

# Algoritmos

La resolución de un problema con una computadora se hace escribiendo un programa, que exige al menos los siguientes pasos:

1. **Definición o análisis del problema.**
2. **Diseño del algoritmo.**
3. **Transformación del algoritmo en un programa.**
4. **Ejecución y validación del programa.**

Las características más sobresalientes de la resolución de problemas son:

**Análisis.** El problema se analiza teniendo presente la especificación de los requisitos dados por el cliente de la empresa o por la persona que encarga el programa.

**Diseño.** Una vez analizado el problema, se diseña una solución que conducirá a un algoritmo que resuelva el problema.

**Codificación (implementación).** La solución se escribe en la sintaxis del lenguaje de alto nivel y se obtiene un programa fuente que se compila a continuación.

**Ejecución, verificación y depuración.** El programa se ejecuta, se comprueba rigurosamente y se eliminan todos los errores (denominados “bugs”, en inglés) que puedan aparecer.

**Mantenimiento.** El programa se actualiza y modifica, cada vez que sea necesario, de modo que se cumplan todas las necesidades de cambio de sus usuarios.

**Documentación.** Escritura de las diferentes fases del ciclo de vida del software, esencialmente el análisis, diseño y codificación, unidos a manuales de usuario y de referencia, así como normas para el mantenimiento.

Las dos primeras fases conducen a un diseño detallado escrito en forma de algoritmo. Durante la tercera fase (codificación) se implementa el algoritmo en un código escrito en un lenguaje de programación, reflejando las ideas desarrolladas en las fases de análisis y diseño.

Las fases de compilación y ejecución traducen y ejecutan el programa. En las fases de verificación y depuración el programador busca errores de las etapas anteriores y los elimina. Comprobará que mientras más tiempo se gaste en la fase de análisis y diseño, menos se gastará en la depuración del programa. Por último, se debe realizar la documentación del programa.

Antes de conocer las tareas a realizar en cada fase, se considera el concepto y significado de la palabra algoritmo.

La palabra algoritmo se deriva de la traducción al latín de la palabra Al-Khōwârizmi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.

Un algoritmo es un método para resolver un problema mediante una serie de pasos precisos, definidos y finitos.

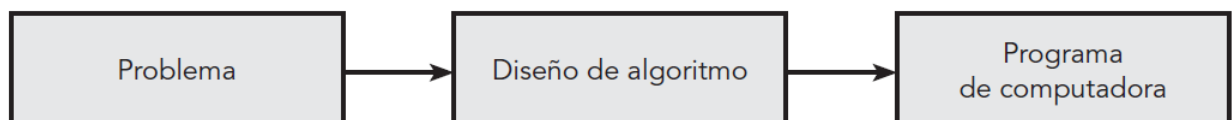
Un algoritmo es un método para resolver un problema. Aunque la popularización del término ha llegado con el advenimiento de la era informática, algoritmo proviene, como se comentó anteriormente, de Mohammed Al-Khōwārizmi, matemático persa que vivió durante el siglo IX y alcanzó gran reputación por el enunciado de las reglas paso a paso para sumar, restar, multiplicar y dividir números decimales; la traducción al latín del apellido en la palabra algorismus derivó posteriormente en algoritmo. Euclides, el gran matemático griego (del siglo IV a. C.), quien inventó un método para encontrar el máximo común divisor de dos números, se considera junto con Al-Khōwārizmi el otro gran padre de la algoritmia (ciencia que trata de los algoritmos).

El profesor Niklaus Wirth, inventor de Pascal, Modula-2 y Oberon, tituló uno de sus más famosos libros, Algoritmos + Estructuras de datos = Programas, significándonos que solo se puede llegar a realizar un buen programa con el diseño de un algoritmo y una correcta estructura de datos.

La resolución de un problema exige el diseño de un algoritmo que resuelva el problema propuesto.

Los pasos para la resolución de un problema son:

- 1. Diseño del algoritmo, que describe la secuencia ordenada de pasos, sin ambigüedades, que conducen a la solución de un problema dado. (Análisis del problema y desarrollo del algoritmo.)**
- 2. Expresar el algoritmo como un programa en un lenguaje de programación adecuado. (Fase de codificación.)**
- 3. Ejecución y validación del programa por la computadora.**



Para llegar a la realización de un programa es necesario el diseño previo de un algoritmo, de modo que sin algoritmo no puede existir un programa.

Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta. En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el algoritmo será siempre el mismo. Así, por ejemplo, en una analogía con la vida diaria, una receta de un plato de cocina se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos para la elaboración del plato se realizarán sin importar el idioma del cocinero.

En la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es tan solo un medio para expresar un algoritmo y una computadora es solo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente.

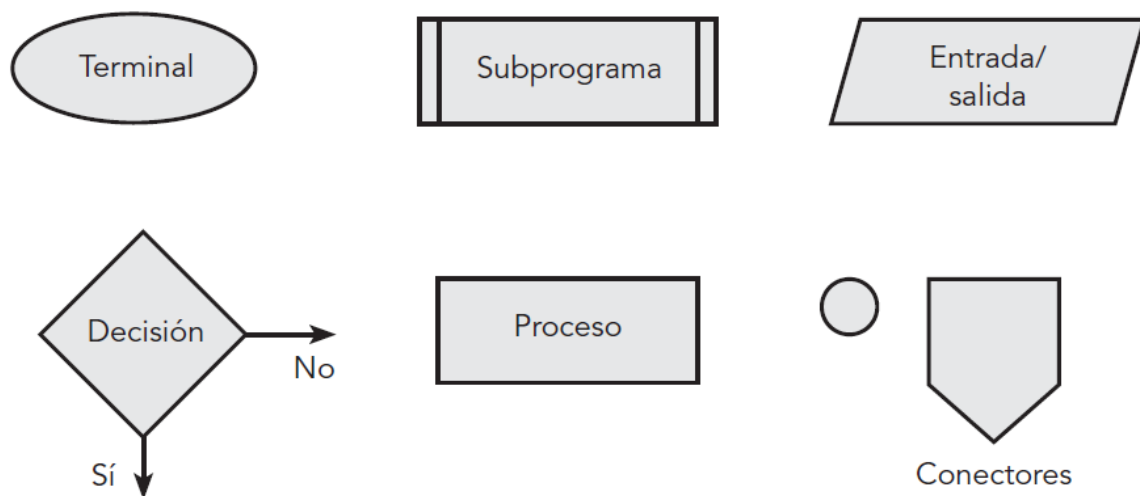
## Herramientas gráficas y alfanuméricas

Las dos herramientas más utilizadas comúnmente para diseñar algoritmos son: diagramas de flujo y pseudocódigos.

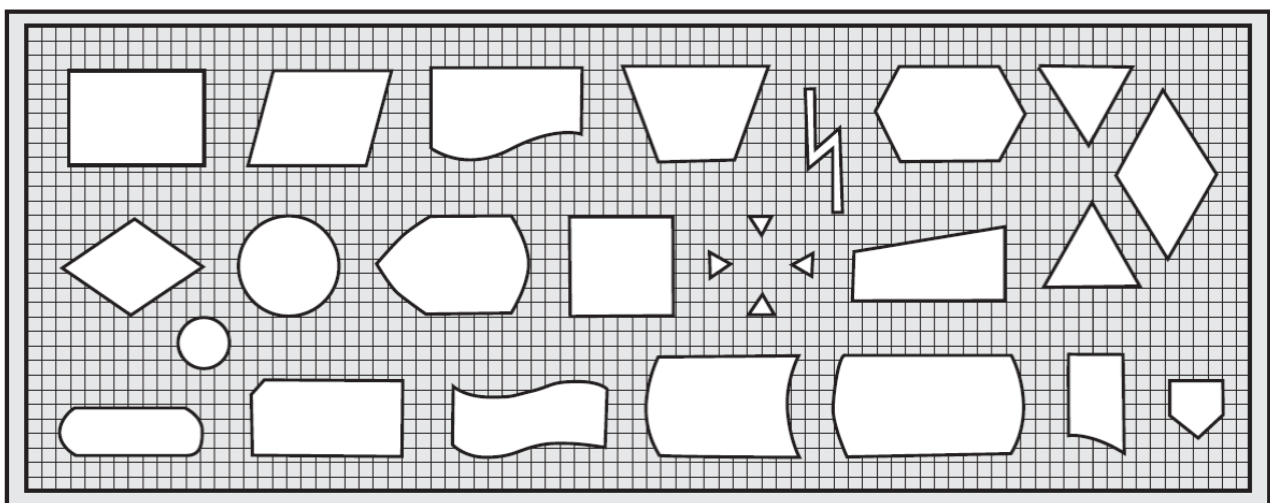
Un **diagrama de flujo** (flowchart) es una representación gráfica de un algoritmo. Los símbolos utilizados han sido normalizados por el Instituto Norteamericano de Normalización (ANSI), y los más frecuentemente empleados se muestran en las siguientes figuras, junto con una plantilla utilizada para el dibujo de los diagramas de flujo.

El **pseudocódigo** es una herramienta de programación en la que las instrucciones se escriben en palabras similares al inglés o español, que facilitan tanto la escritura como la lectura de programas. En esencia, el pseudocódigo se puede definir como un lenguaje de especificaciones de algoritmos.

Aunque no existen reglas para escritura del pseudocódigo en español, se ha recogido una notación estándar que se utilizará en el libro y que ya es muy empleada en los libros de programación en español. Las palabras reservadas básicas se representarán en letras negritas minúsculas. Estas palabras son traducción libre de palabras reservadas de lenguajes como C. Más adelante se indicarán los pseudocódigos fundamentales para utilizar en esta obra.



## Características de los algoritmos



Las características fundamentales que debe cumplir todo algoritmo son:

- Debe ser preciso.
- Debe tener un orden:
  - Entrada: Ingredientes y utensilios empleados.
  - Proceso: Elaboración de la receta en la cocina.
  - Salida: Terminación del plato (por ejemplo, cordero).

### Ejemplo:

Un cliente ejecuta un pedido a una fábrica. La fábrica examina en su banco de datos la ficha del cliente, si el cliente es solvente entonces la empresa acepta el pedido; en caso contrario, rechazará.

Los pasos del algoritmo son:

1. Inicio.
2. Leer el pedido.
3. Examinar la ficha del cliente.
4. Si el cliente es solvente, aceptar pedido; en caso contrario, rechazar pedido.
5. Fin.

### Ejemplo:

Se desea diseñar un algoritmo para saber si un número es primo o no. Un número es primo si solo puede dividirse entre sí mismo y entre la unidad (es decir, no tiene más divisores que él mismo y la unidad). Por ejemplo, 9, 8, 6, 4, 12, 16, 20, etc., no son primos, ya que son divisibles entre números distintos a ellos mismos y a la unidad. Así, 9 es divisible entre 3, 8 lo es entre 2, etc. El algoritmo de resolución del problema pasa por dividir sucesivamente el número entre 2, 3, 4..., etcétera.

1. Inicio.
2. Poner X igual a 2 ( $X \leftarrow 2$ , X variable que representa a los divisores del número que se busca N).
3. Dividir N entre X ( $N/X$ ).
4. Si el resultado de  $N/X$  es entero, entonces N no es número primo y se bifurca al punto 7; en caso contrario continuar el proceso.
5. Suma 1 a X ( $X \leftarrow X + 1$ ).
6. Si X es igual a N, entonces N es primo; en caso contrario bifurcar al punto 3.
7. Fin.

Por ejemplo, si N es 131, los pasos anteriores serían:

1. Inicio.

2.  $X = 2$ .
3.  $131/X$ . Como el resultado no es entero, se continúa el proceso.
4.  $X \leftarrow 2 + 1$ , luego  $X = 3$ .
5. Como  $X$  no es 131, se continúa el proceso.
6.  $131/X$  resultado no es entero.
7.  $X \leftarrow 3 + 1$ ,  $X = 4$ .
8. Como  $X$  no es 131 se continúa el proceso.
9.  $131/X$ . . ., etc.
10. Fin.

### Ejemplo:

Realizar la suma de todos los números pares entre el 2 y el 1000. El problema consiste en sumar  $2 + 4 + 6 + 8 \dots + 1\ 000$ . Utilizaremos las palabras SUMA y NÚMERO (variables, serán denominadas más tarde) para representar las sumas sucesivas ( $2 + 4$ ), ( $2 + 4 + 6$ ), ( $2 + 4 + 6 + 8$ ), etc. La solución se puede escribir con el siguiente algoritmo:

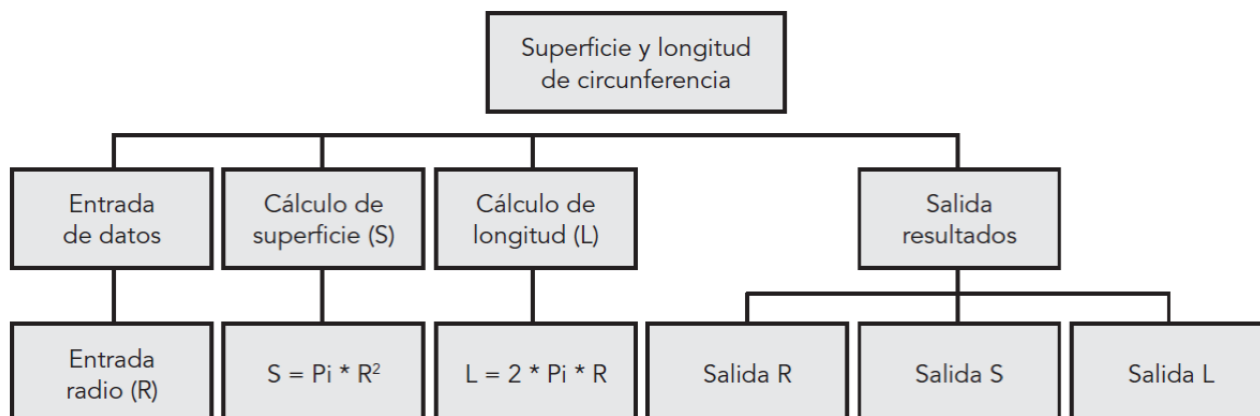
1. Inicio.
2. Establecer SUMA a 0.
3. Establecer NÚMERO a 2.
4. Sumar NÚMERO a SUMA. El resultado será el nuevo valor de la suma (SUMA).
5. Incrementar NÚMERO en 2 unidades.
6. Si  $NÚMERO \leq 1000$  bifurcar al paso 4; en caso contrario, escribir el último valor de SUMA y terminar el proceso.
7. Fin.

## Diseño de algoritmos

Una computadora no tiene capacidad para solucionar problemas mas que cuando se le proporcionan los sucesivos pasos a realizar. Estos pasos sucesivos que indican las instrucciones a ejecutar por la máquina constituyen, como ya conocemos, el algoritmo.

La información proporcionada al algoritmo constituye su entrada y la información producida por el algoritmo constituye su salida.

Los problemas complejos se pueden resolver más eficazmente con la computadora cuando se dividen en subproblemas que sean más fáciles de solucionar que el original. Es el método de divide y vencerás (*divide and conquer*), mencionado anteriormente, y que consiste en dividir un problema complejo en otros más simples. Así, el problema de encontrar la superficie y la longitud de un círculo se puede dividir en tres problemas más simples o subproblemas.



La descomposición del problema original en subproblemas más simples y a continuación la división de estos subproblemas en otros más simples que pueden ser implementados para su solución en la computadora se denomina diseño descendente (top-down design). Normalmente, los pasos diseñados en el primer esbozo del algoritmo son incompletos e indicarán solo unos pocos pasos (un máximo de doce pasos aproximadamente). Tras esta primera descripción, estos se amplían en una descripción más detallada con más pasos específicos. Este proceso se denomina refinamiento del algoritmo (*stepwise refinement*). Para problemas complejos se necesitan con frecuencia diferentes niveles de refinamiento antes de que se pueda obtener un algoritmo claro, preciso y completo.

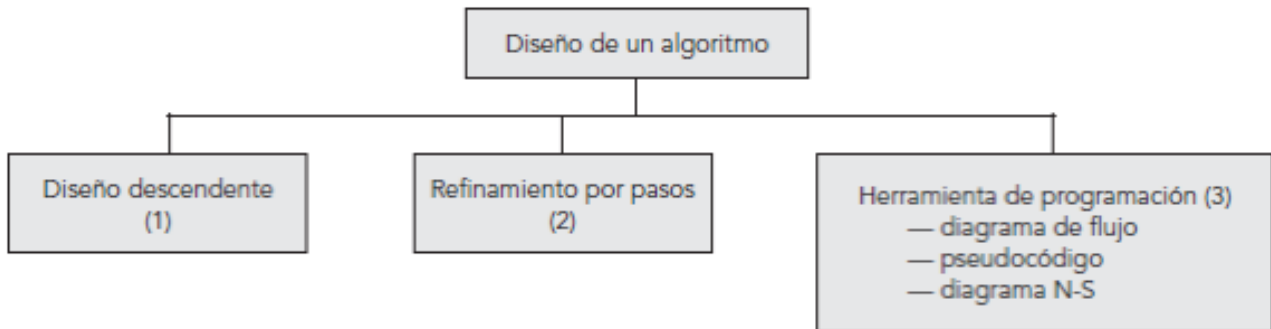
El problema de cálculo de la circunferencia y superficie de un círculo se puede descomponer en subproblemas más simples: 1) leer datos de entrada; 2) calcular superficie y longitud de circunferencia, y 3) escribir resultados (datos de salida).

Subproblema	Refinamiento
<i>leer</i> radio calcular superficie calcular circunferencia <i>escribir</i> resultados	leer radio $\text{superficie} = 3.141592 * \text{radio}^2$ $\text{circunferencia} = 2 * 3.141592 * \text{radio}$ <i>escribir</i> radio, circunferencia, superficie




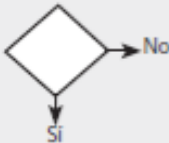










Las ventajas más importantes del diseño descendente son:

- El problema se comprende más fácilmente al dividirse en partes más simples denominadas módulos.
- Las modificaciones en los módulos son más fáciles.
- La comprobación del problema se puede verificar fácilmente.

Tras los pasos anteriores (diseño descendente y refinamiento por pasos) es preciso representar el algoritmo mediante una determinada herramienta de programación: diagrama de flujo, pseudocódigo o diagrama N-S. Así pues, el diseño del algoritmo se descompone en las fases



recogidas en la figura.

Símbolos principales	Función
	Terminal (representa el comienzo, "inicio", y el final, "fin" de un programa. Puede representar también una parada o interrupción programada que sea necesario realizar en un programa).
	Entrada/Salida (cualquier tipo de introducción de datos en la memoria desde los periféricos, "entrada", o registro de la información procesada en un periférico, "salida").
	Proceso (cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transferencia, etcétera).
	Decisión (indica operaciones lógicas o de comparación entre datos, normalmente dos, y en función del resultado de la misma determina cuál de los distintos caminos alternativos del programa se debe seguir; normalmente tiene dos salidas, respuestas SÍ o NO).
	Decisión múltiple (en función del resultado de la comparación, se seguirá uno de los diferentes caminos de acuerdo con dicho resultado).
	Conector (sirve para enlazar dos partes cualesquiera de un organigrama a través de un conector en la salida y otro conector en la entrada). Se refiere a la conexión en la misma página del diagrama.
	Indicador de dirección o línea de flujo (indica el sentido de ejecución de las operaciones).
	Línea conectora (sirve de unión entre dos símbolos).
	Conector (conexión entre dos puntos del organigrama situado en páginas diferentes).
	Llamada a subrutina o a un proceso predeterminado (una subrutina es un módulo independientemente del programa principal, que recibe una entrada procedente de dicho programa, realiza una tarea determinada y regresa, al terminar, al programa principal).
	Pantalla (se utiliza en ocasiones en lugar del símbolo de E/S).
	Impresora (se utiliza en ocasiones en lugar del símbolo de E/S).
	Teclado (se utiliza en ocasiones en lugar del símbolo de E/S).
	Comentarios (se utiliza para añadir comentarios clasificadores a otros símbolos del diagrama de flujo. Se pueden dibujar a cualquier lado del símbolo).



## Ejemplo:

Calcular la media de una serie de números positivos, suponiendo que los datos se leen desde un terminal. Un valor de cero, como entrada, indicará que se ha alcanzado el final de la serie de números positivos.

El primer paso a dar en el desarrollo del algoritmo es descomponer el problema en una serie de pasos secuenciales. Para calcular una media se necesita sumar y contar los valores. Por consiguiente, nuestro algoritmo en forma descriptiva sería:

1. Inicializar contador de números C y variable sumas.
2. Leer un número.
3. Si el número leído es cero:
  - Calcular la media
  - Imprimir la media
  - Calcular la suma
  - Incrementar en uno el contador de números
  - Ir al paso 2
4. Fin.

## Pseudocódigo

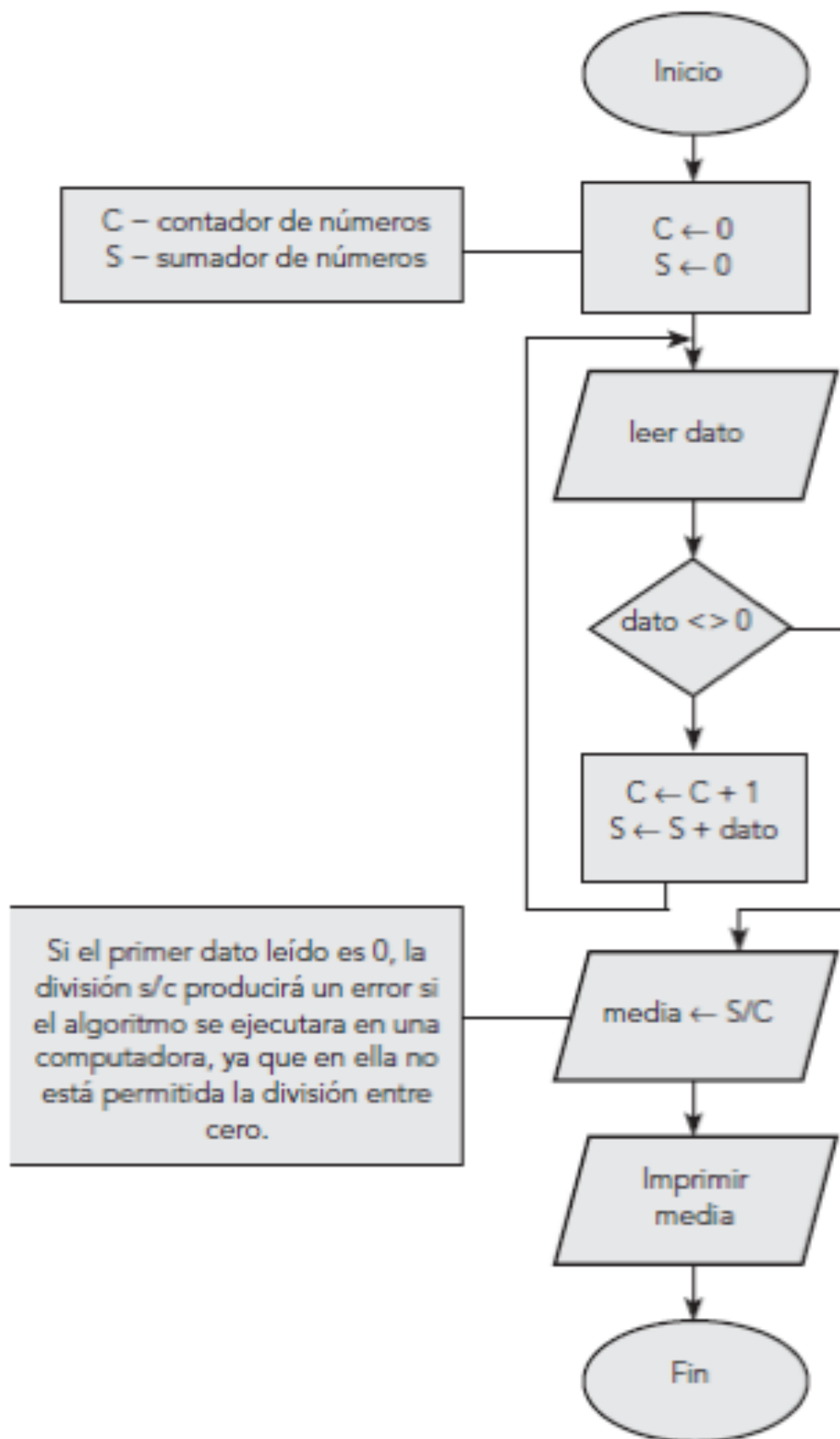
```
entero: dato, C
real: Media, S
C ← 0
S ← 0
escribir ('Datos numéricos;
para finalizar se introduce 0')
repetir
    leer(dato)
    si dato <> = 0 entonces
        C ← C + 1
        S ← S + dato
    fin si
hasta dato = 0
{Calcula la media y la escribe}
si (C > 0) entonces
```

Media  $\leftarrow S/C$

escribir (Media)

fin si

## Diagrama de flujo



Ejemplo:

Suma de los números pares comprendidos entre 2 y 100.

### **Pseudocódigo**

```
entero: numero, Suma
Suma ← 2
numero ← 4
mientras (numero <= 100) hacer
    suma ← suma + numero
    numero ← numero + 2
fin mientras
escribe ('Suma pares entre 2 y 100 =', suma)
```

