assign val (op entrada-procedimiento-compilado) (reg proc))    (goto (reg val))  rama-primitiva19    (assign val (op aplicar-procedimiento-primitivo) (reg proc) (reg argl))  after-call17    (assign argl (op lista) (reg val))    (restore proc)    (test (op procedimiento-primitivo?) (reg proc))    (branch (label rama-primitiva22))  rama-compilada21    (assign continue (label después de la llamada20))    (asignar valor (op entrada de procedimiento compilado) (reg proc))    (ir a (reg val))  rama primitiva22    (asignar valor (op aplicar procedimiento primitivo) (reg proc) (reg argl)) |
|  |
| **Figura 5.18:**   Ejemplo de salida del compilador (continúa en la página siguiente). Véase el ejercicio  [5.35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.35) . |

|  |
| --- |
| después de la llamada20    (asignar argl (op lista) (reg val))    (restaurar entorno)    (asignar val (op valor-variable-búsqueda) (const x) (reg env))    (asignar argl (op cons) (reg val) (reg argl))    (restaurar proc)    (restaurar continuar)    (prueba (op procedimiento-primitivo?) (reg proc))    (rama (etiqueta rama-primitiva25))  rama-compilada24    (asignar val (op entrada-procedimiento-compilado) (reg proc))    (goto (reg val))  rama-primitiva25    (asignar val (op aplicar-procedimiento-primitivo) (reg proc) (reg argl))    (goto (reg continuar))  después de la llamada23  después de lambda15    (ejecutar (op ¡definir-variable!) (const f) (reg val) (reg env))    (asignar val (const ok)) |
|  |
| **Figura 5.18:**   (continuación) |

**Ejercicio 5.36.**  ¿Qué orden de evaluación produce nuestro compilador para los operandos de una combinación? ¿De izquierda a derecha, de derecha a izquierda o en algún otro orden? ¿En qué parte del compilador se determina este orden? Modifique el compilador para que produzca algún otro orden de evaluación. (Vea la discusión del orden de evaluación para el evaluador de control explícito en la sección  [5.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.1) ). ¿Cómo afecta el cambio del orden de evaluación de los operandos a la eficiencia del código que construye la lista de argumentos?

**Ejercicio 5.37.**  Una forma de entender el mecanismo de conservación del compilador para optimizar el uso de la pila es ver qué operaciones adicionales se generarían si no usáramos esta idea. Modifique la conservación para que siempre genere las operaciones de guardar y restaurar . Compila algunas expresiones simples e identifica las operaciones de pila innecesarias que se generan. Compara el código con el generado con el mecanismo de conservación intacto.

**Ejercicio 5.38.**  Nuestro compilador es inteligente a la hora de evitar operaciones de pila innecesarias, pero no es inteligente en absoluto cuando se trata de compilar llamadas a los procedimientos primitivos del lenguaje en términos de las operaciones primitivas suministradas por la máquina. Por ejemplo, considere cuánto código se compila para calcular (+ a 1) : el código establece una lista de argumentos en argl , coloca el procedimiento de adición primitivo (que encuentra buscando el símbolo + en el entorno) en proc , y prueba si el procedimiento es primitivo o compuesto. El compilador siempre genera código para realizar la prueba, así como código para las ramas primitivas y compuestas (solo se ejecutará una de las cuales). No hemos mostrado la parte del controlador que implementa primitivos, pero suponemos que estas instrucciones hacen uso de operaciones aritméticas primitivas en las rutas de datos de la máquina. Considere cuánto menos código se generaría si el compilador pudiera *abrir el código de* los primitivos, es decir, si pudiera generar código para usar directamente estas operaciones primitivas de la máquina. La expresión (+ a 1) podría compilarse en algo tan simple como  [43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_822)

(asignar val (op valor-variable-de-busqueda) (const a) (reg env))   
(asignar val (op +) (reg val) (const 1))

En este ejercicio ampliaremos nuestro compilador para que admita la codificación abierta de primitivas seleccionadas. Se generará un código de propósito especial para las llamadas a estos procedimientos primitivos en lugar del código de aplicación de procedimiento general. Para admitir esto, aumentaremos nuestra máquina con registros de argumentos especiales arg1 y arg2 . Las operaciones aritméticas primitivas de la máquina tomarán sus entradas de arg1 y arg2 . Los resultados se pueden colocar en val , arg1 o arg2 .

El compilador debe poder reconocer la aplicación de una primitiva de código abierto en el programa fuente. Aumentaremos el envío en el procedimiento de compilación para reconocer los nombres de estas primitivas además de laspalabras reservadas (las formas especiales) que reconoce actualmente. [44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_823) Para cada forma especial, nuestro compilador tiene un generador de código. En este ejercicio, construiremos una familia de generadores de código para las primitivas de código abierto.

a. A diferencia de las formas especiales, las primitivas de código abierto necesitan que se evalúen todos sus operandos. Escriba un generador de código spread-arguments para que lo utilicen todos los generadores de código de código abierto. Spread-arguments debe tomar una lista de operandos y compilar los operandos dados destinados a registros de argumentos sucesivos. Tenga en cuenta que un operando puede contener una llamada a una primitiva de código abierto, por lo que los registros de argumentos deberán conservarse durante la evaluación del operando.

b. Para cada uno de los procedimientos primitivos = , \* , - y + , escriba un generador de código que tome una combinación con ese operador, junto con un descriptor de destino y de enlace, y produzca código para distribuir los argumentos en los registros y luego realizar la operación dirigida al destino dado con el enlace dado. Solo necesita manejar expresiones con dos operandos. Haga que la compilación se envíe a estos generadores de código.

c. Pruebe el nuevo compilador en el ejemplo factorial . Compare el código resultante con el resultado obtenido sin codificación abierta.

d. Amplíe sus generadores de código para + y \* de modo que puedan manejar expresiones con una cantidad arbitraria de operandos. Una expresión con más de dos operandos deberá compilarse en una secuencia de operaciones, cada una con solo dos entradas.

**[5.5.6 Direccionamiento léxico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.5.6)**

Una de las optimizaciones más comunes que realizan los compiladores es la optimización de la búsqueda de variables. Nuestro compilador, tal como lo hemos implementado hasta ahora, genera código que utiliza la operación de búsqueda de valor de variable de la máquina evaluadora. Esto busca una variable comparándola con cada variable que esté vinculada actualmente, trabajando cuadro por cuadro hacia afuera a través del entorno de tiempo de ejecución. Esta búsqueda puede ser costosa si los cuadros están profundamente anidados o si hay muchas variables. Por ejemplo, considere el problema de buscar el valor de x mientras se evalúa la expresión (\*xyz) en una aplicación del procedimiento que es devuelto por

(sea ((x 3) (y 4))   
  (lambda (a b c d e)   
    (sea ((y (\* a b x))   
          (z (+ c d x)))   
      (\* x y z))))

Dado que una expresión let es simplemente azúcar sintáctica para una combinación lambda , esta expresión es equivalente a

((lambda(x y)   
   (lambda(a b c d e)   
     ((lambda(y z) (\* x y z))   
      (\* a b x)   
      (+ c d x))))   
 3   
 4)

Cada vez que lookup-variable-value busca x , debe determinar que el símbolo x no es igual a y o z (en el primer marco), ni a a , b , c , d o e (en el segundo marco). Supondremos, por el momento, que nuestros programas no usan define -- que las variables están vinculadas solo con lambda . Debido a que nuestro lenguaje esde alcance léxico, el entorno de ejecución para cualquier expresión tendrá una estructura que es paralela a la estructura léxica del programa en el que aparece la expresión. [45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_824) Por lo tanto, el compilador puede saber, cuando analiza la expresión anterior, que cada vez que se aplica el procedimiento, la variable x en (\*xyz) se encontrará dos cuadros fuera del cuadro actual y será la primera variable en ese cuadro.

Podemos explotar este hecho inventando un nuevo tipo de operación de búsqueda de variables, lexical-address-lookup , que toma como argumentos un entorno y una *dirección léxica* que consta de dos números: un *número de cuadro* , que especifica cuántos cuadros pasar, y un *número de desplazamiento* , que especifica cuántas variables pasar en ese cuadro.Lexical-address-lookup producirá el valor de la variable almacenada en esa dirección léxica en relación con el entorno actual. Si agregamos la operación lexical-address-lookup a nuestra máquina, podemos hacer que el compilador genere código que haga referencia a las variables mediante esta operación, en lugar de lookup-variable-value . De manera similar, nuestro código compilado puede usar una nueva¡Operación de conjunto de direcciones léxicas! en lugar de ¡operación de conjunto de valores de variables !

Para generar dicho código, el compilador debe poder determinar la dirección léxica de una variable a la que está a punto de compilar una referencia. La dirección léxica de una variable en un programa depende de dónde se encuentre uno en el código. Por ejemplo, en el siguiente programa, la dirección de x en la expresión < *e1* > es (2,0) - dos cuadros atrás y la primera variable en el cuadro. En ese punto y está en la dirección (0,0) y c está en la dirección (1,2). En la expresión < *e2* >, x está en (1,0), y está en (1,1), y c está en (0,2).

((lambda(x y)   
   (lambda(a b c d e)   
     ((lambda(y z) < *e1* >)   
      < *e2* >   
      (+ c d x))))   
 3   
 4)

Una forma en que el compilador puede producir código que utilice direccionamiento léxico es mantener una estructura de datos llamada *entorno de tiempo de compilación* . Esto lleva un registro de qué variables estarán en qué posiciones en qué marcos en el entorno de tiempo de ejecución cuando se ejecuta una operación de acceso a variables en particular. El entorno de tiempo de compilación es una lista de marcos, cada uno de los cuales contiene una lista de variables. (Por supuesto, no habrá valores vinculados a las variables, ya que los valores no se calculan en tiempo de compilación). El entorno de tiempo de compilación se convierte en un argumento adicional para compilar y se pasa a cada generador de código. La llamada de nivel superior para compilar utiliza un entorno de tiempo de compilación vacío. Cuando se compila un cuerpo lambda , compile-lambda-body extiende el entorno de tiempo de compilación con un marco que contiene los parámetros del procedimiento, de modo que la secuencia que compone el cuerpo se compila con ese entorno extendido. En cada punto de la compilación, compile-variable y compile-assignment utilizan el entorno de tiempo de compilación para generar las direcciones léxicas apropiadas.

Los ejercicios  [5.39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.39) a  [5.43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.43) describen cómo completar este esquema de la estrategia de direccionamiento léxico para incorporar la búsqueda léxica al compilador. El ejercicio  [5.44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.44) describe otro uso del entorno de tiempo de compilación.

**Ejercicio 5.39.**  Escriba un procedimiento lexical-address-lookup que implemente la nueva operación de búsqueda. Debe tomar dos argumentos (una dirección léxica y un entorno de ejecución) y devolver el valor de la variable almacenada en la dirección léxica especificada. Lexical-address-lookup debe señalar un error si el valor de la variable es el símbolo \*unassigned\* . [46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_826) Escriba también un procedimiento lexical-address-set! que implemente la operación que cambia el valor de la variable en una dirección léxica especificada.

**Ejercicio 5.40.**  Modifique el compilador para mantener el entorno de compilación como se describió anteriormente. Es decir, agregue un argumento compile-time-environment a compile y a los diversos generadores de código, y extiéndalo en compile-lambda-body .

**Ejercicio 5.41.**  Escriba un procedimiento find-variable que tome como argumentos una variable y un entorno de compilación y devuelva la dirección léxica de la variable con respecto a ese entorno. Por ejemplo, en el fragmento de programa que se muestra arriba, el entorno de compilación durante la compilación de la expresión < *e1* > es ((yz) (abcde) (xy)) . Find-variable debería producir

(encontrar-variable 'c '((y z) (a b c d e) (x y)))   
*(1 2)*  
  
(encontrar-variable 'x '((y z) (a b c d e) (x y)))   
*(2 0)*  
  
(encontrar-variable 'w '((y z) (a b c d e) (x y)))   
*no encontrado*

**Ejercicio 5.42.**   Utilizando find-variable del ejercicio  [5.41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.41) , reescriba compile-variable y compile-assignment para generar instrucciones de dirección léxica. En los casos en los que find-variable devuelva not-found (es decir, donde la variable no está en el entorno de tiempo de compilación), debe hacer que los generadores de código utilicen las operaciones del evaluador, como antes, para buscar la vinculación. (El único lugar en el que puede estar una variable que no se encuentra en el entorno de tiempo de compilación es en el entorno global, que es parte del entorno de tiempo de ejecución pero no es parte del entorno de tiempo de compilación. [47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_830) Por lo tanto, si lo desea, puede hacer que las operaciones del evaluador busquen directamente en el entorno global, que se puede obtener con la operación (op get-global-environment) , en lugar de hacer que busquen en todo el entorno de tiempo de ejecución que se encuentra en env .) Pruebe el compilador modificado en algunos casos simples, como la combinación lambda anidada al principio de esta sección.

**Ejercicio 5.43.**  [En la sección 4.1.6,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.6) argumentamos  que las definiciones internas para la estructura de bloques no deberían considerarse definiciones "reales" . En cambio, el cuerpo de un procedimiento debería interpretarse como si las variables internas que se están definiendo se instalaran como variables lambda ordinarias inicializadas a sus valores correctos utilizando set!. La sección  [4.1.6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.6) y el ejercicio  [4.16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.16) mostraron cómo modificar el intérprete metacircular para lograr esto escaneando las definiciones internas. Modifique el compilador para que realice la misma transformación antes de compilar el cuerpo de un procedimiento.

**Ejercicio 5.44.**  En esta sección nos hemos centrado en el uso del entorno de tiempo de compilación para producir direcciones léxicas. Pero existen otros usos para los entornos de tiempo de compilación. Por ejemplo, en el ejercicio  [5.38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.38) aumentamos la eficiencia del código compilado mediante la codificación abierta de procedimientos primitivos. Nuestra implementación trató los nombres de los procedimientos de codificación abierta como palabras reservadas. Si un programa volviera a vincular dicho nombre, el mecanismo descrito en el ejercicio  [5.38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.38) lo seguiría codificando como un primitivo, ignorando la nueva vinculación. Por ejemplo, considere el procedimiento

(lambda (+ \* a b x y)   
  (+ (\* a x) (\* b y)))

que calcula una combinación lineal de x e y . Podríamos llamarlo con argumentos +matrix , \*matrix y cuatro matrices, pero el compilador de codificación abierta seguiría codificando abiertamente el + y el \* en (+ (\* ax) (\* by)) como + y \* primitivos . Modifique el compilador de codificación abierta para consultar el entorno de tiempo de compilación a fin de compilar el código correcto para expresiones que involucren los nombres de procedimientos primitivos. (El código funcionará correctamente siempre que el programa no defina o establezca estos nombres).

**[5.5.7 Interfaz del código compilado con el evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_5.5.7)**

Aún no hemos explicado cómo cargar código compilado en la máquina evaluadora ni cómo ejecutarlo. Supondremos que la máquina evaluadora de control explícito se ha definido como en la sección  [5.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.4) , con las operaciones adicionales especificadas en la nota al pie  [38.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#footnote_Temp_809) Implementaremos un procedimientocompilar y listo , que compila una expresión de Scheme, carga el código de objeto resultante en la máquina evaluadora y hace que la máquina ejecute el código en el entorno global del evaluador, imprima el resultado e ingrese al bucle controlador del evaluador. También modificaremos el evaluador para que las expresiones interpretadas puedan llamar a procedimientos compilados además de a los interpretados. Luego podemos colocar un procedimiento compilado en la máquina y usar el evaluador para llamarlo:

(compilar y listo   
 '(define (factorial n)   
    (si (= n 1)   
        1   
        (\* (factorial (- n 1)) n))))   
*;;; Valor de EC-Eval:*   
*ok*   
*;;; Entrada de EC-Eval:*  
(factorial 5)   
*;;; Valor de EC-Eval:*   
*120*

Para permitir que el evaluador maneje procedimientos compilados (por ejemplo, para evaluar la llamada a factorial anterior), necesitamos cambiar el código en apply-dispatch (sección  [5.4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.1) ) para que reconozca los procedimientos compilados (a diferencia de los procedimientos compuestos o primitivos) y transfiera el control directamente al punto de entrada del código compilado: [48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_833)

aplicar-enviar   
  (prueba (op procedimiento-primitivo?) (reg proc))   
  (rama (etiqueta aplicar-primitivo))   
  (prueba (op procedimiento-compuesto?) (reg proc))     
  (rama (etiqueta aplicar-compuesto))   
  (prueba (op procedimiento-compilado?) (reg proc))     
  (rama (etiqueta aplicar-compilado))   
  (goto (etiqueta tipo-procedimiento-desconocido))  
compilado-aplicar   
  (restaurar continuar)   
  (asignar val (op entrada-procedimiento-compilado) (reg proc))   
  (goto (reg val))

Tenga en cuenta la restauración de continue en compiled-apply . Recuerde que el evaluador se organizó de modo que en apply-dispatch , la continuación estaría en la parte superior de la pila. El punto de entrada del código compilado, por otro lado, espera que la continuación esté en continue , por lo que continue debe restaurarse antes de que se ejecute el código compilado.

Para permitirnos ejecutar algún código compilado cuando iniciamos la máquina evaluadora, agregamos una instrucción de bifurcación al comienzo de la máquina evaluadora, lo que hace que la máquina vaya a un nuevo punto de entrada si el registro de bandera está configurado. [49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_834)

  (rama (etiqueta entrada externa))       *; se ramifica si  el indicador  está establecido*  
bucle de impresión y evaluación de lectura   
  (ejecutar (op inicializar pila))   
  ...

La entrada externa supone que la máquina se inicia con val que contiene la ubicación de una secuencia de instrucciones que coloca un resultado en val y termina con (goto (reg continue)) . Comenzar en este punto de entrada salta a la ubicación designada por val , pero primero asigna continue para que la ejecución regrese a print-result , que imprime el valor en val y luego va al comienzo del bucle read-eval-print del evaluador. [50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_835)

entrada externa   
  (ejecutar (op inicializar pila))   
  (asignar entorno (op obtener entorno global))   
  (asignar continuar (etiqueta imprimir resultado))   
  (goto (reg val))

Ahora podemos usar el siguiente procedimiento para compilar una definición de procedimiento, ejecutar el código compilado y ejecutar el bucle read-eval-print para poder probar el procedimiento. Debido a que queremos que el código compilado regrese a la ubicación en continue con su resultado en val , compilamos la expresión con un objetivo de val y un enlace de return . Para transformar el código objeto producido por el compilador en instrucciones ejecutables para la máquina de registro del evaluador, usamos el procedimiento ensamblar del simulador de la máquina de registro (sección  [5.2.2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.2.2) ). Luego inicializamos el registro val para que apunte a la lista de instrucciones, establecemos el indicador para que el evaluador vaya a external-entry e iniciamos el evaluador.

(define (expresión de compilación y ejecución)   
  (let ((instrucciones   
         (ensamblar (declaraciones   
                    (compilar expresión 'val 'retorno))   
                   eceval)))   
    (establecer! el-entorno-global (entorno-de-configuración))   
    (establecer-contenido-del-registro! eceval 'val instrucciones)   
    (establecer-contenido-del-registro! eceval 'bandera verdadero)   
    (iniciar eceval)))

Si hemos configurado la monitorización de la pila, como al final de la sección  [5.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.4) , podemos examinar el uso de la pila del código compilado:

(compilar y listo   
 '(define (factorial n)   
    (si (= n 1)   
        1   
        (\* (factorial (- n 1)) n))))   
  
*(total-pushes = 0 máxima-profundidad = 0)*   
*;;; Valor de EC-Eval:*   
*ok*   
*;;; Entrada de EC-Eval:*  
(factorial 5)   
*(total-pushes = 31 máxima-profundidad = 14)*   
*;;; Valor de EC-Eval:*   
*120*

Compare este ejemplo con la evaluación de (factorial 5) utilizando la versión interpretada del mismo procedimiento, que se muestra al final de la sección  [5.4.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4.4) . La versión interpretada requirió 144 inserciones y una profundidad máxima de pila de 28. Esto ilustra la optimización que resulta de nuestra estrategia de compilación.

**[Interpretación y compilación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_toc_%_sec_Temp_836)**

Con los programas de esta sección, ahora podemos experimentar con las estrategias de ejecución alternativas de interpretación y compilación. [51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_837) Un intérprete eleva la máquina al nivel del programa de usuario; un compilador baja el programa de usuario al nivel del lenguaje de máquina. Podemos considerar el lenguaje Scheme (o cualquier lenguaje de programación) como una familia coherente de abstracciones erigidas sobre el lenguaje de máquina. Los intérpretes son buenos para el desarrollo y la depuración de programas interactivos porque los pasos de la ejecución del programa están organizados en términos de estas abstracciones y, por lo tanto, son más inteligibles para el programador. El código compilado puede ejecutarse más rápido, porque los pasos de la ejecución del programa están organizados en términos del lenguaje de máquina y el compilador es libre de hacer optimizaciones que trasciendan las abstracciones de nivel superior. [52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_838)

Las alternativas de interpretación y compilación también conducen aExisten diferentes estrategias para portar lenguajes a nuevas computadoras. Supongamos que deseamos implementar Lisp para una nueva máquina. Una estrategia es comenzar con el evaluador de control explícito de la sección  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4) y traducir sus instrucciones a instrucciones para la nueva máquina. Una estrategia diferente es comenzar con el compilador y cambiar los generadores de código para que generen código para la nueva máquina. La segunda estrategia nos permite ejecutar cualquier programa Lisp en la nueva máquina compilándolo primero con el compilador que se ejecuta en nuestro sistema Lisp original y vinculándolo con una versión compilada de la biblioteca de tiempo de ejecución. [53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_839) Mejor aún, podemos compilar el compilador mismo y ejecutarlo en la nueva máquina para compilar otros programas Lisp. [54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "footnote_Temp_840) O podemos compilar uno de los intérpretes de la sección  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) para producir un intérprete que se ejecute en la nueva máquina.

**Ejercicio 5.45.**  Al comparar las operaciones de pila utilizadas por el código compilado con las operaciones de pila utilizadas por el evaluador para el mismo cálculo, podemos determinar hasta qué punto el compilador optimiza el uso de la pila, tanto en velocidad (reduciendo el número total de operaciones de pila) como en espacio (reduciendo la profundidad máxima de la pila). Comparar este uso optimizado de la pila con el rendimiento de una máquina de propósito especial para el mismo cálculo nos da una idea de la calidad del compilador.

a. El ejercicio  [5.27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.27) le pidió que determinara, como una función de *n* , la cantidad de empujes y la profundidad máxima de pila que necesita el evaluador para calcular *n* ! utilizando el procedimiento factorial recursivo dado anteriormente. El ejercicio  [5.14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.14) le pidió que hiciera las mismas mediciones para la máquina factorial de propósito especial que se muestra en la figura  [5.11 . Ahora realice el mismo análisis utilizando el procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.11) factorial compilado .

Tome la relación entre la cantidad de impulsos en la versión compilada y la cantidad de impulsos en la versión interpretada, y haga lo mismo para la profundidad máxima de la pila. Dado que la cantidad de operaciones y la profundidad de la pila utilizadas para calcular *n* ! son lineales en *n* , estas relaciones deberían acercarse a constantes a medida que *n* se hace grande. ¿Cuáles son estas constantes? De manera similar, encuentre las relaciones entre el uso de la pila en la máquina de propósito especial y el uso en la versión interpretada.

Compare las proporciones entre código de propósito especial e interpretado y las proporciones entre código compilado e interpretado. Debería descubrir que la máquina de propósito especial funciona mucho mejor que el código compilado, ya que el código del controlador personalizado debería ser mucho mejor que el que produce nuestro compilador de propósito general rudimentario.

b. ¿Puede sugerir mejoras al compilador que lo ayuden a generar código con un rendimiento más cercano al de la versión personalizada?

**Ejercicio 5.46.**  Realice un análisis como el del ejercicio  [5.45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.45) para determinar la efectividad de compilar el procedimiento recursivo de Fibonacci en árbol.

(define (fib n)   
  (si (< n 2)   
      n   
      (+ (fib (- n 1)) (fib (- n 2)))))

en comparación con la eficacia de utilizar la máquina de Fibonacci de propósito especial de la figura  [5.12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_fig_5.12) . (Para la medición del rendimiento interpretado, consulte el ejercicio  [5.29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.29) ). Para Fibonacci, el recurso de tiempo utilizado no es lineal en *n* ; por lo tanto, las proporciones de las operaciones de la pila no se acercarán a un valor límite que sea independiente de *n* .

**Ejercicio 5.47.**   Esta sección describe cómo modificar el evaluador de control explícito para que el código interpretado pueda llamar a procedimientos compilados. Muestre cómo modificar el compilador para que los procedimientos compilados puedan llamar no sólo a procedimientos primitivos y procedimientos compilados, sino también a procedimientos interpretados. Esto requiere modificar compile-procedure-call para manejar el caso de procedimientos compuestos (interpretados). Asegúrese de manejar todas las mismas combinaciones de destino y enlace que en compile-proc-appl . Para hacer la aplicación del procedimiento real, el código necesita saltar al punto de entrada de aplicación compuesta del evaluador . Esta etiqueta no puede ser referenciada directamente en el código objeto (ya que el ensamblador requiere que todas las etiquetas a las que hace referencia el código que está ensamblando estén definidas allí), por lo que agregaremos un registro llamado compapp a la máquina del evaluador para contener este punto de entrada y agregaremos una instrucción para inicializarlo:

  (asignar compapp (etiqueta aplicación-compuesta))   
  (rama (etiqueta entrada-externa))       *; se ramifica si  el indicador  está establecido*  
bucle-de-impresión-evaluación-lectura   
  ...

Para probar el código, comience por definir un procedimiento f que llame a un procedimiento g . Utilice la función de compilación y ejecución para compilar la definición de f e iniciar el evaluador. Ahora, escriba en el evaluador, defina g e intente llamar a f .

**Ejercicio 5.48.**  La interfaz de compilación y ejecución implementada en esta sección es complicada, ya que el compilador solo se puede llamar una vez (cuando se inicia la máquina evaluadora). Aumente la interfaz de compilador-intérprete proporcionando una primitiva de compilación y ejecución que se pueda llamar desde dentro del evaluador de control explícito de la siguiente manera:

*;;; Entrada de EC-Eval:*  
(compilar y ejecutar   
 '(define (factorial n)   
    (si (= n 1)   
        1   
        (\* (factorial (- n 1)) n))))   
*;;; Valor de EC-Eval:*   
*ok*   
*;;; Entrada de EC-Eval:*  
(factorial 5)   
*;;; Valor de EC-Eval:*   
*120*

**Ejercicio 5.49.**   Como alternativa al uso del bucle de lectura-evaluación-impresión del evaluador de control explícito, diseñe una máquina de registro que realice un bucle de lectura-compilación-ejecución-impresión. Es decir, la máquina debe ejecutar un bucle que lea una expresión, la compile, ensamble y ejecute el código resultante e imprima el resultado. Esto es fácil de ejecutar en nuestra configuración simulada, ya que podemos organizar la llamada a los procedimientos de compilación y ensamblaje como ``operaciones de máquina de registro''.

**Ejercicio 5.50.**  Utilice el compilador para compilar el evaluador metacircular de la sección  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) y ejecute este programa utilizando el simulador de máquina de registros. (Para compilar más de una definición a la vez, puede empaquetar las definiciones en un begin ). El intérprete resultante se ejecutará muy lentamente debido a los múltiples niveles de interpretación, pero lograr que todos los detalles funcionen es un ejercicio instructivo.

**Ejercicio 5.51.**  Desarrolle una implementación rudimentaria de Scheme en C (o algún otro lenguaje de bajo nivel de su elección) traduciendo el evaluador de control explícito de la sección  [5.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_5.4) a C. Para ejecutar este código, también deberá proporcionar rutinas de asignación de almacenamiento adecuadas y otro soporte de tiempo de ejecución.

**Ejercicio 5.52.**  Como contrapunto al ejercicio  [5.51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.51) , modifique el compilador para que compile los procedimientos de Scheme en secuencias de instrucciones de C. Compile el evaluador metacircular de la sección  [4.1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1) para producir un intérprete de Scheme escrito en C.

[33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_794) Esta es una afirmación teórica. No estamos afirmando que las rutas de datos del evaluador sean un conjunto de rutas de datos particularmente conveniente o eficiente para una computadora de propósito general. Por ejemplo, no son muy buenas para implementar cálculos de punto flotante de alto rendimiento o cálculos que manipulen intensamente vectores de bits.

[34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_795) En realidad, la máquina que ejecuta el código compilado puede ser más simple que la máquina intérprete, porqueno utilizará los registros exp y unev . El intérprete los utilizó para almacenar fragmentos de expresiones no evaluadas. Sin embargo, con el compilador, estas expresiones se integran en el código compilado que ejecutará la máquina de registros. Por la misma razón,No necesitamos las operaciones de máquina que se ocupan de la sintaxis de expresiones, pero el código compilado utilizará algunas operaciones de máquina adicionales (para representar objetos de procedimiento compilados) que no aparecían en la máquina evaluadora de control explícito.

[35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_797) Sin embargo, tenga en cuenta que nuestro compilador es un programa Scheme y que los procedimientos de sintaxis que utiliza para manipular expresiones son los procedimientos de Scheme reales utilizados con el evaluador metacircular. En cambio, para el evaluador de control explícito, supusimos que había operaciones de sintaxis equivalentes disponibles como operaciones para la máquina de registros. (Por supuesto, cuando simulamos la máquina de registros en Scheme, utilizamos los procedimientos de Scheme reales en nuestra simulación de la máquina de registros).

[36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_803) Este procedimiento utiliza una característica de Lisp llamada*comilla invertida* (o *cuasi-comilla* ) que resulta útil para construir listas. Anteponer una lista con un símbolo de comilla invertida es muy parecido a ponerla entre comillas, excepto que se evalúa cualquier elemento de la lista que esté marcado con una coma.

Por ejemplo, si el valor de linkage es el símbolo branch25 , entonces la expresión `((goto (label ,linkage))) evalúa a la lista ((goto (label branch25))) . De manera similar, si el valor de x es la lista (abc) , entonces `(1 2 ,(car x)) evalúa a la lista (1 2 a) .

[No](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_806) podemos simplemente usar las etiquetastrue-branch,false-branchyafter-ifcomo se muestra arriba, porque podría haber más de unifen el programa.El compilador utiliza el procedimiento make-label para generar etiquetas. Make-label toma un símbolo como argumento y devuelve un nuevo símbolo que comienza con el símbolo dado. Por ejemplo, las llamadas sucesivas a (make-label 'a) devolverían a1 , a2 , y así sucesivamente. Make-label se puede implementar de manera similar a la generación de nombres de variables únicos en el lenguaje de consulta, de la siguiente manera:

(define el contador de etiquetas 0)   
  
(define (nuevo número de etiqueta)   
  (¡establece! contador de etiquetas (+ 1 contador de etiquetas))   
  contador de etiquetas)  
  
(define (make-label nombre)   
  (string->symbol   
    (string-append (símbolo->string nombre)   
                   (number->string (new-label-number)))))

Necesitamos operaciones de máquina para implementar una estructura de datos para representar procedimientos compilados, análoga a la estructura para procedimientos compuestos descrita en la  [sección](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_809) [4.1.3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_sec_4.1.3):

(define (make-compiled-procedure entry env)   
  (lista 'compiled-procedure entry env))  
  
(define (procedimiento compilado? proc)   
  (lista etiquetada? proc 'procedimiento compilado))  
  
(define (procedimiento compilado entrada c-proc) (cadr c-proc))  
  
(define (entorno de procedimiento compilado c-proc) (caddr c-proc))

[En realidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_812) , señalamos un error cuando el objetivo no esvaly el enlace esreturn, ya que el único lugar donde solicitamosde retornoes en los procedimientos de compilación, y nuestra convención es que los procedimientos devuelven sus valores enval.

[40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_813) Haciendo unaEl compilador puede parecer una idea sencilla de generar código recursivo de cola, pero la mayoría de los compiladores de lenguajes comunes, incluidos C y Pascal, no lo hacen y, por lo tanto, estos lenguajes no pueden representar procesos iterativos solo en términos de llamadas a procedimientos.La principal característica de la recursión de cola en estos lenguajes es que sus implementaciones utilizan la pila para almacenar argumentos de procedimientos y variables locales, así como direcciones de retorno. Las implementaciones de Scheme descritas en este libro almacenan argumentos y variables en memoria para que sean recolectados como basura. La razón para utilizar la pila para variables y argumentos es que evita la necesidad de recolección de basura en lenguajes que de otra manera no la requerirían, y generalmente se cree que es más eficiente. Los compiladores Lisp sofisticados pueden, de hecho, utilizar la pila para argumentos sin destruir la recursión de cola. (Ver(Consulte Hanson 1990 para obtener una descripción). También existe cierto debate sobre si la asignación de pila es en realidad más eficiente que la recolección de basura en primer lugar, pero los detalles parecen depender de puntos finos de la arquitectura informática. (ConsulteApelación 1987 y(Véase Miller y Rozas 1994 para puntos de vista opuestos sobre este tema).

[41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_814) La variableall-regs está vinculado a la lista de nombres de todos los registros:

(definir all-regs '(env proc val argl continuar))

[42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_815) Tenga en cuenta quelas llamadasde conservaciónse agregancon tresargumentos. Aunque la definición de append que se muestra en este libro acepta solo dos argumentos, Scheme proporciona de manera estándar un procedimiento append que toma una cantidad arbitraria de argumentos.

Hemos utilizado el mismo símbolo+ [aquí para indicar tanto el procedimiento del lenguaje fuente como la operación de la máquina. En general ,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_822) no habrá una correspondencia uno a uno entre los primitivos del lenguaje fuente y los primitivos de la máquina.

[En general ,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_823) convertir los primitivos en palabras reservadas es una mala idea, ya que el usuario no puede volver a vincular estos nombres a procedimientos diferentes. Además, si añadimos palabras reservadas a un compilador que está en uso, los programas existentes que definen procedimientos con estos nombres dejarán de funcionar. Consulte el ejercicio [5.44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.44)para obtener ideas sobre cómo evitar este problema.

[45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_824) Esto no es cierto si permitimos definiciones internas, a menos que las analicemos. Véase el ejercicio [5.43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.43).

[46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_826) Esta es la modificación de la búsqueda de variables.Se requiere si implementamos el método de escaneo para eliminar definiciones internas (ejercicio  [5.43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.43) ). Necesitaremos eliminar estas definiciones para que funcione el direccionamiento léxico.

[47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_830) Las direcciones léxicas no se pueden utilizar para acceder a las variables en el entorno global, porque estos nombres se pueden definir y redefinir de forma interactiva en cualquier momento. Con las definiciones internas escaneadas, como en el ejercicio [5.43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.43), las únicas definiciones que ve el compilador son las del nivel superior, que actúan sobre el entorno global. La compilación de una definición no hace que el nombre definido se introduzca en el entorno de tiempo de compilación.

[48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_833) Por supuesto, tanto los procedimientos compilados como los procedimientos interpretados son compuestos (no primitivos). Para que sea compatible con la terminología utilizada en el evaluador de control explícito, en esta sección utilizaremos "compuesto" para referirnos a interpretados (en contraposición a compilados).

[49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_834) Ahora que la máquina evaluadora comienza con unarama, siempre debemos inicializar elde indicadoresantes de iniciar la máquina evaluadora. Para iniciar la máquina en su bucle de lectura-evaluación-impresión normal, podríamos usar

(define (start-eceval)   
  (set! the-global-environment (setup-environment))   
  (set-register-contents! eceval 'flag false)   
  (start eceval))

[50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_835) Dado que un procedimiento compilado es un objeto que el sistema puede intentar imprimir, también modificamos la operación de impresión del sistemauser-print(de la sección [4.1.4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_sec_4.1.4)) para que no intente imprimir los componentes de un procedimiento compilado:

(define (objeto de impresión de usuario)   
  (cond ((objeto de procedimiento compuesto)   
         (mostrar (lista 'objeto de procedimiento compuesto   
                        (objeto de parámetros de procedimiento)   
                        (objeto de cuerpo de procedimiento)   
                        '<entorno de procedimiento>)))   
        ((objeto de procedimiento compilado)   
         (mostrar '<procedimiento compilado>))   
        (de lo contrario (objeto de visualización))))

Podemos hacerlo aún mejor ampliando el compilador para permitir que el código compilado llame a procedimientos interpretados. Véase el  [ejercicio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_837) [5.47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_thm_5.47).

[52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_838) Independientemente de la estrategia de ejecución,Se puede incurrir en una sobrecarga significativa si insistimos en que los errores encontrados en la ejecución de un programa de usuario se detecten y señalen, en lugar de permitir que se mate el sistema o se produzcan respuestas incorrectas. Por ejemplo, una referencia a una matriz fuera de límites se puede detectar comprobando la validez de la referencia antes de realizarla. Sin embargo, la sobrecarga de la comprobación puede ser muchas veces el costo de la referencia a la matriz en sí, y un programador debe sopesar la velocidad frente a la seguridad al determinar si es deseable dicha comprobación. Un buen compilador debe ser capaz de producir código con dichas comprobaciones, debe evitar comprobaciones redundantes y debe permitir a los programadores controlar el alcance y el tipo de comprobación de errores en el código compilado.

Los compiladores de lenguajes populares, como C y C++, prácticamente no incluyen operaciones de comprobación de errores en el código en ejecución, para que todo funcione lo más rápido posible. Como resultado, los programadores deben proporcionar explícitamente la comprobación de errores. Desafortunadamente, la gente a menudo se olvida de hacerlo, incluso en aplicaciones críticas donde la velocidad no es una limitación. Sus programas tienen vidas rápidas y peligrosas. Por ejemplo, el famosoEl "gusano" que paralizó Internet en 1988 explotó laEl sistema operativo UNIX *T M* no comprueba si el búfer de entrada tienese desbordó en el demonio de dedo. (Véase Spafford 1989.)

[53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "call_footnote_Temp_839) Por supuesto, tanto con la interpretación como con la estrategia de compilación también debemos implementar para la nueva máquina la asignación de almacenamiento, la entrada y la salida, y todas las diversas operaciones que consideramos "primitivas" en nuestra discusión del evaluador y el compilador. Una estrategia para minimizar el trabajo aquí es escribir tantas de estas operaciones como sea posible en Lisp y luego compilarlas para la nueva máquina. En última instancia, todo se reduce a un pequeño núcleo (como la recolección de basura y el mecanismo para aplicar las primitivas reales de la máquina) que se codifica a mano para la nueva máquina.

Esta estrategia conduce a divertidas pruebas de corrección del compilador, como comprobar si la compilación de un programa en la nueva máquina, utilizando el compilador compilado, es idéntica a la compilación del programa en el sistema Lisp original. Rastrear la fuente de las diferencias es divertido, pero a menudo frustrante, porque los resultados son extremadamente sensibles a los detalles minúsculos [.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#call_footnote_Temp_840)

[**Referencias**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_Temp_849)

Abelson, Harold, Andrew Berlin, Jacob Katzenelson, William McAllister, Guillermo Rozas, Gerald Jay Sussman y Jack Wisdom. 1992. El kit de herramientas de supercomputadoras: un marco general para computación de propósito especial. *Revista internacional de electrónica de alta velocidad* 3(3):337-361.

Allen, John. 1978. *Anatomía del ceceo.* Nueva York: McGraw-Hill.

ANSI X3.226-1994. *Estándar nacional americano para sistemas de información – Lenguaje de programación – Common Lisp.*

Appel, Andrew W. 1987. La recolección de basura puede ser más rápida que la asignación de pilas. *Information Processing Letters* 25(4):275-279.

Backus, John. 1978. ¿Puede la programación liberarse del estilo von Neumann? *Communications of the ACM* 21(8):613-641.

Baker, Henry G., Jr. 1978. Procesamiento de listas en tiempo real en una computadora serial. *Communications of the ACM* 21(4):280-293.

Batali, John, Neil Mayle, Howard Shrobe, Gerald Jay Sussman y Daniel Weise. 1982. La arquitectura Scheme-81: sistema y chip. En *Actas de la Conferencia del MIT sobre Investigación Avanzada en VLSI,* editado por Paul Penfield, Jr. Dedham, MA: Artech House.

Borning, Alan. 1977. ThingLab: un sistema orientado a objetos para crear simulaciones utilizando restricciones. En *Actas de la 5.ª Conferencia conjunta internacional sobre inteligencia artificial.*

Borodin, Alan, e Ian Munro. 1975. *La complejidad computacional de problemas algebraicos y numéricos.* Nueva York: American Elsevier.

Chaitin, Gregory J. 1975. Aleatoriedad y prueba matemática. *Scientific American* 232(5):47-52.

Church, Alonzo. 1941. *Los cálculos de la conversión lambda.* Princeton, NJ: Princeton University Press.

Clark, Keith L. 1978. La negación como fracaso. En *Logic and Data Bases.* Nueva York: Plenum Press, pp. 293-322.

Clinger, William. 1982. La llamada no determinista por necesidad no es ni perezosa ni por su nombre. En *Actas del Simposio ACM sobre Lisp y programación funcional,* págs. 226-234.

Clinger, William y Jonathan Rees. 1991. Macros que funcionan. En *Actas de la Conferencia ACM de 1991 sobre Principios de Lenguajes de Programación,* págs. 155-162.

Colmerauer A., ​​H. Kanoui, R. Pasero y P. Roussel. 1973. Un sistema de comunicación hombre-máquina en francés. Informe técnico, Groupe Intelligence Artificielle, Université d'Aix Marseille, Luminy.

Cormen, Thomas, Charles Leiserson y Ronald Rivest. 1990. *Introducción a los algoritmos.* Cambridge, MA: MIT Press.

Darlington, John, Peter Henderson y David Turner. 1982. *Programación funcional y sus aplicaciones.* Nueva York: Cambridge University Press.

Dijkstra, Edsger W. 1968a. La estructura del sistema de multiprogramación ``THE''. *Comunicaciones de la ACM* 11(5):341-346.

Dijkstra, Edsger W. 1968b. Procesos secuenciales cooperativos. En *Lenguajes de programación* , editado por F. Genuys. Nueva York: Academic Press, págs. 43-112.

Dinesman, Howard P. 1968. *Rompecabezas matemáticos superiores* . Nueva York: Simon and Schuster.

deKleer, Johan, Jon Doyle, Guy Steele y Gerald J. Sussman. 1977. AMORD: Control explícito del razonamiento. En *Actas del Simposio ACM sobre Inteligencia Artificial y Lenguajes de Programación,* págs. 116-125.

Doyle, Jon. 1979. Un sistema de mantenimiento de la verdad. *Inteligencia Artificial* 12:231-272.

Feigenbaum, Edward y Howard Shrobe. 1993. El Proyecto Nacional Japonés de Quinta Generación: Introducción, estudio y evaluación. En *Future Generation Computer Systems,* vol. 9, págs. 105-117.

Feeley, Marc. 1986. Deux Approches Ã l'implantation du language Scheme. Tesis de maestría, Universidad de Montreal.

Feeley, Marc y Guy Lapalme. 1987. Uso de cierres para la generación de código. *Journal of Computer Languages* ​​12(1):47-66.

Feller, William. 1957. *Introducción a la teoría de la probabilidad y sus aplicaciones,* volumen 1. Nueva York: John Wiley & Sons.

Fenichel, R., y J. Yochelson. 1969. Un recolector de basura Lisp para sistemas informáticos de memoria virtual. *Communications of the ACM* 12(11):611-612.

Floyd, Robert. 1967. Algoritmos no deterministas. *JACM,* 14(4):636-644.

Forbus, Kenneth D. y Johan deKleer. 1993. *Building Problem Solvers.* Cambridge, MA: MIT Press.

Friedman, Daniel P. y David S. Wise. 1976. CONS no debería evaluar sus argumentos. En *Automata, Languages, and Programming: Third International Colloquium,* editado por S. Michaelson y R. Milner, pp. 257-284.

Friedman, Daniel P., Mitchell Wand y Christopher T. Haynes. 1992. *Fundamentos de lenguajes de programación.* Cambridge, MA: MIT Press/McGraw-Hill.

Gabriel, Richard P. 1988. El por qué de Y. *Lisp Pointers* 2(2):15-25.

Goldberg, Adele y David Robson. 1983. *Smalltalk-80: El lenguaje y su implementación.* Reading, MA: Addison-Wesley.

Gordon, Michael, Robin Milner y Christopher Wadsworth. 1979. *LCF de Edimburgo.* Lecture Notes in Computer Science, volumen 78. Nueva York: Springer-Verlag.

Gray, Jim y Andreas Reuter. 1993. *Procesamiento de transacciones: conceptos y modelos.* San Mateo, CA: Morgan-Kaufman.

Green, Cordell. 1969. Aplicación de la demostración de teoremas a la resolución de problemas. En *Actas de la Conferencia Conjunta Internacional sobre Inteligencia Artificial,* págs. 219-240.

Green, Cordell y Bertram Raphael. 1968. El uso de técnicas de demostración de teoremas en sistemas de preguntas y respuestas. En *Actas de la Conferencia Nacional de la ACM,* págs. 169-181.

Griss, Martin L. 1981. Portable Standard Lisp, una breve descripción general. Utah Symbolic Computation Group Operating Note 58, Universidad de Utah.

Guttag, John V. 1977. Tipos de datos abstractos y desarrollo de estructuras de datos. *Communications of the ACM* 20(6):397-404.

Hamming, Richard W. 1980. *Codificación y teoría de la información.* Englewood Cliffs, NJ: Prentice-Hall.

Hanson, Christopher P. 1990. Asignación eficiente de pila para lenguajes recursivos de cola. En *Actas de la Conferencia ACM sobre Lisp y programación funcional,* págs. 106-118.

Hanson, Christopher P. 1991. Una macroinstalación de cierres sintácticos. *Lisp Pointers,* 4(3).

Hardy, Godfrey H. 1921. Srinivasa Ramanujan. *Actas de la Sociedad Matemática de Londres* XIX(2).

Hardy, Godfrey H. y EM Wright. 1960. *Introducción a la teoría de números.* Cuarta edición. Nueva York: Oxford University Press.

Havender, J. 1968. Cómo evitar bloqueos en sistemas multitarea. *IBM Systems Journal* 7(2):74-84.

Hearn, Anthony C. 1969. Standard Lisp. Informe técnico AIM-90, Proyecto de Inteligencia Artificial, Universidad de Stanford.

Henderson, Peter. 1980. *Programación funcional: aplicación e implementación.* Englewood Cliffs, NJ: Prentice-Hall.

Henderson, Peter, 1982. Geometría funcional. En *Acta de la conferencia del Simposio ACM de 1982 sobre Lisp y programación funcional,* págs. 179-187.

Hewitt, Carl E. 1969. PLANNER: Un lenguaje para demostrar teoremas en robots. En *Actas de la Conferencia Conjunta Internacional sobre Inteligencia Artificial,* págs. 295-301.

Hewitt, Carl E. 1977. Visualización de las estructuras de control como patrones de transmisión de mensajes. *Journal of Artificial Intelligence* 8(3):323-364.

Hoare, CAR 1972. Prueba de corrección de representaciones de datos. *Acta Informatica* 1(1).

Hodges, Andrew. 1983. *Alan Turing: El enigma.* Nueva York: Simon and Schuster.

Hofstadter, Douglas R. 1979. *Gödel, Escher, Bach: Una eterna trenza dorada.* Nueva York: Basic Books.

Hughes, RJM 1990. Por qué es importante la programación funcional. En *Research Topics in Functional Programming* , editado por David Turner. Reading, MA: Addison-Wesley, págs. 17-42.

IEEE Std 1178-1990. 1990. *Estándar IEEE para el lenguaje de programación Scheme.*

Ingerman, Peter, Edgar Irons, Kirk Sattley y Wallace Feurzeig; con la ayuda de M. Lind, Herbert Kanner y Robert Floyd. 1960. THUNKS: A way of compiling procedure statement, with some comments on procedure statement. Manuscrito inédito. (También, comunicación privada de Wallace Feurzeig.)

Kaldewaij, Anne. 1990. *Programación: la derivación de algoritmos.* Nueva York: Prentice-Hall.

Kohlbecker, Eugene Edmund, Jr. 1986. Extensiones sintácticas en el lenguaje de programación Lisp. Tesis doctoral, Universidad de Indiana.

Konopasek, Milos y Sundaresan Jayaraman. 1984. *El libro TK!Solver: una guía para la resolución de problemas en ciencia, ingeniería, negocios y educación.* Berkeley, CA: Osborne/McGraw-Hill.

Knuth, Donald E. 1973. *Algoritmos fundamentales.* Volumen 1 de *El arte de la programación informática.* Segunda edición. Reading, MA: Addison-Wesley.

Knuth, Donald E. 1981. *Algoritmos seminuméricos.* Volumen 2 de *El arte de la programación informática.* Segunda edición. Reading, MA: Addison-Wesley.

Kowalski, Robert. 1973. La lógica de predicados como lenguaje de programación. Informe técnico 70, Departamento de Lógica Computacional, Escuela de Inteligencia Artificial, Universidad de Edimburgo.

Kowalski, Robert. 1979. *Lógica para la resolución de problemas.* Nueva York: North-Holland.

Lamport, Leslie. 1978. Tiempo, relojes y ordenación de eventos en un sistema distribuido. *Communications of the ACM* 21(7):558-565.

Lampson, Butler, JJ Horning, R. London, JG Mitchell y GK Popek. 1981. Informe sobre el lenguaje de programación Euclid. Informe técnico, Computer Systems Research Group, Universidad de Toronto.

Landin, Peter. 1965. Una correspondencia entre Algol 60 y la notación lambda de Church: Parte I. *Comunicaciones de la ACM* 8(2):89-101.

Lieberman, Henry y Carl E. Hewitt. 1983. Un recolector de basura en tiempo real basado en la vida útil de los objetos. *Communications of the ACM* 26(6):419-429.

Liskov, Barbara H. y Stephen N. Zilles. 1975. Técnicas de especificación para abstracciones de datos. *IEEE Transactions on Software Engineering* 1(1):7-19.

McAllester, David Allen. 1978. Un sistema de mantenimiento de la verdad de tres valores. Memo 473, Laboratorio de Inteligencia Artificial del MIT.

McAllester, David Allen. 1980. Una perspectiva sobre el mantenimiento de la verdad. Memo 551, Laboratorio de Inteligencia Artificial del MIT.

McCarthy, John. 1960. Funciones recursivas de expresiones simbólicas y su cálculo por máquina. *Communications of the ACM* 3(4):184-195.

McCarthy, John. 1967. Una base para una teoría matemática de la computación. En *Computer Programing and Formal Systems* , editado por P. Braffort y D. Hirschberg. Holanda Septentrional.

McCarthy, John. 1978. La historia de Lisp. En *Actas de la Conferencia SIGPLAN de la ACM sobre la historia de los lenguajes de programación.*

McCarthy, John, PW Abrahams, DJ Edwards, TP Hart y MI Levin. 1965. *Manual del programador de Lisp 1.5.* Segunda edición. Cambridge, MA: MIT Press.

McDermott, Drew y Gerald Jay Sussman. 1972. Manual de referencia de Conniver. Memo 259, Laboratorio de Inteligencia Artificial del MIT.

Miller, Gary L. 1976. Hipótesis de Riemann y pruebas de primalidad. *Journal of Computer and System Sciences* 13(3):300-317.

Miller, James S., y Guillermo J. Rozas. 1994. La recolección de basura es rápida, pero una pila es más rápida. Memo 1462, Laboratorio de Inteligencia Artificial del MIT.

Moon, David. 1978. Manual de referencia de MacLisp, versión 0. Informe técnico, Laboratorio de Ciencias de la Computación del MIT.

Moon, David y Daniel Weinreb. 1981. Manual de la máquina Lisp. Informe técnico, Laboratorio de Inteligencia Artificial del MIT.

Morris, JH, Eric Schmidt y Philip Wadler. 1980. Experiencia con un lenguaje de procesamiento de cadenas aplicativo. En *Actas del 7º Simposio anual ACM SIGACT/SIGPLAN sobre los principios de los lenguajes de programación.*

Phillips, Hubert. 1934. *El libro de problemas de la esfinge* . Londres: Faber and Faber.

Pitman, Kent. 1983. Manual revisado de MacLisp (edición del sábado por la tarde). Informe técnico 295, Laboratorio de Ciencias de la Computación del MIT.

Rabin, Michael O. 1980. Algoritmo probabilístico para probar la primalidad. *Journal of Number Theory* 12:128-138.

Raymond, Eric. 1993. *El nuevo diccionario del hacker.* Segunda edición. Cambridge, MA: MIT Press.

Raynal, Michel. 1986. *Algoritmos para exclusión mutua.* Cambridge, MA: MIT Press.

Rees, Jonathan A. y Norman I. Adams IV. 1982. T: Un dialecto de Lisp o Lambda: La herramienta de software definitiva. En *Acta de la conferencia del Simposio ACM de 1982 sobre Lisp y programación funcional,* págs. 114-122.

Rees, Jonathan y William Clinger (eds). 1991. El informe revisado sobre el lenguaje algorítmico Scheme. *Lisp Pointers,* 4(3).

Rivest, Ronald, Adi Shamir y Leonard Adleman. 1977. Un método para obtener firmas digitales y criptosistemas de clave pública. Nota técnica LCS/TM82, Laboratorio de Ciencias de la Computación del MIT.

Robinson, JA 1965. Una lógica orientada a máquinas basada en el principio de resolución. *Revista de la ACM* 12(1):23.

Robinson, JA 1983. Programación lógica: pasado, presente y futuro. *Computación de nueva generación* 1:107-124.

Spafford, Eugene H. 1989. El gusano de Internet: crisis y consecuencias. *Comunicaciones de la ACM* 32(6):678-688.

Steele, Guy Lewis, Jr. 1977. Desacreditando el mito de la «llamada a un procedimiento costoso». En *Actas de la Conferencia Nacional de la ACM,* págs. 153-62.

Steele, Guy Lewis, Jr. 1982. Una descripción general de Common Lisp. En *Actas del Simposio ACM sobre Lisp y programación funcional,* págs. 98-107.

Steele, Guy Lewis, Jr. 1990. *Common Lisp: The Language.* Segunda edición. Prensa digital.

Steele, Guy Lewis, Jr., y Gerald Jay Sussman. 1975. Scheme: Un intérprete para el cálculo lambda extendido. Memo 349, Laboratorio de Inteligencia Artificial del MIT.

Steele, Guy Lewis, Jr., Donald R. Woods, Raphael A. Finkel, Mark R. Crispin, Richard M. Stallman y Geoffrey S. Goodfellow. 1983. *El diccionario del hacker.* Nueva York: Harper & Row.

Stoy, Joseph E. 1977. *Semántica denotacional.* Cambridge, MA: MIT Press.

Sussman, Gerald Jay y Richard M. Stallman. 1975. Técnicas heurísticas en el análisis de circuitos asistido por computadora. *IEEE Transactions on Circuits and Systems* CAS-22(11):857-865.

Sussman, Gerald Jay y Guy Lewis Steele Jr. 1980. Restricciones: un lenguaje para expresar descripciones casi jerárquicas. *AI Journal* 14:1-39.

Sussman, Gerald Jay y Jack Wisdom. 1992. Evolución caótica del sistema solar. *Science* 257:256-262.

Sussman, Gerald Jay, Terry Winograd y Eugene Charniak. 1971. Manual de referencia de Microplanner. Memo 203A, Laboratorio de Inteligencia Artificial del MIT.

Sutherland, Ivan E. 1963. SKETCHPAD: Un sistema de comunicación gráfica hombre-máquina. Informe técnico 296, MIT Lincoln Laboratory.

Teitelman, Warren. 1974. Manual de referencia de Interlisp. Informe técnico, Centro de investigación de Xerox en Palo Alto.

Thatcher, James W., Eric G. Wagner y Jesse B. Wright. 1978. Especificación de tipos de datos: Parametrización y el poder de las técnicas de especificación. En *Acta de la Décima Conferencia Anual de la ACM sobre Teoría de la Computación* , págs. 119-132. Turner, David. 1981. El futuro de los lenguajes aplicativos. En *Actas de la 3.ª Conferencia Europea sobre Informática,* Lecture Notes in Computer Science, volumen 123. Nueva York: Springer-Verlag, págs. 334-348.

Wand, Mitchell. 1980. Estrategias de transformación de programas basadas en la continuidad. *Journal of the ACM* 27(1):164-180.

Waters, Richard C. 1979. Un método para analizar programas de bucle. *IEEE Transactions on Software Engineering* 5(3):237-247.

Winograd, Terry. 1971. Procedimientos como representación de datos en un programa informático para la comprensión del lenguaje natural. Informe técnico AI TR-17, Laboratorio de Inteligencia Artificial del MIT.

Winston, Patrick. 1992. *Inteligencia artificial* . Tercera edición. Reading, MA: Addison-Wesley.

Zabih, Ramin, David McAllester y David Chapman. 1987. Lisp no determinista con retroceso dirigido por dependencia. *AAAI-87* , págs. 59-64.

Zippel, Richard. 1979. Algoritmos probabilísticos para polinomios dispersos. Tesis doctoral, Departamento de Ingeniería Eléctrica y Ciencias de la Computación, MIT.

Zippel, Richard. 1993. *Cálculo polinomial efectivo.* Boston, MA: Kluwer Academic Publishers.

[**Lista de ejercicios**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_Temp_850)

[1,1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.1)   
[1,2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.2)   
[1,3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.3)   
[1,4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.4)   
[1,5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.5)   
[1,6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.6)   
[1,7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.7)   
[1,8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.8)   
[1,9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.9)   
[1,10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.10)   
[1,11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.11)   
[1,12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.12)   
[1,13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.13)   
[1,14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.14)   
[1,15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.15)   
[1,16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.16)   
[1,17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.17)   
[1,18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.18)   
[1,19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.19)   
[1,20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.20)   
[1,21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.21)   
[1,22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.22)   
[1,23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.23)   
[1,24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.24)   
[1,25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.25)   
[1,26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.26)   
[1,27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.27)   
[1,28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.28)   
[1,29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.29)   
[1,30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.30) [1,31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.32) [1,32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.33) [1,33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.35) [1,34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.36) [1,35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.37) [1,36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.38) [1,37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.41) [1,38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.42) [1,39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.2) [1,40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.39) [1,41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.34)   
[1,42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.31) [1,43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.43) [1,44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.44) [1,45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.45) [1,46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.46) [2,1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.1) [2,2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_1.40) [2,3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.3) [2,4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.4) [2,5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.5) [2,6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.6) [2,7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.7) [2,8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.8) [2,9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.9) [2,10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.10) [2,11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.11) [2,12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.12) [2,13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.13) [2,14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.14) [2,15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.16) [2,16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.17) [2,17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.18) [2,18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.23) [2,19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.19) [2,20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.20) [2,21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.21) [2,22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.22) [2,23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.24) [2,24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.15) [2,25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.25) [2,26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.26) [2,27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.27) [2,28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.28) [2,29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.29) [2,30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.30) [2,31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.31) [2,32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.32) [2,33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.33) [2,34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.34) [2,35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.35) [2,36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.36) [2,37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.37) [2,38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.45) [2,39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.47) [2,40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.39) [2,41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.51) [2,42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.48) [2,43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.43) [2,44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.44) [2,45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.38) [2,46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.46) [2,47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.40) [2,48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.41) [2,49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.42) [2,50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.49) [2,51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.50) [2,52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.52) [2,53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.53) [2,54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.54) [2,55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.55) [2,56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.56) [2,57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.57) [2,58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.58) [2,59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.59) [2,60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.60) [2,61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.61) [2,62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.62) [2,63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.63) [2,64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.67) [2,65](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.66) [2,66](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.70) [2,67](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.68) [2,68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.69) [2,69](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.78) [2,70](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.64) [2,71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.65) [2,72](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.71) [2,73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.73) [2,74](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.74) [2,75](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.75) [2,76](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.76) [2,77](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.77) [2,78](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.72) [2,79](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.79) [2,80](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.80) [2,81](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.81) [2,82](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.82) [2,83](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.83) [2,84](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.84) [2,85](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.85) [2,86](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.86) [2,87](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.87) [2,88](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.88) [2,89](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.89) [2,90](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.90) [2,91](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.91) [2,92](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.92) [2,93](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.93) [2,94](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.94) [2,95](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.95) [2,96](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.96) [2,97](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_2.97) [3,1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.1) [3,2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.2) [3,3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.3) [3,4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.4) [3,5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.5) [3,6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.6) [3,7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.7) [3,8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.8) [3,9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.9) [3,10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.10) [3,11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.11) [3,12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.12) [3,13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.13) [3,14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.14) [3,15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.15) [3,16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.16) [3,17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.17) [3,18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.18) [3,19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.26) [3,20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.31) [3,21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.20) [3,22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.21) [3,23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.23) [3,24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.24) [3,25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.25) [3,26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.27) [3,27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.19) [3,28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.22) [3,29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.28) [3,30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.29) [3,31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.30) [3,32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.32) [3,33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.33) [3,34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.34) [3,35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.35)  
[3,36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.36)   
[3,37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.37)   
[3,38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.38)   
[3,39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.39)   
[3,40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.40)   
[3,41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.41)   
[3,42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.42)   
[3,43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.43)   
[3,44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.44) [3,45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.46) [3,46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.47) [3,47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.48) [3,48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.49) [3,49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.52) [3,50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.58)   
[3,51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.45) [3,52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.51) [3,53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.53) [3,54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.54) [3,55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.55) [3,56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.56) [3,57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.57) [3,58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.50) [3,59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.59) [3,60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.60) [3,61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.61) [3,62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.62) [3,63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.63) [3,64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.64) [3,65](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.65) [3,66](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.67) [3,67](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.69) [3,68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.76) [3,69](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.70) [3,70](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.74) [3,71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.71) [3,72](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.72) [3,73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.73) [3,74](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.66) [3,75](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.68) [3,76](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.81) [3,77](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.77) [3,78](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.78) [3,79](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.79) [3,80](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.80) [3,81](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.75) [3,82](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_3.82) [4,1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.1) [4,2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.2) [4,3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.3) [4,4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.4) [4,5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.5) [4,6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.6) [4,7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.7) [4,8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.8) [4,9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.9) [4,10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.10) [4,11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.11) [4,12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.12) [4,13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.13) [4,14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.14) [4,15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.15) [4,16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.16) [4,17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.17) [4,18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.18) [4,19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.19) [4,20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.20) [4,21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.21) [4,22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.22) [4,23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.23) [4,24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.24) [4,25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.25) [4,26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.26) [4,27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.27) [4,28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.28) [4,29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.29) [4,30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.30) [4,31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.31) [4,32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.32) [4,33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.33) [4,34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.35) [4,35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.36) [4,36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.38) [4,37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.45) [4,38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.48) [4,39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.39) [4,40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.34) [4,41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.43) [4,42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.49) [4,43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.44) [4,44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.37) [4,45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.40) [4,46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.41) [4,47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.46) [4,48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.47) [4,49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.42) [4,50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.50) [4,51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.51) [4,52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.52) [4,53](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.53) [4,54](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.54) [4,55](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.55) [4,56](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.56) [4,57](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.57) [4,58](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.58) [4,59](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.59) [4,60](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.68) [4,61](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.69) [4,62](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.70) [4,63](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.74) [4,64](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.66) [4,65](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.71) [4,66](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.67) [4,67](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.72) [4,68](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.60) [4,69](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.61) [4,70](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.65) [4,71](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.64) [4,72](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.63) [4,73](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.62) [4,74](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.73) [4,75](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.75) [4,76](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.76) [4,77](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.77) [4,78](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.78) [4,79](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_4.79) [5,1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.1) [5,2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.2) [5,3](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.3) [5,4](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.4) [5,5](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.5) [5,6](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.6) [5,7](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.7) [5,8](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.8) [5,9](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.9) [5,10](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.10) [5,11](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.11) [5,12](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.12) [5,13](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.13) [5,14](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.14) [5,15](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.15) [5,16](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.16) [5,17](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.17) [5,18](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.18) [5,19](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.19) [5,20](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.20) [5,21](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.21) [5,22](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.22) [5,23](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.23) [5,24](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.24) [5,25](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.25) [5,26](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.26) [5,27](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.27) [5,28](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.28) [5,29](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.29) [5,30](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.42) [5,31](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.32) [5,32](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.38) [5,33](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.34) [5,34](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.35) [5,35](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.39) [5,36](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.36) [5,37](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.37) [5,38](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.40) [5,39](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.31) [5,40](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.46) [5,41](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.30) [5,42](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.47) [5,43](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.43) [5,44](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.33) [5,45](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.44) [5,46](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.45) [5,47](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.41) [5,48](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.48) [5,49](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.49) [5,50](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.50) [5,51](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.51)  
[5.52](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_thm_5.52)

[**Índice**](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-4.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_toc_%_chap_Temp_851)

|  |
| --- |
| Cualquier inexactitud en este índice puede explicarse por el hecho de que ha sido preparado con ayuda de un ordenador.  Donald E. Knuth, *Algoritmos fundamentales* (volumen 1 de *El arte de la programación informática* ) |

[! en nombres](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc" \l "%_idx_2872)   
[" (comillas dobles)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2008)  
cálculo, *véase* cálculo lambda [notación para funciones matemáticas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1122) , *véase* suma pi [(sigma) notación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_986) , *véase* theta [' (comillas simples)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2000) [leer y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4526) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5406) [\* (procedimiento de multiplicación primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_168) [+ (procedimiento de adición primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_162) [, (coma, usada con comillas invertidas)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6288) [- (procedimiento de resta primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_170) [como negación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_410) [/ (procedimiento de división primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_176) [< (predicado de comparación numérica primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_394) [= (predicado de igualdad numérica primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_396) [=número?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2090) [=cero? (genérico)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2568) [para polinomios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2752) [> (predicado de comparación numérica primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_392) [>=](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_470) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_472) [? , en nombres de predicados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_524) [#f](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_378) [#t](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_372) [` (comillas invertidas)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6286) ; , *véase* punto y coma  
[](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1122)  
  
  


[Abelson, Harold](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_106)   
[abs](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_358) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_414) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_418)   
[valor absoluto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_348)   
[datos abstractos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1282) , *ver también* abstracción de datos   
[modelos abstractos para datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1426)  
sintaxis abstracta   
    [en evaluador metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4236)   
    [en intérprete de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5296)  
abstracción, *ver también* medios de abstracción; abstracción de datos; procedimientos de orden superior   
    [patrón común y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_980) [procedimental](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_576)   
    [metalingüístico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4196) [en diseño de máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5546) [de búsqueda en programación no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4924) [barreras de abstracción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1264) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1380) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2286) [en sistema de números complejos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2304) [en sistema aritmético genérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2498) [secuencia acelerada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3976) [acumular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1018) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1742) [igual que fold-right](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1810) [acumular-n](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1794) [acumulador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1736) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2902) [Ã chárya, Bháscara](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_748) [Función de Ackermann](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_708) [adquirir un mutex](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3642) [acciones, en máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5534) [valor real](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4722) [Ada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5168) [procedimientos recursivos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_684) [Adams, Norman I., IV](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4646) [add (genérico)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2504) [usado para coeficientes polinomiales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2710) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2714) [add-action!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3374) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3408) [add-binding-to-frame!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4476) [add-complex](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2312) [add-complex-to-schemenum](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2578) [add-interval](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1488) [add-lists](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4786) [add-poly](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2674) [add-rat](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1308) [add-rule-or-assertion!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5382) [add-streams](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3864) [add-terms](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2708) [add-to-agenda!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3420) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3472) [add-vect](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1938) [sumando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2074) sumador [completo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3364) [medio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3356) [acarreo de ondulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3390) [sumador (restricción primitiva)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3532) [aditividad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1278) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2290) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2426) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2534) [dirección aritmética](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5838) [de direcciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5834) [Adelman, Leonard](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_926) [adjoin-arg](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6062) [conjunto-contiguo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2106) [representación-de-árbol-binario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2162) [representación-de-lista-ordenada representación-de-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2132) [lista-desordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2118) [para conjuntos ponderados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2274) [término-contiguo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2694) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2734) [advance-pc](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5732) [after-delay](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3376) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3424) agenda, *ver* simulación-de-circuito-digital álgebra [de A'h-mose , simbólica,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_826) *ver* álgebra simbólica [expresión algebraica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2648) [diferenciación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2046) [representación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2058) [simplificación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2084) [especificación algebraica para datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1442) [estructura de bloque](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_628) de Algol [llamada-por-nombre paso de argumentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3806) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4716) [thunks](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3822) ,  
  
    [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4700)   
    [debilidad en el manejo de objetos compuestos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3568)  
algoritmo [probabilístico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_910)   
    [óptimo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1780) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2816) [aliasing](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2998) [all-regs (compilador)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6374) [Allen, John](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5960) [alternativa de if](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_438) [always-true](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5338) [amb](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4822) amb evaluador, *ver* evaluador no determinista [ambeval](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4994) [un-elemento-de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4830) [un-entero-a-partir-de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4832) [computadora analógica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4064) analizar [metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4656) [no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4992) analizar-... [metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4658) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4664) [no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5004) [analizar-amb](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5016) [analizar evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4636) [como base para el evaluador no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4980) [dejar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4662) [y (lenguaje de consulta)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5124) [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5198) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5318) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5428) [y (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_446) [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_450) [por qué una forma especial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_466) [sin subexpresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4396) [y-puerta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3346) [y-puerta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3382) ángulo [datos-direccionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2470) [representación polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2350) [representación rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2334) [con datos etiquetados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2412) [ángulo-polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2396) [ángulo-rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2384) [anunciar-salida](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4536) [APL](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1764) [Appel, Andrew W.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6366) [append](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1632) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1636) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3164) [como acumulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1768) [append! vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3162) [con número arbitrario de argumentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6396) [como registro máquina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5926) [``qué es'' (reglas) vs. ``cómo'' (procedimiento)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5074) [append!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3166) [como máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5928) [append-instruction-sequences](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6260) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6390) [append-to-form (rules)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5158) [aplicación?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4340) [orden aplicativo evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_342) [en Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_344) [orden normal vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_482) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_854) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4682) [apply (lazy)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4724) [apply (metacircular)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4250) [primitivo apply vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4578) [apply (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2458) [apply-dispatch](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6072) [modificado para código compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6474) [apply-generic](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2462) [con coerción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2590) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2626) [con coerción mediante generación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2636) [con coerción de múltiples argumentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2632) [con coerción para simplificar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2638) [con paso de mensajes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2490) [con torre de tipos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2610) [apply-primitive-procedure](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4248) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4440) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4514) [apply-rules](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5358) [árbitro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3684) [arctangent](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2322) [argl registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6028) paso de argumentos, *ver* paso de argumentos de llamada por nombre; paso de argumentos de llamada por necesidad argumento [(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_188) [número arbitrario de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_194)  
  
    , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1652) [*De caelo* de Aristóteles](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3688)   
    [retrasado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4070) [(comentario de Buridan sobre)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3688) [dirección](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5840) aritmética aritmética [genérica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2496) , *véase también* operaciones aritméticas genéricas [sobre números complejos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2302) [sobre intervalos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1480)     sobre polinomios, *véase* aritmética de polinomios [sobre series de potencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3930) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3936) [sobre números racionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1294) [procedimientos primitivos para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_160) [artículos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4960) [código ASCII](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2214) [ensamblar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5686) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5694) [ensamblador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5636) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5682) [¡aserción! (intérprete de consultas)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5240) [aserción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5104) [implícito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5148) [asignar (en máquina de registro)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5518) [simulando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5720) [almacenar etiqueta en registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5556) [asignar-nombre-reg](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5724) [asignar-valor-exp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5726) [asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2836) , *véase también* ¡conjunto! [beneficios de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2916) [los errores asociados con](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3004) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3022) [costos del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2976) [operador de asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2842) , *véase también* ¡conjunto! [valor-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4296) [asignación variable-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4294) [asignación asignación?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4292) [assoc](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3282) [atan (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2324) [operaciones atómicas soportadas en hardware](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3682) [requisito atómico para prueba-y-conjunto!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3672) [adjuntar-etiqueta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2368) [usando tipos de datos Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2558) [augend](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2076) [automágicamente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4846) [búsqueda automática](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4808) , *véase también* [historial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4860) de búsqueda de [asignación automática de almacenamiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5832) [promedio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_516) [amortiguamiento promedio amortiguamiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1128) [promedio promediador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1160) [( restricción)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3560)  
  
[B-tree](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2172)   
[backquote](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6282)   
[backtracking](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4852) , *véase también* computación no determinista   
[Backus, John](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4166)   
[Baker, Henry G., Jr.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5948)   
[árbol binario equilibrado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2164) , *véase también* árbol binario [cuenta bancaria](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2848)   
[móvil equilibrada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1702) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3124) [intercambio de saldos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3626) [conjunto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2996) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3026) [conjunto, modelado con flujos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4176) [conjunto, con acceso concurrente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3588) [modelo de flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4148) [serializado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3614) [protegido por contraseña](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2912) [transferencia de dinero](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3636) [barrera de sincronización](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3710) [Barth, John](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4188) [Restricciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1546) básicas en datos compuestos [debilidad en el manejo de objetos compuestos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3570) [Batali, John Dean](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6004) [begin (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2874) [implícito en consecuente de cond y en el cuerpo del procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2878) [begin-actions](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4328) [begin?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4326) [abajo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1880) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1976) [Hipótesis de Bertrand](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3880) [al lado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1878) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1972) [bignum](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5880) números binarios, adición de, *ver* sumador [búsqueda binaria](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2144) [árbol binario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2142) [balanceado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2166) [conversión de una lista a una](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2188) [conversión a una lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2182) [para codificación de Huffman](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2240) [representada con listas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2150) [conjuntos representados como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2138) [tabla estructurada como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3310) [enlace](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_588) [enlace](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3042) [coeficientes binomiales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_740) [profundos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4490) [caja negra](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_572) [estructura de bloque](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_612) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4600) [en modelo de entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3112) [en lenguaje de consulta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5448) [proceso bloqueado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3664) [cuerpo de un procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_324) [Bolt Beranek y Newman Inc.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_62) [Borning, Alan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3492) [Borodin, Alan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1788) [variable enlazada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_584) [notación de caja y puntero](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1528) [marcador de fin de lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1560) [rama (en máquina de registro)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5506) [simulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5740) [de rama de un árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_258) [punto de interrupción de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5826) [destino de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5744) [rama error](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_18) [de corazón roto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5980) [captura de una variable libre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_606) [orden de asignaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3020) [efecto secundario con aliasing](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3002) [burocracia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5254) [Buridan, Jean](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3686) [ocupado esperando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3666)

C   
    [compilación de Scheme en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6520)   
    [manejo de errores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6186) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6496)   
    [procedimientos recursivos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_688)   
    [restricciones en datos compuestos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1550)   
    [Intérprete de Scheme escrito en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6518) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6522)   
[ca ... r](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1588)  
[protocolos de coherencia de caché](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3594) [calculadora](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1108)   
[cadr](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1586) [, puntos fijos con paso de argumentos de llamada por nombre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3808) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4714) [paso de argumentos de llamada por necesidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3816) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4712) [memorización y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3902) [llamada a cada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3398) [cáncer de la](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_286) [forma canónica de punto y coma, para polinomios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2770) [que capturan una variable libre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_604) [car (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1324) [axioma para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1454) [implementado con vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5896) [como operación de lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1580) [origen del nombre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1334) [implementación procedimental de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1462) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1470) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3206) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3212) , [[5]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4776) [números de Carmichael](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_912) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_948) análisis de casos [programación dirigida por datos vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4246) [general](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_350) , *véase también* cond [con dos casos ( si )](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_426) [cd ... r](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1590) [cdr (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1328) [axioma para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1456) [implementado con vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5898) [como operación de lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1582) [origen del nombre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1336) [implementación procedimental de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1464) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1472) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3208) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3214) , [[5]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4778) [cdr hacia abajo en una](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1606) [celda de lista, en serializador implementación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3662) [convertidor de celsius-fahrenheit](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3518) [expresión orientada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3566) [centro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1518) [Cesí ro, Ernesto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2942) [cesaro-stream](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4128) [cesaro-test](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2950) [Chaitin, Gregory](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2932) [Chandah-sutra](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_810) cambio e igualdad [significado de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2988) [datos compartidos y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3182) cambio de dinero, *ver* conteo cambio [caos en el sistema solar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_104) [Chapman, David](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4904) cadenas de caracteres [procedimientos primitivos para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5410) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6320) [cita de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2006) [carácter, codificación ASCII](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2218) [Charniak, Eugene](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4872) [Chebyshev, Pafnutii L'vovich](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3878) [ajedrez, rompecabezas de ocho reinas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1848) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4944) [implementación de chip de Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6002) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6014) [retroceso cronológico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4858) [Chu Shih-chieh](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_750) [Church numerales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1474) [Church, Alonzo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1056) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1476) [tesis de Church-Turing](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4562) circuito     digital, *ver* simulación de circuito digital [modelada con flujos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4030) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4088)      
[Clark, Keith L.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5274)   
[cláusula, de una](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_362)   
    [sintaxis adicional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4402) [cond](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_362)  
[Clinger, William](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4378) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4718)   
[suposición de mundo cerrado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5272)   
[clausura](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1266)   
    [en álgebra abstracta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1536)   
    [propiedad de clausura de cons](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1532)  
     [propiedad de clausura de operaciones de lenguaje de imágenes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1860) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1886)   
    [falta de en muchos idiomas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1542)   
[carbón,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1872)  
código bituminoso   
    [ASCII](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2216)   
    [longitud fija](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2222)  
    Huffman, *ver* código Huffman [prefijo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2236)   
    [Morse código](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2230) [de longitud variable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2226) [generador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6242) [argumentos de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6246) [valor de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6258) [coeff](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2706) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2748) [coerción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2580) [en manipulación algebraica en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2774) [procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2582) [aritmético polinomial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2718) [tabla](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2588) [Colmerauer, Alain](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5054) [combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_156) [combinación como operador de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1162) [expresión compuesta como operador de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_474) [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_244) [lambda expresión como operador de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1048) [como operador de combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1164) [como un árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_252) [combinación, medios de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_124) , *ver también* clausura [coma, usada con comentarios de comillas invertidas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6290) [en programas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1836) [Tratamiento de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1600) [Common Lisp de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_92) [nil](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1600) [compactación recolector de basura](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5962) compilación, *ver* compilador [compilar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6244) [compilar-y-go](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6472) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6484) [compilar-y-ejecutar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6514) [compilar-aplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6346) [compilar-asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6312) [compilar-definición](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6314) [compilar-si](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6324) [compilar-lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6340) [compilar-enlace](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6292) [compilar-aplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6376) [llamada a procedimiento de compilación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6350) [compilación entre comillas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6304) [compilación autoevaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6302) [secuencia de compilación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6328) [entorno de tiempo de compilación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6440) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6450) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6452) [codificación abierta y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6464) [variable de compilación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6306) [aplicación compilada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6476) [entrada de procedimiento compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6336) [entorno de procedimiento compilado ¿](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6338) [procedimiento compilado?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6334) [intérprete de compilador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6194) [vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6210) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6492) [recursión de cola, asignación de pila y recolección de basura](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6356) [compilador para Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6224) , *véase también* generador de código; entorno de tiempo de compilación; secuencia de instrucciones; descriptor de enlace; registro de destino [análisis evaluador vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6234) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6238) [asignaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6308)     generadores de código, *véase* compilar-... [combinaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6344) [condicionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6316) [definiciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6310) [eficiencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6232) [ejemplo compilación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6404) [control explícito evaluador vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6226) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6274) ,  
  
    [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6488)   
    [procedimientos de sintaxis de expresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6240)   
    [que interactúan con el evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6466)   
    [generación de etiquetas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6318) [expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6330)   
    [lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6330)  
     [direccionamiento léxico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6428)   
    [código de enlace](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6278)   
    [uso de operaciones de máquina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6222)   
    [monitoreo del rendimiento (uso de pila) del código compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6486) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6506) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6510)   
    [codificación abierta de primitivas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6422) , [[2] orden de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6458) [procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6342)   
    [de evaluación de operandos aplicaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6416) [citas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6298) [uso de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6220) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6228) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6372) [ejecución de código compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6468) [escaneo de definiciones internas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6448) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6456) [expresiones autoevaluadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6296) [secuencias de expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6326) [uso de pila](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6262) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6270) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6418) [estructura del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6236) [código recursivo de cola generado por](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6352) [variables](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6300) [paquete complejo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2538) números complejos [representación polar representación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2340) [rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2320) [forma rectangular vs. polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2306) [representada como datos etiquetados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2356) [aritmética de números complejos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2300) [compleja->compleja](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2630) [interconectada con el sistema aritmético genérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2540) [estructura del sistema](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2418) [composición de funciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1226) [datos compuestos, necesidad de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1256) [expresión compuesta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_154) , *ver también* combinación; forma especial [como operador de combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_476) [procedimiento compuesto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_296) , *ver también* procedimiento [usado como procedimiento primitivo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_332) [procesamiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5196) [de consultas compuestas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5122) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5316) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5422) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5430) , [[5]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5436) [aplicación compuesta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6080) [procedimiento compuesto?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4446) [computabilidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4554) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4596) [proceso computacional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_8) , *ver también* proceso [informática](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4200) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4558) [matemáticas vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_498) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5036) [representación concreta de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1286) [concurrencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3578) [corrección de programas concurrentes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3600) [interbloqueo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3694) [programación funcional y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4162) [mecanismos para controlar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3608) [cond (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_352) [cláusula adicional sintaxis](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4400) [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_384) [cláusula de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_364) [if vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_442) [implícito comenzar en consecuente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2880) [cond->if](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4370) [acciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4368) [cond cláusulas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4362) [cond cláusula cond-else?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4364) [predicado cond](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4366) [cond?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4360) expresión condicional [cond](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_356) [if](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_424) [congruente módulo *n*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_888) [unir](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5320) [conectar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3530) ,  
  
[[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3558)   
[conector(es), en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3516)   
    [operaciones del sistema de restricción en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3520)   
    [la representación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3544)   
[de Conniver](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4882)   
[cons (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1320)   
    [axioma para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1452)   
    [propiedad de cierre de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1534)   
    [implementado con mutadores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3158)   
    [implementado con vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5904)   
    [como lista operación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1592)   
    [significado del nombre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1332)   
    [implementación procedimental de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1460) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1468) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3160) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3204) , [[5]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3210) , [[6]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4774)   
[cons una lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1628)   
[cons-stream (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3738) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3778)   
    [evaluación perezosa y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4770)   
    [por qué una forma especial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3784)   
[conciencia, expansión de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4262)  
consecuente   
    [de cláusula cond](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_388)  
     [de if](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_436)   
[const (en máquina de registros)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5530)   
    [simulando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5772)   
    [sintaxis de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5618)   
[constante (restricción primitiva)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3540)   
[constante, especificando en máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5620)   
[exp-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5780)   
[constante valor-exp-constante](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5782)  
[restricción red](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3514)  
restricción(es) propagación   
    [primitiva](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3512)   
    [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3484)   
[constructor-arglist](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6348)  
[constructor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1292)   
    [como barrera de abstracción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1382)   
[contenidos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2372)  
     [usando Scheme tipos de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2562)  
continuación   
    [en evaluador no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4984) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5000) , *ver también* falla continuación; éxito continuación   
    [en simulador de máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5700)   
[continuar registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5554)   
    [en evaluador de control explícito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6024)   
    [recursión y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5582)   
[fracción continua](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1138)   
    [*e* como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1146)   
    [proporción áurea como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1140)   
    [tangente como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1152)   
[estructura de control](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5252)   
[controlador para máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5470)   
    [diagrama de controlador secuencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5488)   
[de interfaz convencional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1268)   
    [como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1720)   
[Cormen, Thomas H.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2176)   
[división de esquina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1892)  
[corrección de un programa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_500)   
[cos (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1112)  
coseno   
    [punto fijo de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1104)   
    [serie de potencia para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3916)   
[radiación cósmica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_914)   
[conteo-cambio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_730)   
[conteo-hojas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1682) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1690)   
    [como acumulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1792)   
    [como máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5924)   
[conteo-pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3192)  
[conteo cambio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_728) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1646)   
[cuentas de tarjetas de crédito, internacionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3718)   
[Cressey, David](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5982)   
[operaciones de tipo cruzado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2574)   
[criptografía](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_920)   
[cubo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_788) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_962) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1194)  
raíz cúbica   
    [como punto fijo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1172)   
    [por el método de Newton](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_560)   
[raíz cúbica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1176)  
[tiempo actual, para simulación agenda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3450)   
[tiempo actual](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3422) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3458)   
[ciclo en lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3170)  
    [detector](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3194)

[Darlington, John](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4170)   
[datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_12) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_130)   
    [resumen](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1284) , *ver también* abstracción de datos   
    [modelos abstractos para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1424)   
    [especificación algebraica para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1440) [representación concreta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1288)   
    [compuesta de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1258) [jerárquica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1540) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1672) [significado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1414) [estructurado en lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1344) de     mutable, *ver* objetos de datos mutables representación procedimental [numérica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_134) [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1446) [como programa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4542) [compartido](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3176) [simbólico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1984) [etiquetado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2360) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5866) [abstracción de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1262) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1280) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2288) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2310) , [[5]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4280) , *ver también* evaluador metacircular [para cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3228) base de datos [programación dirigida por datos e](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2482) [indexación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5190) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5374) [Insatiable Enterprises](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2484) [programación lógica de personal y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5098) [Microshaft personal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5106) [como conjunto de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2202) [rutas de datos para máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5468) [diagrama de ruta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5478) de datos tipos de datos [en Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2552) [en lenguajes fuertemente tipados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4098) [programación dirigida por datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2298) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2424) [análisis de caso vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4244) [en evaluador metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4388) [en intérprete de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5308) [recursión dirigida por datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2720) [evitación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3698) [de punto muerto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3696) [recuperación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3700) [depuración](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_22) [punto decimal en números](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_542) [declarativo vs. imperativo conocimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_492) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5030) [programación lógica y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5070) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5256) [computación no determinista y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4816) [decodificación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2272) [descomposición del programa en partes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_568) [enlace profundo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4488) [reverso profundo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1692) [operaciones diferidas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_652) [define (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_224) [con notación de cola punteada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1650) [modelo de entorno de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3078) [lambda vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1042) [para procedimientos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_306) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1046) [valor de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_228) [azúcar sintáctico de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4300) [por qué una forma especial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_270) define (forma especial)     interna, *ver* definición interna [¡define-variable!,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4460) [ [2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4484) [integral definida](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1002) [estimada con simulación de Monte Carlo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2960) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4142) definición, *ver* define ; definición interna [valor-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4306) [definición variable-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4304) [definición ¿definición?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4302) [deKleer, Johan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4898) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5264) [retraso (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3768) [explícita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4072) [explícita vs. implementación automática](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4798) [usando lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3796)  
  
    [evaluación perezosa y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4768)   
    [memorizada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3800) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3900)   
    [por qué una forma especial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3786)   
[de retraso, en el circuito digital](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3342)   
[retraso-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4742)  
[argumento retrasado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4068)   
[evaluación retrasada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-19.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2834) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3730)   
    [asignación y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3830)   
    [explícita vs. automática](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4800)   
    [en el evaluador perezoso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4668)   
    [evaluación de orden normal e](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4094)   
    [impresión y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3826)   
    [secuencias y cola de eliminación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4056)   
[de objetos retrasada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3772)   
[.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3240) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3258)   
[denom](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1304) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1352)   
    [axioma para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1420)   
    [reducir a los términos más bajos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1394)   
[polinomio denso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2726)   
[retroceso dirigido por dependencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4888)   
[deposit , con serializador externo mensaje](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3632)   
[de depósito para cuenta bancaria](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2898)   
[búsqueda en profundidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4854)   
[deque](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3266)   
[deriv (numérico)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1192)   
[deriv (simbólico) derivada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2056)   
    [dirigida por datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2480)   
[de una función](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1186)   
[expresiones derivadas en el evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4356)   
    [suma al evaluador de control explícito diseño,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6116) [ecuación diferencial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4062)   
[estratificada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1980) , *ver también* resolver [segundo orden](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4082) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4084) diferenciación reglas [numéricas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1190) [para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2048) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2094) [simbólico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2042) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2478) [difusión, simulación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3606) [señal digital](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3338) [simulación de circuito digital](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3328) [agenda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3410) [implementación de agenda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3436) [función primitiva cajas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3368) [que representan cables](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3392) [simulación de muestra](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3428) [Dijkstra, Edsger Wybe](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3658) [Dinesman, Howard P.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4932) [*Aritmética* de Diofanto , copia de Fermat de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_882) [disjoin](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5324) despacho [comparando diferentes estilos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2494) [en el tipo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2430) , *ver también* programación dirigida por datos [display (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_940) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1358) [línea de visualización](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3766) [flujo de visualización](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3764) [¿distinto?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4934) [div (genérico)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2510) [div-complejo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2318) [div-intervalo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1500) [división por cero](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1512) [div-poli](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2760) [div-rata](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1314) [div-serie](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3938) [div-términos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2762) [¿divide?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_864) [¿divisible?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3838) [división de números enteros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_532) [perro, perfectamente racional, comportamiento de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3690) [DOS/Windows](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6190) [producto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1802) -punto notación de cola punteada [para parámetros de procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1648) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2456) [en patrón de consulta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5116) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5350) [en regla de lenguaje de consulta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5162) [lectura y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5354) [Doyle, Jon controlador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4890) [de línea de dibujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1948) bucle [en evaluador de control explícito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6136) [en evaluador perezoso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4732) [en evaluador metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4518)  
  
    [en evaluador no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4908) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5018)   
    [en intérprete de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5232) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5288)   
bucle de controlador   
    [para evaluador perezoso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4736)   
    [para evaluador metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4532)   
    [para evaluador no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5024)

*e*   
    [como fracción continua](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1148)   
    [como solución a la ecuación diferencial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4078)   
[*e x* , serie de potencias para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3914)   
[la Tierra, medición de la circunferencia de la eficiencia del marco](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3848)   
[edge1 marco](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1918)   
[edge2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1920)  
, *véase también* orden de crecimiento, *véase también* orden de crecimiento   
    [de la compilación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6230)   
    [del acceso a la base de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5188)   
    [de la evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4642)   
    [de Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_100)   
    [del procesamiento de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5202)   
    [del proceso recursivo de árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_732)   
[EIEIO](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3716)   
[rompecabezas de ocho reinas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1846) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4946)   
[circuitos eléctricos, modelados con flujos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4032) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4090)   
[¿elemento del conjunto?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2104)  
     [representación de árbol binario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2160)   
    [representación de lista ordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2128)   
    [representación de lista desordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2116)   
[else (símbolo especial en cond )](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_416)   
[lenguaje integrado, diseño de lenguaje usando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4672)   
[lista vacía](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1594)   
    [denotada como '()](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2018)  
     [reconociendo con nulo?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1618)  
[flujo vacío](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3744)   
[agenda vacía?,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3414) [ [2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3470)   
[lista de argumentos vacía](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6060)   
[secuencia de instrucciones vacía](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6268)   
[cola vacía?,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3234) [ [2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3250)   
[lista de términos vacía?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2696) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2742)   
[nombre encapsulado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2886)   
[entorno de inclusión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3044)   
[entorno de inclusión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4466)   
[codificar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2278)  
[marcador de fin de lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1564)   
[segmento final](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1402) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1958) [ingeniería](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_916)   
[de fin de enlace vs. matemáticas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6294) [entrada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2152) [intervalo de enumeración](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1744) [árbol de enumeración](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1750) [enumerador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1730) [entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_240) [de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6020) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3038)     tiempo de compilación, *ver* entorno de tiempo de compilación [como contexto para la evaluación global](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_268) [de inclusión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3046)     , *ver* entorno global [alcance léxico y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_622) [en el intérprete de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5450) [cambio de nombre vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5444) [modelo de entorno de evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-19.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2832) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3034) [estructura del entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3058) [definiciones internas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3114) [estado local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3102) [paso de mensajes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3120) [evaluador metacircular y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4216) [aplicación de procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3082) [reglas de ejemplo para la evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3064) [recursión de cola y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3094) [eq? (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2026) [para objetos arbitrarios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3188) [como igualdad de punteros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3186) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5868) [implementación para símbolos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5884) [igualdad numérica y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5874) [equ? (predicado genérico)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2564) [equal-rat?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1316) [equal?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2032) ​​igualdad [en sistema aritmético genérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2566) [de listas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2034) [de números](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_408) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2040) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5876)  
  
    [transparencia referencial y ecuación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2994)   
    [de símbolos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2022)  
, solución, *véase* método del semiintervalo; método de Newton; resolver [el error](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1088)  
[de Eratóstenes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3846) [(procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1088) manejo de errores [en código compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6494) [en evaluador de control explícito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6148) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6182) [Escher, Maurits Cornelis](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1858) [estimar integral](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2962) [estimar pi](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2948) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2954) [Algoritmo de Euclides](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_836) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5476) , *ver también* máximo común divisor [orden de crecimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_844) [para polinomios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2786) [*Elementos* de Euclides](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_838) [prueba de Euclides de un número infinito de primos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3876) [anillo euclidiano](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2790) [Euler, Leonhard](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1144) [prueba del pequeño teorema de Fermat](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_878) [acelerador de series](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3968) [transformada de Euler](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3970) [ev-aplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6056) [ev-asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6110) [ev-comienzo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6084) [ev-definición](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6114) [ev-si](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6106) [ev-lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6044) [ev-citado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6042) [ev-autoevaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6038) ev-secuencia [con recursión de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6086) [sin recursión de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6100) [ev-variable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6040) [eval (perezoso)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4720) [eval (metacircular)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4230) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4242) [analizar versión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4654) [primitivo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4576) [dirigido a datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4392) [eval vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4576) [eval (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4572) [Esquema MIT](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4584) [utilizado en el intérprete de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5334) [eval-asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4266) [eval-definition](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4268) [eval-dispatch](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6034) [eval-if (lazy)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4730) [eval-if (metacircular)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4258) [eval-sequence](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4264) evaluación     orden aplicativo, *ver* orden aplicativo evaluación     retrasada, *ver* evaluación retrasada     modelo de entorno de, *ver* modelo de entorno de modelos de evaluación [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6132)     orden normal, *ver* orden normal evaluación [de una combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_246) [de y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_452) [de cond](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_386) [de if](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_430) [de o](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_460) [de expresiones primitivas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_266) [de formas especiales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_274)     orden de subexpresión evaluación, *ver* orden de evaluación     modelo de sustitución de, *ver* modelo de sustitución de procedimiento [evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4198) de aplicación , *ver también* intérprete [como máquina abstracta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4548) [metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4214) [como máquina universal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4550) evaluadores, *ver*  
  
evaluador metacircular; evaluador analizador; evaluador perezoso; evaluador no determinista; intérprete de consultas; evaluador de control explícito   
[even-fibs](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1724) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1754)   
[even?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_804)  
estrella vespertina, *ver* Venus   
[simulación impulsada por eventos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3330)   
[evlis recursión de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6066) [intercambio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3628)  
[de enteros exactos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_528) [signo de exclamación en nombres](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2870) [ejecutar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5670) ejecución-aplicación [metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4660) procedimiento de ejecución [no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5012) [en el análisis evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4652) [en evaluador no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4982) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4986) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5002) [en simulador de máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5668) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5716) [exp registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6018) [cláusulas de expansión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4372) [control-explícito evaluador para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5996) [asignaciones de Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6108) [combinaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6048) [procedimientos compuestos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6078) [condicionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6104) [controlador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6032) [rutas de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6006) [definiciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6112) [expresiones derivadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6118) [bucle de controlador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6134) [manejo de errores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6150) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6184) [expresiones sin subexpresiones para evaluar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6036) [como programa en lenguaje](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6208) [máquina modelo de máquina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6158) [modificado para código compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6470) [monitoreo de rendimiento (uso de pila)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6160) [evaluación de orden normal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6124) [operaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6008) [de evaluación de operandos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6058) [optimizaciones (adicionales)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6276) [procedimientos primitivos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6074) [procedimiento aplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6046) [registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6016) [ejecución](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6128) [de secuencias de expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6082) [formas especiales (adicionales)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6120) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6122) [uso de pila](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6050) [recursión de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6088) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6166) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6174) [como máquina universal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6198) [expmod](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_898) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_944) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_946) [crecimiento exponencial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_782) [de árbol-recursivo cálculo de número de Fibonacci](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_722) [exponenciación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_790) [módulo *n*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_896) expresión, *ver también* expresión compuesta; expresión primitiva     algebraica, *ver* expresiones algebraicas [simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1272) [autoevaluativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4238) , *ver también* símbolo(s) [estilo de programación orientado a expresiones vs. imperativo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3574) expt [versión iterativa lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_794) [versión recursiva lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_792) [máquina de registro para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5590) [entorno extendido](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4458) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4478) [extender-si-consistente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5348) [extender-si-posible](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5370) [entrada-externa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6482) [extraer-etiquetas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5688) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5692)  
  
[#f](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_380)  
[factorial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_640) , *ver también* factorial   
    [flujo infinito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3890)   
    [con letrec](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4622)  
     [sin letrec o definir](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4626)  
factorial   
    [como una](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4546)   
    [compilación de máquina abstracta de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6406) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6412)   
    [estructura del entorno en la evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3088)   
    [versión iterativa lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_646)   
    [versión recursiva lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_642)   
    [máquina de registro para (iterativa)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5490) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5532)   
    [máquina de registro para (recursiva)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5572) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5586)   
    [uso de pila, compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6508)   
    [uso de pila, interpretado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6170) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6172)   
    [uso de pila, máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5814)   
    [con asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3018)   
    [con procedimientos de orden superior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1012)   
[falla continuación (evaluador no determinista)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4990) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4998)   
    [construido por amb](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5014)  
     [construido por asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5008)   
    [construido por controlador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5020)   
[falla de bucle, en cálculo no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4826)   
    [error vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5006)   
    [búsqueda y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4840)   
[falso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_370)   
[falso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_382)   
[falso?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4436)   
[fast-expt](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_798)   
[fast-prime?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_906)  
[bucle de retroalimentación, modelado con flujos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4058)   
[Feeley, Marc](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4648)   
[Feigenbaum, Edward](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5086)   
[Fenichel, Robert](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5942)   
[Fermat, Pierre de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_872)   
[Fermat prueba para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_868)   
    [la variante de primalidad del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_952)   
[Pequeño Teorema de Fermat](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_886) [prueba](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_876)   
    [de forma alternativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_960) [prueba de fermat obtener](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_904) [aserciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5378) [obtener reglas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5380) fib [versión iterativa lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_726) [versión logarítmica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_828) [máquina de registro para (árbol recursivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5584) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5588) [uso de pila, compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6512) [uso de pila,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6178) [versión recursiva de árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_720) interpretada , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6180) [con memorización](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3322) [con nombrado let](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4422) [números de Fibonacci](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_718) , *ver también* fib [algoritmo MCD de Euclides y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_846)     flujo infinito de , *ver* fibs [fibs (flujo infinito)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3840) [definición implícita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3868) [FIFO buffer](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3226) [filter](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1026) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1732) [filter](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1740) [filtrado-acumular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1024) [buscar aserciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5344) [buscar-divisor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_862) [primer-elemento-de-agenda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3416) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3480) [elementos de primera clase en el lenguaje](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1218) [primera-exp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4332) [primer-marco](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4468) [primer-operando primer](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4348) [-segmento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3466) [primer-término](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2698) , [[2] computación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2738) [de punto fijo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1096) [con calculadora](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1106) [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1102) [raíz cúbica de coseno como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1174) [cuarta raíz como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1240) [proporción áurea como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1132) [como Mejora iterativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1252) [en el método de Newton](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1180) [*n* -ésima raíz como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1244)  
  
    [raíz cuadrada como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1120) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1168) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1208)   
    [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1204)   
    [unificación de funciones transformadas y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5366)   
[código de longitud fija](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2220)   
[punto fijo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1100)  
     [como mejora iterativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1250)   
[punto fijo de transformación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1206) [registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5660)   
[de bandera](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5660)   
[mapa plano](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1826)   
[flatten-stream](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5392)   
[flip-horiz](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1884) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1974)   
[flip-vert](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1882) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1964)   
[pares invertidos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1888) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1902) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1904)   
[Floyd, Robert](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4862)   
[fold-left](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1814)   
[fold-right](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1812)   
[for-each](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1670) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4750)   
[for-each-except](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3548)  
[Forbus, Kenneth D.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4896)   
[force](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3774) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3798)   
    [forzando un thunk vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4708)   
[force a thunk](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4704)   
[force-it](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4740)  
     [versión memorizada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4748)   
[¡olvidar valor!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3528) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3556)   
[parámetros formales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_320)   
    [nombres del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_580)   
    [alcance de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_598)   
[las expresiones de entrada de formato](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_200)   
[Fortran](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_40) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1762)   
    [inventor de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4168)   
    [restricciones en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1544)   
[el reenvío de datos compuestos dirección](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5988)   
[raíz cuarta, como punto fijo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1238)  
fracción, *ver* número(s) racional(es)   
[marco (modelo de entorno)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3040)   
    [como repositorio de estado local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3098) [marco](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1864)   
    [global](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3050) (lenguaje de imágenes) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1910) [mapa de coordenadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1924) [marco (intérprete de consultas)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5180) , *ver también* coincidencia de patrones; [representación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5416) de unificación [mapa de coordenadas de marco](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1926) [valores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4474) [de marco variables](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4472) [de marco pila enmarcada disciplina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6052) [Franz Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_74) [registro libre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5906) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5976) [lista libre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5908) [captura](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_608) [de variable libre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_590) [en definición interna](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_618) [Friedman, Daniel P.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3812) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4204) [franja](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1696) [como un árbol enumeración](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1746) [front-ptr](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3242) [cola frontal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3236) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3254) [sumador completo función](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3362) [sumadora completa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3366) (matemática) [notación para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1124) [la composición](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1228) [de Ackermann de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_710) [la derivada del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1188) [punto fijo del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1098) [procedimiento frente a](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_490) [la aplicación repetida](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1230) [racional del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2778) [suavizado de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1232) [la función box, en circuitos digitales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3340) [programación funcional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2978) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4118) [concurrencia y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4164) [lenguajes de programación funcional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4160) [tiempo y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4146)

[Gabriel, Richard P.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4632)  
[garbage collection](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5930)  
    [memoization and](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4744)  
    [mutation and](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3156)  
    [tail recursion and](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6360)  
garbage collector   
    [compacting](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5964)  
    [mark-sweep](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5958)  
    [stop-and-copy](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5934)  
GCD, *see* greatest common divisor   
[gcd](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_842)  
    [register machine for](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5474), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5634)  
[gcd-terms](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2794)  
[general-purpose computer, as universal machine](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6202)  
[generate-huffman-tree](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2280)  
[generating sentences](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4976)  
[generic arithmetic operations](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2502)  
    [structure of system](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2500)  
[generic operation](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1274)  
[generic procedure](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2268), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2292)  
    [generic selector](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2404), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2422)  
[Genesis](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5170)  
[get](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2440), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3300)  
[get-contents](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5644)  
[get-global-environment](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6140)  
[get-register](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5680)  
[get-register-contents](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5630), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5676)  
[get-signal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3370), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3404)  
[get-value](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3524), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3552)  
[glitch](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_20)  
[global environment](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_242), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3062)  
    [in metacircular evaluator](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4498)  
[global frame](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3048)  
[Goguen, Joseph](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1430)  
[golden ratio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_724)  
    [as continued fraction](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1142)  
    [as fixed point](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1130)  
[Gordon, Michael](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4108)  
[goto (in register machine)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5514)  
    [label as destination](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5558)  
    [simulating](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5746)  
[goto-dest](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5750)  
[grammar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4962)  
graphics, *see* picture language   
[Gray, Jim](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3704)  
[greatest common divisor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_834), *see also* gcd   
    [generic](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2798)  
    [of polynomials](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2782)  
    [used to estimate [](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2946)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2946)  
    [used in rational-number arithmetic](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1376)  
[Green, Cordell](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5050)  
[Griss, Martin Lewis](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_72)  
[Guttag, John Vogel](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1438)

[half-adder](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3354)  
    [half-adder](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3360)  
    [simulation of](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3430)  
[half-interval method](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1082)  
    [half-interval-method](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1092)  
    [Newton's method vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1184)  
[halting problem](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4588)  
[Halting Theorem](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4592)  
[Hamming, Richard Wesley](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2248), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3894)  
[Hanson, Christopher P.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4382), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6364)  
[Hardy, Godfrey Harold](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3882), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4012)  
[has-value?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3522), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3550)  
[Hassle](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4690)  
[Havender, J.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3702)  
[Haynes, Christopher T.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4208)  
[headed list](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3276), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3452)  
[Hearn, Anthony C.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_70)  
[Henderson, Peter](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1856), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3858), [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4172)  
    [Henderson diagram](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3856)  
[Heraclitus](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-19.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2820)  
[Heron of Alexandria](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_512)  
[Hewitt, Carl Eddie](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_698), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4864), [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5052), [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5950)  
[hiding principle](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2890)  
[hierarchical data structures](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1538), [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1674)  
[hierarchy of types](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2594)  
    [in symbolic algebra](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2764)  
    [inadequacy of](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2614)  
[high-level language, machine language vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4190)  
[higher-order procedures](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_964)  
    [in metacircular evaluator](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4254)  
    [procedure as argument](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_966)  
    [procedure as general method](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1078)  
    [procedure as returned value](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1156)  
    [strong typing and](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4096)  
[Hilfinger, Paul](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2198)  
[Hoare, Charles Antony Richard](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1422)  
[Hodges, Andrew](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4566)  
[Hofstadter, Douglas R.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4570)  
[Horner, W. G.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1778)  
[Horner's rule](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1774)  
``how to'' vs. ``what is'' description, *see* imperative vs. declarative knowledge   
[Huffman code](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2212)  
    [optimality of](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2246)  
    [order of growth of encoding](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2284)  
[Huffman, David](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2238)  
[Hughes, R. J. M.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4792)

[IBM 704](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1338)   
[identidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_996)   
[si (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_420)   
    [cond vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_440)   
    [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_428)   
    [orden normal evaluación de predicado de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_486)   
    [un brazo (sin alternativa)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3474)   
    [, consecuente y alternativa de ¿](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_432)   
    [por qué una forma especial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_556)   
[si-alternativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4322)   
[si-consecuente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4320)   
[si-predicado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4318)   
[si?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4316)  
imag-part [representación polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2346)   
    [dirigida por datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2466) [representación rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2330) [con datos etiquetados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2408) [imag-part-polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2392) [imag-part-rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2380) [programación imperativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3014) [conocimiento imperativo vs. declarativo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_494) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5032) [programación lógica y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5072) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5258) [computación no determinista y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4818) [estilo de programación imperativa vs. orientada a expresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3576) dependencias de implementación, *ver también* valores no especificados [números](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_546) [orden de subexpresiones evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3068) [inc](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_992) [desarrollo incremental de programas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_234) [indeterminado de un polinomio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2654) [indexación de una base de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5192) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5376) [inferencia, método de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5248) [series infinitas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5368) [flujo(s) infinito(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3832) [fusión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3896) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3992) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4006) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4178) [fusión como una relación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4182) [de factoriales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3888)     de números de Fibonacci, *ver* fibs     de números enteros, *ver* enteros [de pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3986)     de números primos, *ver* primos [de números aleatorios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4126) [que representan series de potencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3912) [para modelar señales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4020) [para sumar una serie](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3960) [notación infija, notación prefija vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2096) [informar sobre la falta de valor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3536) [informar sobre el valor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3534) recuperación de información, *ver* base de datos [Ingerman, Peter](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4702) [operación de inicialización de pila](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5656) en máquina de registros , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5808) ¡insertar! [en una tabla unidimensional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3284) [en una tabla bidimensional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3292) [¡insertar cola!,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3238) [ [2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3256) [instalar paquete complejo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2542) [instalar paquete polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2454) [instalar paquete polinomial instalar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2684) [paquete racional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2530) [instalar paquete rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2448) [instalar paquete numérico de esquema](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2520) [instanciar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5304) [instanciar un patrón](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5120) [conteo de instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5816) [procedimiento de ejecución de instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5666) [secuencia de instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6256) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6378) [seguimiento de instrucciones procedimiento de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5818) [ejecución de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5708) [instrucciones texto de instrucción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5706) [entero(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_136) [división](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_534) [exacto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_530) [entero factor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2800) [enteros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3836)  
     
[(flujo infinito)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3836)   
    [definición implícita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3866)   
    [versión de lista perezosa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4790)   
[números enteros a partir de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3834)  
integral, *véase también* integral definida; integración de Monte Carlo   
    [de una](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3920)   
[integral](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1004) de serie de potencias , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4024) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4080)   
    [con argumento retrasado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4074)   
    [con lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1032)  
     [versión de lista diferida](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4802)   
    [necesidad de evaluación retrasada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4060)   
[integración en serie](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3924)  
[implementación de circuito integrado de Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6000) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6012)   
[integrador, para señales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4022)   
[interleave](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4000)   
[interleave-delayed](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5388) [definición interna](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_614)   
[de Interlisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_56) [en el modelo de entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3116) [variable libre en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_616) [let vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1074) [en evaluador no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5010) [posición de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_626) [restricciones en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4604) [el escaneo del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4608) [alcance del nombre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4598) [Internet ``Worm''](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6498) [símbolos de internación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5888) [intérprete](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_30) , *véase también* evaluador [compilador vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6212) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6490) [bucle de lectura-evaluación-impresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_206) [conjunto de intersección](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2110) [representación de árbol binario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2196) [representación de lista ordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2130) [representación de lista desordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2120) [aritmética de intervalo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1478) [invariante cantidad de un proceso iterativo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_816) [inversor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3344) construcciones de iteración del inversor, [véase](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3378) *construcciones* de bucle [mejora iterativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1246) [proceso iterativo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_666) [como un proceso de flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3944) [diseño de algoritmo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_818) [implementado por llamada a procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_552) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_694) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6102) , *véase también* recursión de cola [lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_672) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_772) [proceso recursivo vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_636) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3092) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5570) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6408) [máquina de registro para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5566)  
  
[Jayaraman, Sundaresan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3506)

[Kaldewaij, Anne](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_832)   
[Karr, Alphonse](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-19.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2822)   
[Kepler, Johannes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-30.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5456)   
[clave clave](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2208)  
de un registro   
    [en una base de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2206)   
    [en una tabla](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3270)   
    [que prueba la igualdad de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3306)   
[Khayyam, Omar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_746)   
[Knuth, Donald E.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_744) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_814) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_840) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1776) , [[5]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2924) , [[6]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2944) , [[7]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-38.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6794)   
[Kohlbecker, Eugene Edmund, Jr.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4376)   
[Kolmogorov, AN](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2928)   
[Konopasek, Milos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3504)   
[Kowalski, Robert](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5058)   
[KRC](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1822) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3998)

[etiqueta (en máquina de registros)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5510)   
    [simulando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5770)   
[etiqueta-exp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5784)   
[etiqueta-exp-etiqueta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5786)  
[fórmula de interpolación de Lagrange](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2662)   
[[](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1054)cálculo (cálculo lambda)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1054)   
[lambda (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1034)   
    [definir vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1040)   
    [con notación de cola punteada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1656) expresión  
lambda   
    [como operador de combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1050)   
    [valor del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3076)   
[cuerpo lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4312)   
[parámetros lambda ¿](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4310)   
[lambda?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4308)  
[Lambert, JH](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1150)   
[Lamé, Gabriel](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_850)   
[Teorema de Lamé](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_848)   
[Lamport, Leslie](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3722)   
[Lampson, Butler](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3006)   
[Landin, Peter](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_280) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3810)  
lenguaje, *ver* lenguaje natural; lenguaje de programación  
[Lapalme, Guy ¿](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4650)   
[última-exp?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4330)   
[¿último-operando? ¿](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6064)   
[último-par](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1638) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3168)   
    [reglas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5166)   
[evaluación perezosa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4684)   
[evaluador perezoso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4666)   
[lista perezosa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4756)   
[par perezoso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4760)   
[árbol perezoso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4794)   
[¿hoja?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2252)  
[mínimo compromiso, principio de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2364)   
[conferencia, algo que hacer durante](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1110)   
[la rama izquierda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2154) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2260)  
Leibniz, Baron Gottfried Wilhelm von  
     [prueba del pequeño teorema de Fermat](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_880)   
    [serie para[](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_976)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_976) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3952)   
[Leiserson, Charles E.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2178) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4016)   
[longitud](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1620)  
     [como acumulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1770)   
    [versión iterativa versión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1626)   
    [recursiva](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1624)   
[let (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1062)   
    [modelo de evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3110)   
    [definición interna vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1076) [alcance](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1070)   
    [nombrado de variables](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4414) [como azúcar sintáctico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1068) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3108) [let\* (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4408) [letrec (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4618) [direccionamiento léxico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6430) [dirección léxica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6434) [entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6432) [de alcance léxico estructura y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_620) [búsqueda de direcciones léxicas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6436) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6442) [¡conjunto de direcciones léxicas!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6438) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6444) [Lieberman, Henry](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5952) búfer LIFO, *véase* segmento de líneade pila [representado como un par de puntos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1396) [representado como un par de vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1952) [crecimiento lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_658) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_780) [proceso iterativo lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_674) [orden de crecimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_770) [proceso recursivo lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_662) [orden de crecimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_764) [descriptor de enlace](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6250) [Liskov, Barbara Huberman](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1444) Acrónimo de Lisp [para LISt Procesamiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_38) [evaluación de orden aplicativo en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_346) [DEC PDP-1](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5938) [eficiencia de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_98) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_214) [procedimientos de primera clase en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1222) [Fortran vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_96) [historia del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_24) [sistema de tipos interno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2550) [implementación original en IBM 704](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1340) [Pascal vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_290)  
  
    [idoneidad para escribir evaluadores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4202)   
    [características únicas de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_112)  
los dialectos Lisp   
    [Common Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_94)   
    [Franz Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_76)   
    [Interlisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_58)   
    [MacLisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_48)   
    [MDL](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5986)   
    [Portable Standard Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_68)   
    [Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_44)   
    [Zetalisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_82)   
[lisp-value (intérprete de consultas)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5332)   
[lisp-value (lenguaje de consultas)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5132) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5270)   
    [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5206) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5330) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5440)   
[lista (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1568)   
[estructura de lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1342)   
    [lista vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1572)   
    [mutable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3134)   
    [representado usando vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5854)   
[lista(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1566)   
    [backquote con](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6280)   
    [cdr ing down](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1608)   
    [combinando con append](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1634)   
    [cons ing up](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1630)   
    [conversión de un árbol binario a una](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2184)   
    [conversión a un árbol binario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2190)  
    vacío, *ver* lista vacía   
    [igualdad de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2036)   
    [header](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3278) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3454)   
    [último par de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1640) [longitud](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1622)   
    [perezosa de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4758) [estructura de lista vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1574) [manipulación con car , cdr y cons](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1578) [mapeo sobre el](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1658) [*n* º elemento de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1610) [operaciones en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1602) [representación impresa de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1576) [cita de técnicas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2012) [de inversión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1644) [para manipular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1604) [lista->árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2192) [lista-diferencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6394) [lista-de-valores-arg](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4726) [lista-de-argmentos-retrasados ​​lista-de-valores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4728) [lista-de-valores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4256) [lista-ref](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1612) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4780) [memoria estructurada por lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5828) [unión-de-lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6392) [lives-near (regla)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5140) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5154) [evolución local de un proceso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_632) [nombre local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_578) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1060) [estado local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2838) [mantenido en marcos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3100) [variable de estado local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2844) [ubicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5836) [de la variable local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1058) [Locke, John](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6) [logaritmo (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1134) [logaritmo, que aproxima ln 2](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3982) [crecimiento logarítmico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_784) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_808) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2148) [programación lógica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5028) , *véase también* lenguaje de consulta; intérprete de consultas [computadoras para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5084) [historia de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5040) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5080) [en Japón](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5082) [lenguajes de programación lógica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5076) [lógica matemática vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5246) [acertijos lógicos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4928) [lógico y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3348) [lógico o](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3352) [lógico-no](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3380) búsqueda [en tabla unidimensional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3280) [en conjunto de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2210) [en tabla bidimensional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3290) [etiqueta de búsqueda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5714) [búsqueda-prim](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5798) [valor-de-variable de búsqueda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4456) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4480) [para definiciones escaneadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4612) [construcciones de bucle](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_554) ,  
  
[[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_690)   
    [Implementación en el evaluador metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4426)   
[del límite inferior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1506)

[lenguaje de máquina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6206)   
    [lenguaje de alto nivel vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4192)   
[Macintosh](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6192)   
[MacLisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_46)   
[macro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4374) , *ver también* lector macro   
mago de caracteres, *ver* analista numérico   
magnitud [representación polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2348)   
    [dirigida por datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2468) [representación rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2332) [con datos etiquetados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2410) [magnitud-polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2394) [magnitud-rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2382) [make-account](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2896) [en modelo de entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3126) [con serialización](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3616) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3618) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3620) [make-account-and-serializer](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3630) [make-accumulator](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2904) [make-agenda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3412) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3456) [make-assign](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5722) [make-begin](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4338) [make-branch](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5742) [make-center-percent](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1522) [make-center-width](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1516) [make-code-tree](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2258) [make-compiled-procedure](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6332) [make-complex-from-mag-ang](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2546) [make-complex-from-real-imag](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2544) [make-connector](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3546) [make-cycle](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3172) [make-decrementer](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2984) [make-execution-procedure](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5718) [make-frame](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1914) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1944) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4470) [make-from-mag-ang](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2416) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2474) [paso de mensajes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2492) [representación polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2354) [representación rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2338) [make-from-mag-ang-polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2400) [make-from-mag-ang-rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2388) [make-from-real-imag](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2414) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2472) [representación polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2352) [de paso de mensajes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2488) [representación rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2336) [make-from-real-imag-polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2398) [make-from-real-imag-rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2386) [make-goto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5748) [make-if](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4324) [make-instruction](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5704) [make-instruction-secuence](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6266) [make-interval](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1486) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1502) [make-joint](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3028) [make-label](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6322) [make-label-entry](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5712) [make-lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4314) [make-leaf](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2250) [make-leaf-set](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2276) [make-machine](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5626) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5638) [make-monitored](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2906) [make-mutex](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3668) [make-new-machine](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5672) [make-operation-exp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5790) [make-perform](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5764) [make-point](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1406) [make-poly](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2686) [make-polinonomial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2750) [make-primitive-exp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5774) [make-procedure](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4444) [make-product](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2070) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2092) [make-queue](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3232) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3252) [make-rat](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1300) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1348) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1390) [axioma para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1416) [reducir a los términos más bajos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1378) [make-racional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2532) [make-register](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5642) [make-restore](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5756) [make-save](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5754) [make-scheme-number](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2522) [make-segment](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1398) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1954) [make-serializer](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3660) [make-simplified-withdraw](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2982) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4150) [make-stack](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5650) [con pila monitoreada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5812) [hacer suma](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2068) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2088)  
  
make-table   
    [implementación de paso de mensajes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3296)   
    [tabla unidimensional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3286)   
[make-tableau](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3974)   
[make-term](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2702) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2744)   
[make-test](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5736)   
[make-time-segment](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3442)   
[make-tree](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2158)   
[make-vect](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1932)   
[make-wire](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3358) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3396) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3434)   
[make-withdraw](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2894)  
     [en el modelo de entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3104)   
    [usando let](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3106)  
realizar cambios, *ver* contar cambios   
[map](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1666) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4782)   
    [como acumulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1766)   
    [con múltiples argumentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1664)   
[map-over-symbols](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5412)   
[map-successive-pairs](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4130)  
mapeo   
    [sobre listas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1660)   
    [anidado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1818) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3988)   
    [como un transductor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1734)   
    [sobre árboles](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1706)   
[mark-sweep recolector de basura](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5956)  
función matemática, *ver* función (matemática)   
matemáticas   
    [informática vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_496) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5034)   
    [ingeniería vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_918)   
[matriz, representada como secuencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1796)   
[matriz-\*-matriz matriz](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1808)   
[-\*-vector](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1804)   
[máx. (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1494)   
[McAllester, David Allen](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4892) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4902)   
[McCarthy, John](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_28) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_32) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_36) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4824)   
[McDermott, Drew](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4878)   
[MDL](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5984)   
[medios de abstracción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_126)   
    [definir](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_232)  
[medios de combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_122) , *ver también* medida de cierre   
[en un](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2792)   
[miembro de anillo euclidiano](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4936)   
[memo-fib](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3324)   
[memo-proc](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3804)  
[memoización](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_736) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3314)   
    [llamada por necesidad y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3904)   
    [por demora](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3802)  
     [recolección de basura y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4746)   
    [de thunks](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4710)   
[memoize](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3326)  
memoria   
    [en 1964](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4838) [memq](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2030)  
     [estructurado en lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5830) [fusión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3898) [fusión ponderada por](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4008) fusión flujos infinitos, *ver* flujo(s) infinito(s) [paso de mensajes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1466) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2486) [modelo de entorno y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3122) [en cuenta bancaria](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2900) [en simulación de circuitos digitales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3394) [recursión de cola y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_704) [evaluador metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4212) [evaluador metacircular para Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4210) [análisis](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4638) [de combinaciones de versiones (aplicaciones de procedimiento)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4384) [compilación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6516) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6524) [abstracción de datos en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4224) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4234) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4430) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4486) [expresiones derivadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4352) [de evaluación dirigida por datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4390) [bucle de controlador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4516) [eficiencia del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4640) [modelo de entorno de evaluación en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4218) [operaciones de entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4454) [eval y aplicar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4226) [eval -](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4220)  
  
    [ciclo de aplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4220) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4228)   
    [representación de expresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4232) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4274)   
    [entorno global](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4496)   
    [procedimientos de orden superior en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4252)   
    [lenguaje implementado vs. lenguaje de implementación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4260)   
    [trabajo de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4222)   
    [orden de evaluación de operandos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4272)   
    [procedimientos primitivos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4494)   
    [representación de entornos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4464)   
    [representación de procedimientos representación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4438) [ejecución](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4492)   
    [verdadera y falsa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4432) [formas especiales (adicionales)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4394) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4404) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4406) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4412) , [[5]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4420) , [[6]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4424) [formas especiales como expresiones derivadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4354) [diferenciación simbólica y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4278) [sintaxis del lenguaje evaluado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4276) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4386) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4428) [recursividad de cola no especificada en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6094) [verdadero y falso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4500) [abstracción metalingüística](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4194) [MicroPlanner](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4874) [Microshaft](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5100) [segmento de punto medio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1408) [Miller, Gary L.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_956) [Miller, James S.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6368) [Miller-Rabin prueba de primalidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_954) [Milner, Robin](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4110) [min (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1490) [Minsky, Marvin Lee](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-7.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5936) [Miranda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1824) [MIT](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5066) [Artificial Intelligence Laboratory](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_84) [historia temprana del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1866) [Proyecto MAC](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_54) [Research Laboratory of Electronics](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_34) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5940) MIT Scheme [el flujo vacío](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3756) [eval](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4580) [definiciones internas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4610) [números](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_544) [aleatorio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2964) [entorno inicial del usuario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4582) [sin interrupciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3678) [ML](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4106) modelado [móvil](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1700) [como estrategia de diseño](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-19.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2826) [en ciencia e ingeniería](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_336) [modelos de evaluación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6130) registros modificados, *ver* secuencia de instrucciones [modifica-registro?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6388) [modularidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1756) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-19.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2824) [a lo largo de los límites de los objetos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4184) [programas funcionales vs. objetos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4116) [principio de ocultamiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2892) [flujos y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3942) [mediante el despacho en el tipo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2428) [mediante flujos infinitos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4134) [mediante el modelado con objetos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2920) [módulo *n*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_892) [*modus ponens*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5250) dinero, cambio, *ver* conteo cambio [procedimiento monitoreado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2908) [integración de Monte Carlo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2956) [formulación de flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4138) [simulación de Monte Carlo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2936) [formulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4122) [de flujo flujo infinito de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4132) [Monte Carlo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2952) [Moon, David A.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_50) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5954) lucero de la mañana, *ver* lucero de la tarde [Morris, JH](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3008) [código Morse](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2228) [Mouse, Minnie y Mickey](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5262) [mul (genérico)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2508)  
  
    [utilizado para coeficientes polinómicos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2716)   
[mul-complejo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2316)   
[mul-intervalo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1498)  
     [versión más eficiente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1514)   
[mul-poli](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2676)   
[mul-rat](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1312)   
[mul-series](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3932)   
[mul-streams](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3886)   
[mul-terms](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2712)  
[Multics sistema de tiempo compartido](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5946)   
[vivienda múltiple](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4938)   
[multiplicando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2082)  
[multiplicación por el método campesino ruso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_822)   
multiplicador   
    [restricción primitiva](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3538)   
    [selector](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2080)   
[Munro, Ian](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1790)   
[objetos de datos mutables](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3128) , *véase también* cola; tabla   
    [implementada con asignación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3198)   
    [estructura de lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3132)   
    [pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3136)   
    [representación procedimental de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3196)   
    [datos compartidos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3190)   
[mutador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3130)   
[mutex](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3640)   
[exclusión mutua](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3646)   
[misterio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3174)

nombre, *ver también* nombre local; variable; variable local   
    [encapsulada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2888)   
    [de un parámetro formal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_582)   
    [de un procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_310)   
[llamado let (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4416)  
denominación   
    [de objetos computacionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_216)   
    [de procedimientos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_300)  
convenciones de nomenclatura   
    [! para asignación y mutación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2868)   
    [? para predicados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_520)   
[idioma nativo de la máquina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6204)  
análisis en lenguaje natural   
    , *ver* análisis en lenguaje natural   
    [cita en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1994)  
registros necesarios, *ver* secuencia de instrucciones   
[¿registro de necesidades?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6386)   
[negar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5328)  
[aplicaciones anidadas de car y cdr](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1584)  
[combinaciones anidadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_196)  
definiciones anidadas, *ver* definición interna   
asignaciones anidadas, *ver* asignación   
[nuevo registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5992) [registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5972)   
[de coches nuevos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5972) [registro de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5974)   
[cdrs nuevos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5974)   
[nueva retirada nueva](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2884)   
[línea (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_938) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1362)  
método de Newton   
    [para raíces cúbicas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_562)   
    [para funciones diferenciables](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1178)   
    [método de medio intervalo vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1182)   
    [para raíces cuadradas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_510) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1200) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1212)   
[transformada de newton](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1196)   
[método de newtons](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1198)   
[siguiente (descriptor de enlace)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6252)   
[siguiente a (reglas)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5164)   
nulo   
    [prescindir de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2020)   
    [una lista vacía](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1596)   
    [como marcador de fin de lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1562)   
    [como variable ordinaria en Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1598)   
[¿no más expresiones?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6098)   
[¿no operandos?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4346)  
[nodo de un árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_256)   
[no computable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4594)   
[no determinismo no estricto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4686)   
[, en comportamiento de programas concurrentes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3604) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4180)   
[punto de elección no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4836)   
[computación no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4806)   
[evaluador no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4978)   
    [orden de evaluación de operandos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4970)   
[programación no determinista vs. programación Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4810) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4940) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4950) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5442)  
programas no deterministas   
    [acertijos de lógica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4930)   
    [pares con sumas primos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4812)   
    [análisis sintáctico lenguaje natural](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4954)   
    [ternas pitagóricas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4914) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4918) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4922)   
[evaluación de orden normal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_340)   
    [orden aplicativo vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_480) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_852) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4680)   
    [evaluación retrasada y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4092)   
    [en evaluador de control explícito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6126)   
    [de si](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_484)  
evaluador de orden normal, *véase* evaluador perezoso   
[no (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_462)   
[no(lenguaje de consulta)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5130) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5268)   
    [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5204) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5326) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5438)  
notación en este libro   
    [símbolos en cursiva en sintaxis de expresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_314)   
    [caracteres inclinados para la respuesta del intérprete](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_152)   
[sustantivos raíz](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4956)   
[*n* -ésima, como punto fijo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1242)   
[¿nulo? (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1614)   
    [implementado con punteros tipados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5912)   
[teoría de números](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_874)  
número(s)   
    [comparación del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_404)   
    [punto decimal en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_540)   
    [la igualdad de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_406) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2038) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5878)   
    [en la implementación del sistema aritmético genérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2512)   
    [dependencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_548)   
    [entero vs. número real](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_140)   
    [entero, exacto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_538)   
    [en Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_148)   
    [número racional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_536)   
[¿número? (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2052)   
    [tipos de datos e](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2554)   
    [implementado con punteros tipados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5916)   
[numer](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1302) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1350)   
    [axioma para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1418)   
    [reducir a los términos más bajos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1392)   
[análisis numérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_142)   
[analista numérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1086)   
[datos numéricos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_132)

[objeto obarray](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5886)   
[programa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6218)   
[objeto(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-19.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2828)   
    [beneficios de modelar con](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2918)   
    [con estado variable en el tiempo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2850)   
[lenguajes de programación orientados a objetos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2622) [registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5990)   
[antiguo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5990) [registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5994)   
[oldcr](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5994)   
[unos (flujo infinito)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3862)   
    [versión de lista diferida](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4788)   
[op (en máquina de registros)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5522)   
    [simulando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5788)   
[codificación abierta de primitivos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6424) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6460)   
[operandos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4344)  
[operandos de una combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_184)  
operación   
    [de tipo cruzado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2572)   
    [genérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1276)   
    [en máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5472)   
[tabla de operaciones y tipos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2436)   
    [asignación necesaria para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2856)   
    [implementar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3298)   
[operación-exp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5792)   
[operación-exp-op](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5794)   
[operación-exp-operandos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5796)   
[operador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4342)  
[operador de una combinación combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_182)   
    [como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1166)   
    [expresión compuesta como expresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_478)   
    [lambda como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1052)  
optimalidad   
    [de la regla de Horner](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1782)   
    [del código de Huffman](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2244)   
[or (lenguaje de consulta)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5128)   
    [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5200) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5322)   
[or (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_454)   
    [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_458)   
    [por qué una forma especial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_468)   
    [sin subexpresiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4398)   
[or-gate](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3350)   
    [or-gate](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3384) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3386)   
[orden](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2704) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2746)   
[orden notación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_760)  
orden de evaluación   
    [asignación y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3032)   
    [dependiente de la implementación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3066)   
    [en compilador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6414)   
    [en control explícito evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6070)   
    [en evaluador metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4270)   
    [en Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3030)  
orden de eventos   
    [desacoplamiento aparente de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3794)   
    [la indeterminación real en sistemas concurrentes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3590)   
[orden de crecimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_752)   
    [proceso iterativo lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_768)   
    [proceso recursivo lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_762)   
    [logarítmico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_806)   
    [proceso recursivo de árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_774)  
orden de evaluación de subexpresiones, *ver* orden de evaluación   
[representación de conjuntos en lista ordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2126)   
[números ordinarios (en el sistema aritmético genérico)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2514)   
[marco de origen](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1916)  
[Ostrowski, AM](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1784)   
[superado por (regla)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5152) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5276)

[P operación en paquete](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2442) [semáforo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3654) [representación polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2450)   
    [de número complejo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2536) [representación rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2444) [de número racional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2524) [polinomial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2678) [pintor(es)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1862) [de números de Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2516) [operaciones de orden superior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1898) [operaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1876) [representadas como procedimientos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1946) [transformación y combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1960) [par(es)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1318) [definición axiomática de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1458) [notación de caja y puntero para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1526) [flujo infinito de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3984) [representación procedimental](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1450) [mutable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3138) [perezosa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4762) de , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3200) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4772) [representado usando vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5852) [usado para representar secuencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1558) [usado para representar árbol ¿](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1678) [par? (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1686) [implementado con punteros tipificados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5910) [pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4002) [Pan, VY](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1786) [ejecución en paralelo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3612) [secuencias de instrucciones en paralelo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6402) paralelismo, *ver* concurrencia parámetro, *ver* parámetros formales paso de parámetros, *ver* llamada por nombre paso de argumentos; argumento de llamada por necesidad paso de paréntesis [delimitando combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_178) [delimitando cláusulas cond](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_360) [en la definición de procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_328) [parse](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4966) [parse-...](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4964) [parsing lenguaje natural](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4952) [comprensión del lenguaje real vs. analizador de juguete](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4972) [sumas parciales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3892) [Pascal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_288) [falta de procedimientos de orden superior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4100) [procedimientos recursivos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_686) [restricciones en datos compuestos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1548) [debilidad en el manejo de objetos compuestos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3572) [Pascal, Blaise](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_742) [triángulo de Pascal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_738) [cuenta bancaria protegida por contraseña](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2914) [patrón](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5112) [coincidencia de patrones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5176) [implementación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5342) [unificación vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5212) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5222) [patrón representación variable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5114) [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5298) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5400) [coincidencia de patrones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5346) [registro de pc](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5662) [realizar (en máquina de registros)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5544) [simulando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5762) [acción de realizar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5766) [Perlis, Alan J.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-6.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1552) [chistes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_208) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_278) [permutaciones de un conjunto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1832) [permutaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1840) [Phillips, Hubert](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4942) (pi) [aproximación con el método de medio intervalo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1094) [aproximación con integración de Monte Carlo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2958) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4140) [CesÃ estimación de ro para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2940) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4120) [serie de Leibniz para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_974) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3950) [flujo de aproximaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3954) [fórmula de Wallis para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1014) [flujo pi](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3962) [pi-suma](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_978) [con procedimientos de orden superior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1000) [con lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1030) [lenguaje de imágenes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1854) [Pingala, Ã Charya](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_812)  
  
[canalización](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3582)   
[Pitman, Kent M. Punto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_52)   
[de planificación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4866)   
[, representado como un par](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1404)  
de punteros   
    [en notación de caja y puntero](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1530) [paquete polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2452)  
     [tipificado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5862) [polar?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2376) [paquete polinomial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2680) [poli](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2664) [aritmética de polinomios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2652) [adición](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2670) [división](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2758) [algoritmo de Euclides](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2788) [máximo común divisor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2784) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2810) [interconectado con el sistema aritmético genérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2682) [multiplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2672) [algoritmo probabilístico para MCD](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2812) [funciones racionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2780) [resta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2756) [polinomio(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2650) [forma canónica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2772) [densa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2728) [evaluación con la regla de Horner](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1772) [jerarquía de tipos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2766) [indeterminado de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2656) [escaso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2732) [univariado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2660) [pop](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5652) [Lisp portátil estándar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_66) [portando un lenguaje](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6504) [serie de potencias, como flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3910) [adición](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3926) [división](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3934) [integración](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3922) [multiplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3928) [PowerPC](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3712) [predicado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_390) [de cond cláusula](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_366) [de if](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_434) [convención de nombres para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_518) [código de prefijo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2234) [notación de prefijo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_190) [notación infija vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2098) [preposiciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4968) [preservando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6264) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6272) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6398) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6420) [impresión bonita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_198) [número(s) primo(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_858) [criptografía y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_930) [tamiz de Eratóstenes para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3842) [la prueba de Fermat para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_870)     flujo infinito de, *ver* primos [prueba de Miller-Rabin para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_950) [probar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_856) [pares de suma prima pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4814) [de suma prima](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1828) [flujo infinito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3990) [¿primo?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_866) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3874) [primos (flujo infinito)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3854) [definición implícita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3872) [restricciones primitivas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3510) [expresión primitiva](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_120) [evaluación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_264) [nombre de procedimiento primitivo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_158) [nombre de variable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_218) [número](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_150) procedimientos primitivos (aquellos marcados *ns* no están en el estándar IEEE Scheme) [\*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_166) [+](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_164) [-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_172) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_412) [/](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_174) [<](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_400) [=](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_402) [>](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_398) [aplicar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2460) [atan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2326) [car](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1326) [cdr](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1330) [cons](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1322) [cos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1114) [mostrar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1360) [eq?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2028) [error ( *ns* )](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1090) [eval ( *ns* )](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4574) [lista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1570) [log](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1136) [max](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1496) [min](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1492) [nueva línea](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1364) [¿no es](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_464) [nulo?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1616) [¿número?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2054) [¿par?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1688) [cociente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3908) [aleatorio ( *ns* )](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_902) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2968) [leer](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4524) [resto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_802) [redondear](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2644)  
  
    [tiempo de ejecución ( *ns* )](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_934)   
    [¡conjunto-automático! ¡](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3142)   
    [conjunto-cdr!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3146) [¿símbolo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2062)   
    [sin ?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1118) [referencia-vectorial ¡](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5846) [conjunto-vectorial!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5850) consulta primitiva, *ver* consulta simple [aplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6076) [-primitiva implementación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4508) [-primitiva nombres-procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4510) [-primitivo objetos-procedimiento-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4512) [primitivo ¿procedimiento-primitivo?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4442) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4506) [principio de mínimo compromiso operación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2362) [de impresión en máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5540) [punto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1410) [de impresión cola de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3260) [impresión print-rat](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1370) [resultado de impresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6146) [versión de pila monitorizada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6164) [estadísticas de pila de impresión operación en máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5810) [impresión, primitivas para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1356) [algoritmo probabilístico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_908) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2814) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3850) sonda [en sistema de restricción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3542) [en simulador de circuito digital](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3432) [proc registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6026) [abstracción procedimental](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_574) [representación procedimental de datos datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1448) [mutables](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3202) [procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_114) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_128) [anónimo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1044) [número arbitrario de argumentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_192) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1654) [como argumento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_968) [como caja negra](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_570) [cuerpo de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_322) [compuesto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_298) [creando con define](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_304) [creando con lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1038) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3070) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3074) [como datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_116) [definición de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_292) [primera clase en Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1224) [parámetros formales de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_318) [como método general](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1080) [genérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2270) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2294)     orden superior, *ver* procedimiento de orden superior [implícito comienza en el cuerpo de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2882) [la función matemática vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_488) [memorizado nombre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3320) [monitorizado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2910) [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_312) [nombrar (con define )](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_302) [como patrón para la evolución local de un proceso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_630) [como valor devuelto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1158) [devolviendo múltiples valores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5698) [alcance de parámetros formales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_600) [forma especial vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4694) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4766) procedimiento aplicación [combinación que denota](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_180) [entorno modelo de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3084)     sustitución modelo de, *ver* modelo de sustitución de procedimiento aplicación [cuerpo del procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4450) [entorno de procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4452) [parámetros de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4448) [procedimiento proceso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_10) [iterativo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_668) [lineal iterativo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_676) [lineal recursivo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_664) [evolución local del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_634) [orden de crecimiento de los recursos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_754) [recursivos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_656) [requeridos por](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_756) [la forma del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_650) [árbol recursivo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_714) [producto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1010) [como acumulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1022) [¿producto?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2078) [programa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_14) [como máquina abstracta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4544)  
      [comentarios en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1838)   
    [como datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4540)   
    [desarrollo incremental de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_236)   
    [la estructura de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_238) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_566) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_610) , *ver también* barreras de abstracción   
    [estructurado con subrutinas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4564)   
[contador de programa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5664)  
programación   
    dirigida por datos, *ver* programación dirigida por datos [elementos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_118)   
    [impulsados ​​por la demanda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3792) de     funcional, *ver* programación funcional [imperativo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3016) [estilo odioso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3828) [lenguaje de programación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_16) [diseño de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4670) [lógica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5078) [funcional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4158) [orientado a objetos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2624) [fuertemente tipado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4104) [de muy alto nivel](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_504) [Prolog](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4876) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5056) [solicitud de entrada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4534) [solicitudes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4520) [control explícito evaluador evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6142) [perezoso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4734) [evaluador metacircular evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4530) [no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5022) [intérprete de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5292) [propagar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3426) [propagación de restricciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3482) [probar que los programas son correctos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_502) [secuencia pseudoaleatoria](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2926) [pseudodivisión de polinomios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2802) [pseudoresto de polinomios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2804) [push](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5654) [put](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2438) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3302) rompecabezas [rompecabezas de ocho reinas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1850) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4948) [rompecabezas de lógica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4926) Ternas pitagóricas [con programas no deterministas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4912) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4916) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4920) [con flujos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4004)

[qeval](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5228) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5310)   
[mecánica cuántica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4186)   
[quasiquote](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6284)   
[queens](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1852)  
[query](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5092) , *ver también* consulta simple; consulta compuesta  
[intérprete de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5094)   
    [agregando regla o aserción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5242)  
    consulta compuesta, *ver* consulta compuesta  
     [base de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5372)   
    [controlador bucle](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5234) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5290)   
    [estructura del entorno en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5452)   
    [marco](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5178) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5414)   
    [mejoras a](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5282) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5426) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5432)   
    [bucles infinitos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5260) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5284)   
    [instanciación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5302)   
    [Intérprete de Lisp vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5218) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5230) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5446)   
    [descripción general](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5172)   
    [coincidencia de patrones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5174) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5340)   
    [representación de variable de patrón](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5300) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5398)   
    [problemas con not y lisp-value](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5266) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5434)   
    [evaluador de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5226) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5306)  
    regla, *ver* regla  
    consulta simple, *ver* consulta simple  
     [operaciones de flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5384)   
    [flujos de marcos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5184) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5238)   
    [sintaxis de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5396)   
    [la unificación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5208) del lenguaje de consultas , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5360)   
[lenguaje de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5090) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5096)   
    [abstracción en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5138)  
    consulta compuesta, *ver* consulta compuesta  
     [base de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5102)   
    [prueba de igualdad en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5134)   
    [extensiones a](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5280) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5420)   
    [deducciones lógicas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5156)   
    [lógica matemática vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5244)  
    regla, *ver* regla  
    consulta simple, *ver* consulta simple  
[bucle controlador de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5294)  
[signo de interrogación, en nombres de predicado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_522)   
[cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3220)   
    [doble extremo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3264)   
    [frente a](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3224)   
    [operaciones en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3230)   
    [implementación procedimental de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3262)   
    [parte posterior de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3222)   
    [en simulación agenda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3440)   
[cita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1988)   
    [de cadenas de caracteres](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2004)   
    [de objetos de datos Lisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1998)   
    [en lenguaje natural](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1992) [comillas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2014)  
[simples y dobles](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2002) [(forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2014) [leer y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4528) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5408) [¿entre comillas?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4286) [cociente (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3906)  
  
[Rabin, Michael O.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_958)   
[radicando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_514)   
[Números de Ramanujan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4010)   
[Ramanujan, Srinivasa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4014)   
[rand](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2934)  
     [con reinicio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2974)   
[aleatorio (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_900)   
    [asignación necesaria para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2852) [el generador de números](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2854) [aleatorios dentro de un rango](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2970)  
     [del Esquema MIT](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2966) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2922) [en simulación de Monte Carlo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2938) [en pruebas de primalidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_894) [con reinicio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2972) [con reinicio, versión de flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4136) [números aleatorios (flujo infinito)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4124) [Raphael, Bertram](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5048) [paquete racional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2526) [función racional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2776) [reducción a los términos más bajos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2806) número(s) racional(es) [operaciones aritméticas en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1296) [en el Esquema MIT](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_526) [impresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1354) [reducción a los términos más bajos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1372) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1386) [representado como pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1346) [aritmética de números racionales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1298) [interconectada con el sistema aritmético genérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2528) [necesidad de datos compuestos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1260) [Raymond, Eric](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4678) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4850) [circuito RC](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4028) [lectura (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4522) [manejo de notación de cola punteada por](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5352) [caracteres macro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5404) [operación de lectura](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5538) [en la máquina de registro bucle de lectura-evaluación-impresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_204) , *véase también* bucle de controlador bucle [de lectura-evaluación-impresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6144) [lector macro carácter](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5402) [número real](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_138) parte real [representación polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2344) [dirigida por datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2464) [representación rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2328) [con datos etiquetados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2406) [parte real-polar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2390) [parte real-rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2378) [procedimiento de recepción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5690) [de punto trasero](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3244) [registro, en una base de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2204) [rectángulo, que representa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1412) [un paquete rectangular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2446) [¿rectangular?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2374) [recursión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_248) [datos dirigidos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2722) [que expresan un proceso complicado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_250) [en reglas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5150) [al trabajar con árboles](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1680) [ecuaciones de recursión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_26) [teoría](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4568) de la recursión procedimiento recursivo [definición de procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_564) [recursivo proceso recursivo vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_680) [especificar sin definir](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4624) [proceso recursivo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_654) [proceso iterativo vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_638) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3090) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5568) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6410) [lineal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_660) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_766) [procedimiento recursivo vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_682) [máquina de registro para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5564) [árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_716) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_778) [árbol rojo-negro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2174) [que se reduce a los términos más bajos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1374) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1388) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2808) [Rees, Jonathan A.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4380) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4644) [transparencia referencial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2990) [reg (en máquina de registro)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5526) [simulación de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5768) [acciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5536) [de máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-30.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5458) [controlador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5466)  
  
    [diagrama de controlador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5486)   
    [rutas de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5464)   
    [diagrama de ruta de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5480)   
    [diseño de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5462)   
    [lenguaje para describir](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5492)   
    [monitoreo desempeño](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5804)   
    [simulador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5622)   
    [pila](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5562)   
    [subrutina](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5550)   
    [operación de prueba](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5484)   
[tabla de registros, en simulador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5658)   
[registro(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-30.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5460)   
    [que representan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5640)   
    [rastreo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5822)   
[registro-exp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5776)   
[registro-exp-reg](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5778)  
lenguaje de máquina de registros   
    [asignar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5516) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5598)   
    [rama](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5504) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5606)   
    [const](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5528) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5596) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5616)   
    [punto de entrada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5498)   
    [goto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5512) , [ [2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5610)   
    [instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5494) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5592) [etiqueta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5508)  
     [etiqueta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5496) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5608) [op](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5520) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5600) [realizar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5542) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5602) [reg](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5524) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5594) [restaurar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5578) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5614) [guardar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5574) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5612) [probar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5500) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5604) [simulador de máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5624) [registros modificados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6382) [registros necesarios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6380) [relaciones, computación en términos de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3508) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5038) [relatividad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3724) [relativamente prima , teoría de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1028) [liberación un mutex](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3644) [resto (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_800) [resto módulo *n*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_890) [términos restantes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2796) [eliminar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1842) [¡elimine el primer elemento de la agenda!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3418) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3478) [requiere](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4828) [como una forma especial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5026) [palabras reservadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6426) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6462) [fórmula](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1482) de resistencia para resistencias en paralelo , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1524) [tolerancia de resistencias](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1484) [principio de resolución](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5046) [resolución, cláusula de Horn](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5060) [expresiones-resto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4334) [operandos-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4350) [resto segmentos-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3468) [resto términos-resto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2700) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2740) [restaurar (en máquina de registro)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5580) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5800) [implementación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5922) [simulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5758) [retorno (descriptor de enlace)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6254) [retorno de múltiples valores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5696) [Reuter, Andreas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3706) [revertir](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1642) [como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1816) [reglas de plegado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5286) [Rhind Papiro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_824) [rama derecha](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2156) , [[2] división](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2262) [derecha](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1890) [sumador de acarreo de ondulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3388) [Rivest, Ronald L.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_922) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2180) [circuito RLC](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4086) [Robinson, JA](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5068) [robustez](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1982) [canciones de rock, década de 1950](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2282) [Rogers, William Barton](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1868) [registro raíz](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5966) raíces de ecuación, *véase* método de medio intervalo; método de Newton [rotate90](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1968) [round (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2642) [error de redondeo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_144) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2308)  
      
[Rozas, Guillermo Juan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6370) [Regla](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5136)   
[del algoritmo RSA (lenguaje de consulta)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_928) [aplicando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5216) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5356) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5454) [sin cuerpo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5146) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5160) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5336) [Runkle, John Daniel](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1870) [tiempo de ejecución (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_932) [Método campesino ruso de multiplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_820)  
  
      
[mismo (regla)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5142)   
[misma-variable?,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2066) [ [2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2668)  
igualdad y cambio de  
     [significado de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2986)   
    [datos compartidos y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3180)   
[satisfacer una consulta compuesta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5126)   
[satisfacer un patrón (consulta simple)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5118)   
[guardar (en máquina de registros)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5576) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5802)   
    [implementación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5920)   
    [de simulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5752)   
[de lista de escala](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1662) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1668) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4784)   
[flujo de escala](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3870)   
[árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1708) de escala , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1710)   
[escala-vect](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1942)   
[escaneo de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5978)   
[definiciones de escaneo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4614)  
[escaneo de definiciones internas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4606)   
    [en el compilador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6446) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6454) [Historial de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_86)   
[Scheme de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_42) [Scheme chip](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5998) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6010) [paquete de número de esquema](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2518) [número de esquema->complejo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2584) [número de esquema->número de esquema](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2628) [Schmidt, Eric](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3010) [alcance de una variable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_594) , *ver también* alcance léxico [definición interna](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4602) [en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1072) [los parámetros formales del procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_602) [let](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1072) búsqueda [de árbol binario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2146) [búsqueda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1084) [sistemática](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4844) [en profundidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4856) [secretaria, importancia de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5108) [la cola de segmentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3446) [segmentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3462) [de tiempo de segmento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3444) [segmentos->pintor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1950) [selector](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1290) [como barrera de abstracción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1384) [genérico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2402) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2420) [expresión autoevaluada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4240) [¿autoevaluación?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4282) [semáforo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3648) [de tamaño *n*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3692) [punto y coma](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_284) [comentario introducido por](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1834) [código separador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2232) [acelerador de secuencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3966) secuencia de expresiones [en consecuente de cond](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_444) [en el cuerpo del procedimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_326) [secuencia(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1554) [como interfaz convencional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1718) [como fuente de modularidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1758) [operaciones en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1738) [representado por pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1556) [secuencia->exp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4336) [intercambio serializado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3634) [con evitación de bloqueos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3708) [serializador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3610) [implementando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3638) [con múltiples recursos compartidos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3622) [serie, suma de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_982) [aceleración de secuencia de aproximaciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3964) [con flujos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3956) conjunto [conjunto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2100) (forma especial), *ver también* [base de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2200) de asignación [como operaciones en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2102) [permutaciones de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1830) [representado como árbol binario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2136) [representado como lista ordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2124) [representado como lista desordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2112) [subconjuntos de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1714) [conjunto! (forma especial)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2860) [modelo de entorno de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3080) [valor de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2864) [conjunto-car! (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3140) [implementado con vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5900) [implementación procedimental de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3216) [valor de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3148) [conjunto-cdr!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3144)  
      
[(procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3144)   
    [implementado con vectores](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5902)   
    [implementación procedimental del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3218)   
    [valor del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3152)   
[conjunto-contenido!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5646)   
[conjunto-tiempo-actual!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3460)   
[conjunto-ptr-frontal!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3246)   
[conjunto-proc-ejecucion-instrucción!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5710)   
[conjunto-ptr-trasero!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3248)   
[conjunto-contenido-registro!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5628) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5678)   
[conjunto-segmentos!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3464)   
[conjunto-señal!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3372) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3406)   
[conjunto-valor!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3526) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3554)   
[conjunto-valor-variable!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4462) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4482)   
[entorno de configuración](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4502)  
[sombra un enlace](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3060)   
[Shamir, Adi](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_924)   
[forma de un proceso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_648)   
[datos compartidos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3178)   
[recursos compartidos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3624)   
[estado compartido](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3598)   
[encogerse a la esquina superior derecha](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1966)  
[Shrobe, Howard E.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5088)   
[error de efecto secundario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3000)   
[tamiz de Eratóstenes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3844) [suma](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_988)   
    [de tamiz](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3852) (sigma) notación procesamiento de señales [suavizado de una función](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1236) [suavizado de una señal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4042) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4050) [modelo de flujo de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4018) [cruces por cero de una señal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4034) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4038) , [[3] señal,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4046) [error de señal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6156) [digital](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3336) [diagrama de flujo de señal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1728) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4026) [procesamiento de señales vista de computación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1726) [procesamiento de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5182) [consultas simples](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5110) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5194) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5220) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5312) [consulta simple](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5314) [simplificación de expresiones algebraicas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2086) [Regla de Simpson para la integración numérica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1006) simulación     de circuitos digitales, *ver* simulación de circuitos digitales [impulsado por eventos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3332) [como herramienta de diseño de máquinas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6162) [para monitorear el rendimiento de la máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5806)     Monte Carlo, *ver* simulación de Monte Carlo     de la máquina de registros, *ver* simulador de máquina de registros [seno (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1116) aproximación de seno [para ángulo pequeño](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_786) [serie de potencias para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3918) [flujo singleton](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5394) [SKETCHPAD](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3486) [divisor más pequeño](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_860) [versión más eficiente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_942) [Smalltalk](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3490) [suavizado de una función](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1234) [suavizar una señal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4044) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4052) [snarf](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4674) [dinámica caótica del sistema solar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_102) [Solomonoff, Ray](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2930) [resolver ecuación diferencial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4066) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4076) [versión de lista perezosa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4804) [con definiciones escaneadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4616) resolver ecuación, *ver* método de medio intervalo; método de Newton; resolver [idioma fuente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6214) [programa fuente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6216) [Spafford, Eugene H.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6502) [forma especial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_272) [de polinomio disperso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2730) [como expresión derivada en el evaluador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4358)  
  
    [necesidad de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_558)   
    [procedimiento vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4692) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4764)  
formas especiales (aquellas marcadas *ns* no están en el estándar IEEE Scheme)   
    [y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_448)   
    [comienzan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2876)   
    [cond](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_354)   
    [cons-stream ( *ns* )](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3776)   
    [definen](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_226) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_308)   
    [retrasan ( *ns* )](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3770)   
    [si](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_422)   
    [lambda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1036)   
    [let](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1064)   
    [let\*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4410)   
    [letrec](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4620)  
     [nombrado let](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4418)   
    [o](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_456)   
    [quote](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2016)   
    [set!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2862) [bloque](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_624) [sqrt](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_550)   
[dividido](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1908) [estructurado en modelo de entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3118) [como punto fijo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1126) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1170) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1210) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1214) [como mejora iterativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1248) [con el método de Newton](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1202) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1216) [máquina de registro para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5548) [como límite de flujo flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3980) [sqrt](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3948) [cuadrado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_294) [en modelo de entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3072) [raíz cuadrada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_508) , *ver también* [flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3946) sqrt de aproximaciones [límite cuadrado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1894) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1906) [cuadrado de cuatro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1900) [más cuadrado (restricción)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3562) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3564) [pila](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_678) [de aplastamiento hacia adentro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1970) [enmarcada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6054) [para recursión en máquina de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5560) [que representa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5648) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5918) [asignación de pila y recursión de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6362) [nombre-reg-inst-pila](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5760) [Stallman, Richard M.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3498) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4886) [máquina de registro de inicio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5632) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5674) [start-eceval](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6478) [segmento-de-inicio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1400) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1956) estado     local, *ver* estado local [compartido](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3596) [se desvanece en la formulación de flujo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4154) [variable de estado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_670) , [[2] declaraciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2840) [locales](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2846) , *ver* [declaraciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6384) de secuencia de instrucciones [Steele, Guy Lewis Jr.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_88) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_700) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3024) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3500) , [[5]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4676) , [[6]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4848) [recolector de basura de detener y copiar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5932) [Stoy, Joseph E.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_338) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_830) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4630) [Strachey, Christopher](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1220) [diseño estratificado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1978) [stream(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-19.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2830) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3726) [evaluación retrasada y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4054) [vacío](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3746) [implementado como listas retrasadas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3732) [implementado como listas perezosas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4754) [definición implícita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3860)     infinito, *ver* infinito streams [utilizados en el intérprete de consultas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5186) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5236) [stream-append](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3994) [stream-append-delayed](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5386) [stream-car](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3740) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3780)  
  
[stream-cdr](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3742) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3782)   
[stream-enumerate-interval](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3788)   
[stream-filter](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3790)   
[stream-flatmap](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5390) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5418)   
[stream-for-each](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3762)   
[stream-limit](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3978)   
[stream-map](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3760)  
     [con múltiples argumentos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3824)   
[stream-null?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3750)  
     [en MIT Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3754)   
[stream-ref](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3758)   
[stream-withdraw](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4152)  
[cadena estricta](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4688)  
, *ver* cadena de caracteres  
[lenguaje fuertemente tipado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4102)   
[sub (genérico)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2506)   
[subcomplejo subintervalo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2314)   
[sub](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1508)   
[-rat](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1310)   
[sub-vect](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1940)  
[subrutina en máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5552)   
[subconjuntos de un conjunto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1716)   
[modelo de sustitución de procedimiento aplicación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_334) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3036)   
    [insuficiencia de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2980)   
    [forma de proceso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_644)   
[subtipo éxito](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2596)   
    [múltiple](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2620)   
[continuación (evaluador no determinista)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4988) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4996)   
[cuadratura sucesiva](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_796)   
[suma](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_990)  
     [como acumulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1020)   
    [versión iterativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1008)   
[suma de cubos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_972)  
     [con procedimientos de orden superior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_994)   
[suma de enteros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_970)  
     [con procedimientos de orden superior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_998)   
[suma de cuadrados impares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1722) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1752)   
[suma de cuadrados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_330)  
     [en el modelo de entorno](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3086)   
[suma de primos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3734) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3736)   
[suma?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2072)  
[suma de una serie](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_984)   
    [con](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3958)   
[supertipos de flujos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2600)   
    [múltiples](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2618)   
[Sussman, Gerald Jay](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_90) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_108) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_702) , [[4]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3496) , [[5]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4868) , [[6]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4880) , [[7]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4884)   
[Sussman, Julie Esther Mazel, sobrinas de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1990)   
[Sutherland, Ivan](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3488)   
[símbolo(s)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1986)   
    [igualdad de la cita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2024)   
    [interna](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5890)   
    [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2010)   
    [la representación de la](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5882)   
    [unicidad de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3184)   
[la hoja de símbolo ¿ símbolo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2254)   
[? (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2060)   
    [tipos de datos e](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2556)   
    [implementado con punteros tipados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5914)   
[álgebra simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2646)   
[diferenciación simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2044) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2476)   
[expresión simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1270) , *ver también* símbolo(s)  
[símbolos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2264)  
[SYNC](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3714)  
sincronización, *ver* concurrencia  
análisis sintáctico, separado de la ejecución  
     [en evaluador metacircular](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4634)   
    [en simulador de máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5684) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5730)   
[azúcar sintáctico](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_282)   
    [definir](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4298)   
    [let como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1066)   
    [construcciones de bucle como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_696)   
    [procedimiento vs. datos como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3400)  
sintaxis, *ver también* formas especiales  
    abstracto, *ver* sintaxis abstracta  
     [de expresiones, descripción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_316)   
    [de un lenguaje de programación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_276)  
[Interfaz de sintaxis](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3402)   
[de búsqueda sistemática](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4842)

[#t](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_374)  
[tabla](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3268)   
    [columna vertebral de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3274)   
    [para coerción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2586)   
    [para programación dirigida por datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2434)   
    [local](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3294)   
    [*n* -dimensional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3308)     operación y tipo  
     [unidimensional ,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3272) *ver* tabla de operación y tipo [representada como árbol binario vs. lista desordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3312) [prueba de igualdad de claves](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3304) [bidimensional](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3288) [usado en simulación agenda](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3448) [usado para almacenar valores calculados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3318) [tableau](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3972) [tabulación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_734) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3316) [tack-on-instruction-sequence](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6400) [arquitectura etiquetada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5872) [datos etiquetados](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2358) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5864) [lista etiquetada?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4290) [recursión de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_692) [compilador y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6354) [modelo de entorno de evaluación y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3096) [evaluador de control explícito y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6090) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6168) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6176) [recolección de basura y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6358) [evaluador metacircular y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6096) [en Scheme](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_706) [evaluador recursivo de cola](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6092) tangente [como fracción continua](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1154) [serie de potencia para](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3940) [registro de destino](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6248) [Universidad Tecnológica de Eindhoven](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3652) [Teitelman, Warren](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_60) [lista de términos de polinomio](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2666) [que representa el nodo terminal de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2724) [la lista de términos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2690) [de un árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_260) [prueba (en máquina de registros)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5502) [simulando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5734) [la operación de prueba en la máquina de registros](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-31.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5482) [¡prueba y establece!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3670) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3680) [condición de prueba](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5738) [texto de cita](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4288) [Thatcher, James W.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1432) [EL Sistema de Multiprogramación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3650) el [registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5892) de los autos , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5968) [vector](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5856) el [registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5894) de los cdrs , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5970) [vector](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5858) [el flujo vacío](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3748) [en el esquema del MIT](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3752) [la lista de términos vacía](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2692) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2736) [el entorno global](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4504) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6138) [demostración de teorema (automática)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5042) [( *f* ( *n* )) (theta de *f* ( *n* ))](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_758) [procesador](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4696) [llamada por nombre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3818) [llamada por necesidad](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3820) [forzando](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4706) [la implementación del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4738) [origen del nombre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4698) asignación de tiempo [y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3580) [comunicación y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3720) [en sistemas concurrentes](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3584) [programación funcional y](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4144) [en computación no determinista](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4820) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4834) [propósito del](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3586) [segmento de tiempo, en la agenda división de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3438) [tiempo prueba](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3674) [de primo temporizado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_936) [diagrama de tiempo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3602) [TK!Solver](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3502) [torre de tipos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2604) seguimiento [de ejecución de instrucciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5820) [asignación de registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5824) [pintor de transformadas transparencia,](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1962) [transposición](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1806) [referencial](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2992)

[](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_758)  
  
[un](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1806)  
árbol de matriz   
    [árbol B](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2168)   
    [binario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2140) , *véase también* árbol binario   
    [combinación vista como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_254)   
    [hojas de conteo de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1684)   
    [hojas de enumeración de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1748)   
    [franja de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1698)   
    [Huffman](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2242) [mapeo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1704)   
    [perezoso sobre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4796) [rojo-negro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2170) [representado como pares](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1676) [invirtiendo en todos los niveles](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1694) [acumulación de árboles](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_262) [árbol->lista...](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2186) [mapa de árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1712) [proceso recursivo de árbol](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_712) [orden de crecimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_776) [relaciones trigonométricas](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2342) [verdadero](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_368) [verdadero](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_376) [verdadero?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4434) [error de truncamiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_146) [verdad mantenimiento](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4894) [intentar de nuevo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4910) [máquina de Turing](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4560) [Turing, Alan M.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4556) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4590) [Turner, David](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1820) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3996) , [[3] campo de tipo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4174) [etiqueta de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2296) [tipo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5870) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2366) tipo(s) [de dos niveles](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2548) [operaciones de tipo cruzado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2576) [despacho en](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2432) [jerarquía en álgebra simbólica](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2768) [jerarquía de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2592) [reducción](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2612) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2640) [subtipo y supertipo múltiples](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2616) [elevación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2608) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2634) [subtipo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2598) [supertipo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2602) [torre de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2606) [mecanismo de inferencia de tipos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4114) [etiqueta de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-17.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2370) [tipo usando Scheme tipos de datos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2560) [puntero tipificado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5860) [tipificación expresiones de entrada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_202)  
      
  
[variable no vinculada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3054)   
[unev registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6030)   
[unificación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5210)   
    [descubrimiento de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5044)   
    [implementación de algoritmo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5362)   
    [coincidencia de patrones vs.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5214) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5224)   
[unificar-coincidir](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5364)   
[conjunto de unión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2108)  
     [representación de árbol binario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2194)   
    [representación de lista ordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2134)   
    [representación de lista desordenada](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2122)   
[único (lenguaje de consulta)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5424)   
[pares únicos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1844)  
[cuadrado unitario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1922)   
[polinomio univariante](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2658)   
[máquina universal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4552)   
    [control explícito evaluador como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6196)   
    [computadora de propósito general como](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6200)   
[Universidad de California en Berkeley](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_78)   
[Universidad de Edimburgo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5064)   
[Universidad de Marsella](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5062)   
[UNIX](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6188) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6500)   
[tipo-de-expresión-desconocido tipo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6152)   
[-de-procedimiento-](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6154)  
[desconocido representación-de-lista-desordenada de conjuntos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2114)  
valores-no-especificados   
    [definir](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_230)   
    [mostrar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1366)   
    [si sin alternativa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3476)   
    [nueva-línea](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1368)   
    [¡](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2866)   
    [conjunto! ¡conjunto-automático! ¡](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3150)   
    [conjunto-cdr! ¡](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3154)   
[split-up](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1896)   
[-insts! ¡actualizar-instancias!](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5702)   
[límite-superior](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1504)  
[compatibilidad ascendente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-27.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4752)   
[entorno-inicial-de-usuario (esquema-MIT)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4586)   
[impresión-de-usuario](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4538)  
     [modificado para código compilado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-35.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6480)

[V operación sobre semáforo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3656)   
[val registro](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6022)  
valor   
    [de una combinación](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_186)   
    [de una expresión](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_212) , *véase también* valores no especificados [variable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_220)   
[de procedimiento-valor](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-32.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5728) , *véase también* variable local [límite alcance](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_586) [libre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_592) [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_596) , *véase también* alcance de una variable valor [no vinculado](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3056) [de](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_222) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-21.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3052) [variable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2688) [código de longitud variable](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2224) [¿variable?](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2064) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4284) [vector (estructura de datos)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5842) vector (matemático) [operaciones sobre](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1800) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1930) [en marco de lenguaje de imagen](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1912) [representado como par](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1928) [representado como secuencia](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1798) [vector-ref (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5844) [vector-set! (procedimiento primitivo)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5848) [Verbos](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4958) [de Venus](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1996) [lenguaje de muy alto nivel](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_506)  
  
[Wadler, Philip](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3012)   
[Wadsworth, Christopher](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4112)   
[Wagner, Eric G.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1434)   
[Walker, Francis Amasa](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1874)   
[Wallis, John](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-12.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1016)   
[Wand, Mitchell](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-25.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4206) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-34.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_6068)   
[Waters, Richard C.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1760)   
[peso peso](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2266)   
[-hoja](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2256)  
[Weyl, Hermann](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-13.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1254)  
``qué es'' vs. ``cómo'' descripción, *ver* declarativo vs. imperativo   
[rueda de conocimiento (regla)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5144) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-29.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5278)   
[ancho ancho](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1520)  
[de un intervalo](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1510)   
[Wilde, Oscar (paráfrasis de Perlis de)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-10.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_210)   
[Wiles, Andrew](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-11.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_884)   
[Winograd, Terry](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4870)   
[Winston, Patrick Henry](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4906) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4974)   
[cable, en circuito digital](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3334)   
[Sabiduría, Jack](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_110)   
[Wise, David S.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3814)   
[ilusiones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1306) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-16.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2050)   
[retirar](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-20.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2858)  
     [problemas en sistema concurrente](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3592)   
[sin interrupciones](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-23.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3676)  
[línea del mundo de una partícula](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3728) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4156)   
[Wright, EM](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3884)   
[Wright, Jesse B.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1436)

[xcor-vect](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1934)  
[Centro de investigación de Xerox Palo Alto](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_64) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-22.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_3494)

[Operador *Y*](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-26.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4628)   
[ycor-vect](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-15.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1936)  
[Yochelson, Jerome C.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-33.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_5944)

[Zabih, Ramin](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-28.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4900)   
[cruces por cero de una señal](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4036) , [[2]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4040) , [[3]](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-24.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_4048)   
[prueba cero (genérica)](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2570)   
    [para polinomios](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2754)   
[Zetalisp](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-9.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_80)   
[Zilles, Stephen N.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-14.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_1428)   
[Zippel, Richard E.](https://mitp--content--server-mit-edu.translate.goog/books/content/sectbyfn/books_pres_0/6515/sicp.zip/full-text/book/book-Z-H-18.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc#%_idx_2818)