

Universidad Nacional de Quilmes

Elementos de Programación y Lógica

Cuadernillo del estudiante

por Alan Rodas Bonjour

Ver. 0.6.0 (24/04/2022)

Copyright © 2019 Alan Rodas Bonjour

PUBLICADO ONLINE POR ALAN RODAS BONJOUR

HTTP://ELEMENTOSDEPROGRAMACIONYLOGICA.WEB.UNQ.EDU.AR



Licenciado bajo los términos de la siguiente licencia de contenidos: Licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional (la “licencia”). Puede obtener una copia completa de ella en <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>.

Usted tiene derecho a:

- **Compartir:** copiar y redistribuir el material en cualquier medio o formato
- **Adaptar:** remezclar, transformar y construir a partir del material

El licenciatario no puede revocar estas libertades en tanto usted siga los términos siguientes.

- **Atribución:** Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciatario.
- **NoComercial:** Usted no puede hacer uso del material con propósitos comerciales.
- **CompartirIgual:** Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original.

No hay restricciones adicionales — No puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otras a hacer cualquier uso permitido por la licencia.

No tiene que cumplir con la licencia para elementos del material en el dominio público o cuando su uso esté permitido por una excepción o limitación aplicable.

No se dan garantías. La licencia podría no darle todos los permisos que necesita para el uso que tenga previsto. Por ejemplo, otros derechos como publicidad, privacidad, o derechos morales pueden limitar la forma en que utilice el material.

Primera edición, Febrero 2019



Índice general

I

Computadoras

| | | |
|----------|---|-----------|
| 1 | Computadoras | 15 |
| 1.1 | Hardware (parte física) | 16 |
| 1.1.1 | Periféricos | 17 |
| 1.1.2 | Redes | 20 |
| 1.1.3 | Relación del hardware y el software | 22 |
| 1.2 | Software (parte lógica) | 22 |
| 1.2.1 | Firmware | 23 |
| 1.2.2 | Sistemas Operativos | 24 |
| 1.2.3 | Programas | 26 |
| 1.2.4 | Archivos y Directorios | 27 |
| 1.2.5 | Otros elementos de software | 29 |
| 1.3 | Actividades | 29 |
| 2 | Historia de las computadoras | 31 |
| 2.1 | Los precursores de las computadoras modernas | 31 |
| 2.1.1 | La prehistoria de la computación | 32 |
| 2.1.2 | Las computadoras humanas | 33 |
| 2.1.3 | Las calculadoras mecánicas | 34 |
| 2.1.4 | La primer computadora mecánica | 34 |
| 2.2 | El despertar de la computación | 35 |
| 2.2.1 | La computación como teoría | 35 |
| 2.2.2 | Primeras computadoras electromecánicas | 37 |
| 2.2.3 | IBM, Bell y grandes computadoras | 39 |
| 2.2.4 | Los lenguajes de programación | 40 |

| | | |
|------------|--|-----------|
| 2.2.5 | La revolución de los transistores y los circuitos integrados | 42 |
| 2.3 | La modernidad de las computadoras | 43 |
| 2.3.1 | Las primeras microcomputadoras | 43 |
| 2.3.2 | Los primeros sistemas operativos | 44 |
| 2.3.3 | El Software Libre | 45 |
| 2.3.4 | Las redes, Internet, y la globalización | 47 |
| 2.4 | La actualidad y el futuro | 48 |
| 2.4.1 | La internet de las cosas y Big Data | 48 |
| 2.4.2 | La inteligencia artificial y la automatización | 50 |
| 2.4.3 | La computación cuántica y la computación biológica | 50 |
| 2.5 | Actividades | 51 |

| | | |
|------------|--|-----------|
| II | Información | |
| 3 | Bajo nivel | 57 |
| 3.1 | Representación de Información | 57 |
| 3.1.1 | Códigos y comunicación | 57 |
| 3.1.2 | Electricidad y cables | 59 |
| 3.1.3 | Código Binario | 61 |
| 3.1.4 | Codificación de información | 62 |
| 3.1.5 | Representando números con binario | 65 |
| 3.1.6 | Bits, Bytes y otros | 66 |
| 3.2 | Sistemas de Caja Negra | 67 |
| 3.3 | Actividades | 68 |
| 4 | Informática | 71 |
| 4.1 | Archivos y extensiones | 71 |
| 4.1.1 | Archivos informáticos | 71 |
| 4.1.2 | Extensiones de archivos | 73 |
| 4.2 | Organización de archivos, directorios y rutas | 75 |
| 4.2.1 | Rutas absolutas | 76 |
| 4.2.2 | Rutas relativas | 78 |
| 4.2.3 | Universal Resource Identifier | 80 |
| 4.2.4 | Programas, Editores y Visualizadores | 81 |
| 4.3 | Programas y Lenguajes | 83 |
| 4.3.1 | Lenguajes | 83 |
| 4.3.2 | Lenguajes de programación | 84 |
| 4.3.3 | Lenguajes de marcado | 86 |
| 4.4 | Modelos de comercialización de software | 90 |
| 4.4.1 | Copyright | 90 |
| 4.4.2 | Software Libre | 92 |
| 4.4.3 | Open Source | 94 |
| 4.4.4 | Demistificación de frases sobre el FOSS | 95 |
| 4.5 | Actividades | 96 |

| | | |
|------------|---|------------|
| 5 | Introducción a la lógica | 103 |
| 5.1 | Aproximación a la lógica | 104 |
| 5.1.1 | Historia de la lógica | 104 |
| 5.1.2 | El objeto de estudio de la lógica | 105 |
| 5.2 | Elementos constituyentes de la lógica | 107 |
| 5.2.1 | Valores de verdad | 107 |
| 5.3 | Conectivas | 108 |
| 5.3.1 | Conjunción | 111 |
| 5.3.2 | Disyunción | 112 |
| 5.3.3 | Negación | 112 |
| 5.4 | Utilidad de la lógica en la informática | 113 |
| 5.5 | La lógica como formuladora de preguntas | 113 |
| 5.6 | Precedencia de conectivas | 114 |
| 5.7 | Analisis de expresiones | 115 |
| 5.7.1 | Ánalisis mediante semántica | 116 |
| 5.7.2 | Tablas de verdad | 118 |
| 5.8 | Resumen | 121 |
| 5.9 | Actividades | 122 |
| 6 | Lógica proposicional | 127 |
| 6.1 | Proposiciones, atómicas y compuestas | 127 |
| 6.2 | Pasaje de texto a formulas | 128 |
| 6.2.1 | Pasaje a fórmulas de proposiciones atómicas | 128 |
| 6.2.2 | Pasaje a fórmulas de proposiciones moleculares | 129 |
| 6.3 | Razonamientos | 131 |
| 6.3.1 | Indicadores de conclusión: | 135 |
| 6.3.2 | Indicadores de premisa: | 135 |
| 6.4 | Otras conectivas más complejas | 136 |
| 6.4.1 | Disyunción excluyente | 136 |
| 6.4.2 | Implicación | 137 |
| 6.4.3 | Equivalencia | 137 |
| 6.5 | Formulas bien formadas | 138 |
| 6.5.1 | Orden de precedencia | 140 |
| 6.6 | Analisis de validez de razonamientos por tabla de verdad | 140 |
| 6.7 | Relación entre las premisas y la conclusión | 144 |
| 6.8 | Álgebra de Boole | 145 |
| 6.8.1 | Equivalencia de fórmulas | 145 |
| 6.8.2 | Axiomas y Leyes del Álgebra de Boole | 146 |
| 6.9 | Reglas de inferencia | 147 |
| 6.9.1 | Modus ponendo ponens | 148 |
| 6.9.2 | Modus tollendo tollens | 148 |
| 6.9.3 | Modus ponendo tollens | 149 |

| | | |
|-------------|---|------------|
| 6.9.4 | Modus tollendo ponens | 149 |
| 6.10 | Falacias bien conocidas | 150 |
| 6.10.1 | Afirmación del consecuente | 150 |
| 6.10.2 | Negación del antecedente | 150 |
| 6.11 | Límites de la lógica proposicional | 151 |
| 6.12 | Actividades | 152 |
| 7 | Lógica de predicados | 157 |
| 7.1 | Individuos, Propiedades y Relaciones | 157 |
| 7.1.1 | Individuos | 158 |
| 7.1.2 | Propiedades | 158 |
| 7.2 | Predicados | 159 |
| 7.2.1 | Aridad | 159 |
| 7.2.2 | Predicados y conectivas | 160 |
| 7.3 | Universo de discurso | 160 |
| 7.4 | Cuantificadores | 160 |
| 7.4.1 | Cuantificador universal | 160 |
| 7.4.2 | Cuantificador existencial | 160 |
| 7.4.3 | Cuantificador existencial negado | 160 |
| 7.4.4 | Equivalencia de cuantificadores | 160 |
| 7.5 | Actividades | 160 |

IV

Programación

V

Índice y Anexos

| | | |
|---------------|--|------------|
| Índice | 169 | |
| Anexos | | |
| A | Markdown | 177 |
| A.1 | Sintaxis y Semántica | 178 |
| A.2 | Actividades | 183 |
| B | HTML | 185 |
| B.1 | Browsers | 185 |
| B.2 | W3C y estándares de la web | 186 |
| B.3 | Sistema de etiquetas como marcas | 187 |
| B.4 | Etiquetas básicas de HTML | 189 |
| B.5 | Documentos HTML válidos | 196 |
| B.6 | CSS | 199 |
| B.7 | Formularios y respuestas del servidor | 201 |
| B.8 | Actividades | 205 |

| | | |
|----------|--|------------|
| C | Simbología Lógica y Tablas de Verdad | 213 |
| C.1 | Simbología de las conectivas y sus palabras asociadas en lenguaje natural en la lógica proposicional | 214 |
| C.2 | Simbología de la lógica de predicados | 215 |
| C.3 | Tablas de verdad de conectivas | 216 |



Introducción

Imagen de la supercomputadora Thunderbird, en el Sandia National Laboratory..
Fotografía del Departamento de Energía de los Estados Unidos.

La **informática** es la disciplina o campo de estudio que abarca el **conjunto de conocimientos, métodos y técnicas referentes al tratamiento automático de la información, junto con sus teorías y aplicaciones prácticas, con el fin de almacenar, procesar y transmitir datos e información en formato digital utilizando sistemas computacionales**. Los datos son la materia prima para que, mediante su proceso, se obtenga como resultado información. Para ello, la informática crea y/o emplea sistemas de procesamiento de datos, que incluyen medios físicos en interacción con medios lógicos y las personas que los programan y los usan.

Estructura del libro

El presente se estructura en cuatro unidades y una unidad adicional que contiene el índice y una serie de anexos. Cada unidad a su vez, se divide en capítulos, y cada capítulo en diferentes secciones.

Cada capítulo del presente se apoya en lo visto en capítulos anteriores, y si bien pueden leerse de forma independiente, esto trae aparejado el supuesto de ciertos conocimientos ya vistos, por lo que recomendamos leer el libro de forma completa y continuada.

El texto intenta ser lo más breve y directo posible, para permitir la comprensión incluso a los menos asiduos a la lectura. Por otro lado, se intenta explicar los conceptos con definiciones más pragmáticas que enciclopédicas. Si bien esto puede no ser del agrado de todos los lectores, creemos que la comprensión del concepto es más importante que una definición, la cual, en general, varía de autor en autor. Si bien en algunos lugares se sacrifica precisión en pos de la comprensión, en todo lugar donde esto ocurre es deliberado, pues no se desea sobrecargar el libro en conceptos que, si bien importantes para un profesional de la industria, no son relevantes para quien busca adquirir conocimientos generales de la temática o recién está iniciando su camino como profesional.

Al final de cada capítulo se presentan ejercicios prácticos a realizar, los cuales tienen dos propósitos. Por un lado, muchos de ellos sirven para asentar los conocimientos teóricos vistos

utilizando un enfoque práctico. Por otro, algunos de ellos ocultan detalles teóricos interesantes que no pudieron entrar en estas páginas por motivos de espacio.

Los ejercicios suelen tomar desde unos pocos minutos a una hora en resolverse, y permiten validar al lector si ha adquirido los conocimientos tratados en el capítulo, así como expandir los saberes presentados en el mismo. Recomendamos que se vayan realizando a medida que se avanza con la lectura.

Al final de cada unidad se presenta un conjunto de bibliografía adicional que permitirá al lector expandir sus conocimientos o abordar con mayor profundidad las diversas temáticas encaradas aquí, así como también tener otros puntos de vista de diferentes autores.

La última parte del libro contiene el índice con las palabras clave, y los lugares en donde dichas palabras aparecen en el libro de forma relevante. También posee una serie de anexos, los cuales presentan información adicional sobre una temática particular, o resultan útiles de tener a mano a la hora de realizar las actividades propuestas.

Temas a tratar

Como este campo requiere el uso de **computadoras** (en el sentido amplio de la palabra), comenzaremos en la Unidad I por analizar qué es una computadora, ver como funcionan, de que partes están compuestas y una breve historia de las mismas. Se debe tener en cuenta que nos centraremos en computadoras electrónicas, también llamadas clásicas, las cuales son las que predominan en el mercado. Sin embargo, analizaremos otras variantes de computadoras cuando veamos la historia de las mismas, como las computadoras mecánicas. Asimismo, los conceptos teóricos que rigen las ciencias de la computación, auguran como posibilidades otros modelos computacionales, todavía en etapas muy inmaduras, como las computadoras cuánticas o las biológicas, tema que tocaremos brevemente. Este capítulo busca brindar al lector una introducción amena a cuáles son los elementos de estudio pertinentes, entender cómo llegaron a existir, su importancia en nuestra vida cotidiana actual, y también tener una mínima visión sobre su posible futuro. El enfoque de esta unidad es más bien informativo, y las actividades finales solo expanden sobre diversos elementos de la historia y conocimientos generales.

En la Unidad II analizaremos como las computadoras almacenan **información**, y como se procesa y se interpreta la misma. Haremos una mínima introducción a lo que es el concepto de datos binarios, y veremos como con ese sistema se almacenan números, letras, colores, imágenes, videos y cualquier información que uno desee. Veremos como funcionan los editores de texto y los visualizadores de archivos, y tendremos un primer acercamiento a un lenguaje informático mediante el uso de lenguajes de marcado. Evitaremos mencionar los estándares que rigen las computadoras modernas, pues solamente nos interesaría el concepto subyacente a los mismos, por lo que algunos lectores que posean conocimiento de esta temática podrán sentir que se ha tomado una aproximación muy laxa, pero recuerde el lector que el objetivo final de esta sección es dar una idea mínima sobre el funcionamiento de las computadoras modernas, para que el mismo no resulte “mágico”. Si bien el enfoque de esta unidad es también informativa, las actividades finales que se proponen implican reforzar los elementos conceptuales mencionados para un mayor entendimiento.

Posteriormente en la Unidad III analizaremos la **lógica**, uno los fundamentos teóricos más importantes que subyace en la disciplina. Analizaremos parte de los formalismos que rigen en esta ciencia y veremos como se aplican esos conceptos en matemática y en ciencias de la computación en general. El foco del presente no es realizar un análisis exhaustivo de esta disciplina, la cual es sumamente amplia para abarcarla en tan pocas páginas, sino brindar los conocimientos elementales que permitirán al lector aplicar estos conceptos a diferentes áreas, como matemáticas, programación

e incluso en su día a día. Aparecerán entonces los conceptos de conectivas, y las utilizaremos para formular preguntas nuevas a partir de otras que ya teníamos disponibles, un concepto muy útil a la hora de programar. También hablaremos de razonamientos deductivos, de sus premisas y conclusión, cuándo son válidos y cuándo no, y aplicaremos los conocimientos de forma práctica para introducirnos en la lógica proposicional y descubrir sus formalismos. Finalmente, veremos razonamientos que no pueden resolverse con la lógica proposicional y mojaremos nuestros pies en la superficie de la lógica de predicados, analizando sus elementos, aunque no trabajaremos con el análisis de la validez de dichos razonamientos. Esta unidad es más bien teórico-práctica, ya que, de forma similar a la matemática, si bien se pueden comprender los conceptos de la lógica fácilmente, su aplicación posterior resulta compleja si no se ha expuesto uno a varias actividades de la temática. Así, se recomienda realizar cada uno de los ejercicios propuestos en cada unidad, ya que su realización es fundamental para un entendimiento adecuado de la temática.

Por último, en la Unidad IV veremos como se procesa información utilizando **programación**. Crearemos nuestros primeros programas que solucionarán pequeños problemas, y analizaremos los mismos para descubrir algunas de las estructuras más comúnmente utilizadas en programación. Nuevamente, la disciplina es amplia, y por tanto solo abordaremos actividades muy simples, que incluyen temas sencillos, como la secuencia de comandos para generar programas, el uso de estructuras de control para estructurar ideas y reducir la cantidad de código, así como generar soluciones más genéricas. Por sobre todo trabajaremos en planificar nuestras soluciones en forma de tareas pequeñas para simplificar la complejidad del problema a resolver, así como comunicar correcta y eficazmente nuestra solución. Los lectores que tengan conocimientos previos en esta área sentirán al comienzo que lo presentado es muy básico, y que aporta poco a su saber. Sin embargo, recomendamos no tomarse a la ligera lo presentado en esta unidad, pues el abordaje elegido difiere del seleccionado por muchos autores, y hace especial hincapié en la comunicación de la solución al problema, concepto fundamental para convertirse en un programador profesional. Esta unidad tiene un enfoque bastante distinto a las anteriores, en donde se propone al lector realizar actividades a medida que se avanza en la lectura, para luego analizar diversas soluciones al problema, y a partir de ahí generar el conocimiento. Es recomendable que el lector encare dichas actividades en el momento sugerido, poniendo en pausa la lectura hasta terminar las actividades propuestas, y retomando solo después de haber pensado la solución y haberla finalizado. Las actividades propuestas deben realizarse en computadora, tablet o teléfono inteligente (aunque en este último la experiencia no es la mejor), por lo que se recomienda que el lector cuente con acceso a uno de estos dispositivos durante la lectura de esta última unidad, aunque sea de forma esporádica. Adicionalmente elaboraremos una sintaxis para escribir programas en papel, algo que permitirá realizar las actividades incluso si no se tiene acceso a computadora, y que además será útil para una mejor comprensión de los conceptos trabajados, que se reforzarán al final de cada capítulo con mayor cantidad de actividades.

Requisitos y saberes previos

Para poder comprender los contenidos de este libro no se requieren demasiados saberes previos. Se espera sin embargo, que el lector posea algunos conceptos básicos de matemática, que incluyen aritmética elemental y algo de geometría.

Saberes elementales de la lengua también son útiles, como poder comprender textos, analizar sujeto y predicado, identificar verbos, y poder determinar donde hay un sujeto tácito o se hace referencia a un sujeto antes mencionado.

También se espera que el lector sepa utilizar una computadora, o se haya cruzado en algún momento con una. Si bien existen diversos tipos de computadora, y diversas formas de interactuar

con las mismas, el presente toma por supuesto que el lector ha utilizado una computadora de escritorio o notebook, y que sabe lo que es un teléfono móvil inteligente (smartphone), haya o no interactuado con uno.

La Unidad III, en su última sección, se vuelve más sencilla de comprender si el lector se cruzó previamente con el concepto de función (en términos de análisis matemático), aunque no es imprescindible para la correcta comprensión de los temas abordados.

Para seguir la teoría de la Unidad IV, así como resolver los ejercicios propuestos puede recurrir solamente a la lectura, lápiz y papel. Sin embargo, realizar los ejercicios en la computadora facilitará una mayor comprensión de los conceptos, simplificando el proceso de aprendizaje. Para esto se espera que lector cuente con acceso a una computadora, ya sea una notebook, máquina de escritorio, tablet o incluso un teléfono inteligente y acceso a internet (ya sea continuo o esporádico).

Comentarios finales

La disciplina de las ciencias de la computación y la informática en general son campos relativamente nuevos (aunque sus orígenes datan de más de tres milenios atrás, no se comenzó a considerar la informática como una ciencia en sí misma sino hasta comienzos del siglo XX). Aún así, esta disciplina ha evolucionado mucho en un muy corto período de tiempo, revolucionando de forma sustancial el mundo en el que vivimos.

Si bien las computadoras son omnipresentes en nuestro día a día, todo indica que recién estamos nadando en la superficie de las aguas de lo que es posible con la informática, y que el futuro nos depara una inmersión en las mismas, a profundidades que aún no alcanzamos a vislumbrar. Las posibilidades auguradas por las ciencias de la computación son muchas y muy amplias, y es un campo en completa expansión, que trabaja cada vez más de forma interdisciplinaria con ciencias tan dispares como la física, la biología, la neurología, la lingüística, la economía, la política, e incluso la filosofía.

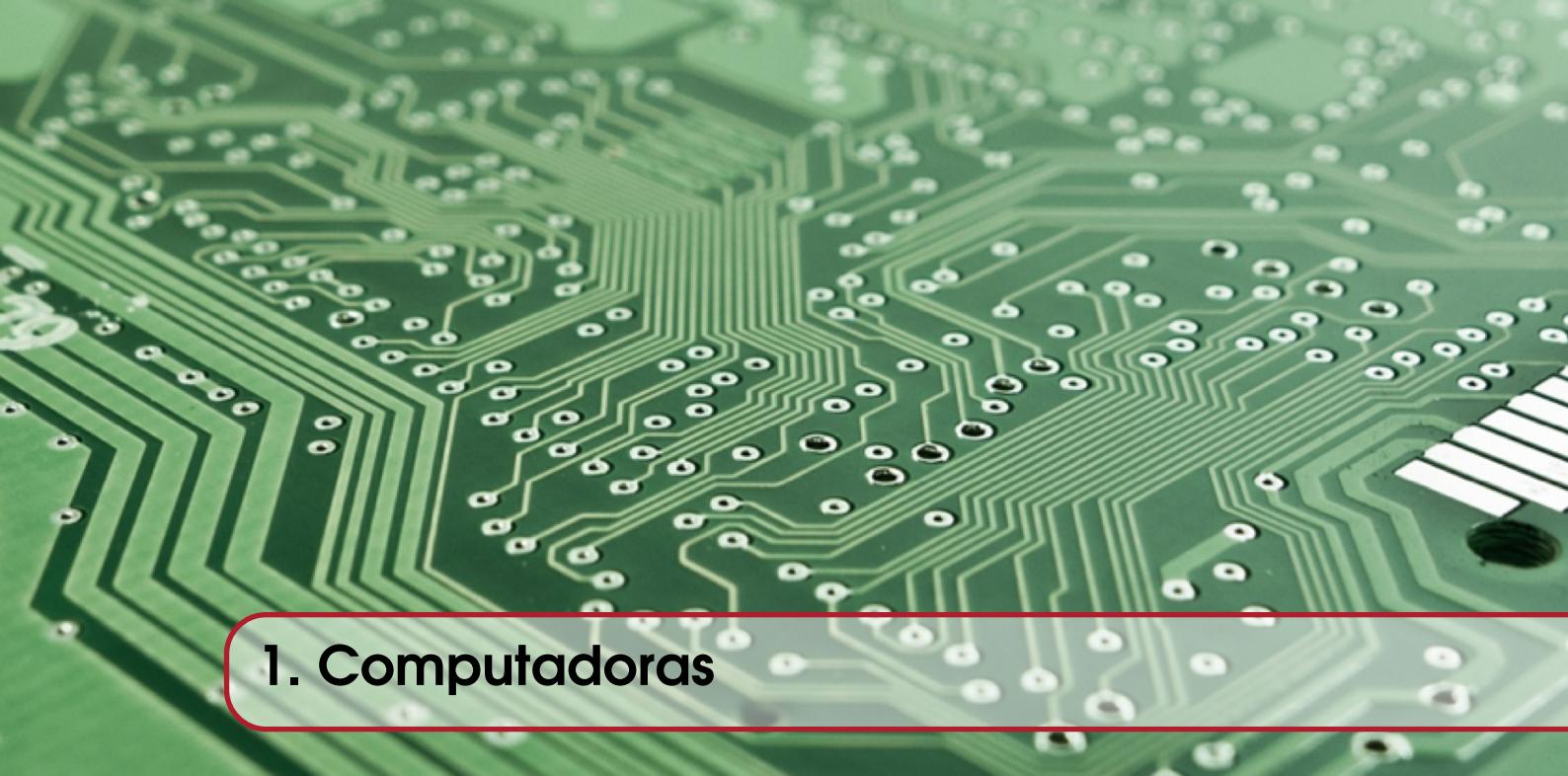
Por otro lado, como ya mencionamos, la informática tiene que ver con el proceso y manejo de la información. Ser meros usuarios de tecnologías, desconociendo completamente los conceptos subyacentes más básicos de los mismos, nos vuelve vulnerables a engaños y manipulaciones por quienes controlan la información. ¿Es bueno el voto electrónico? ¿Qué datos tienen las empresas sobre mi persona y cómo pueden utilizarlos? ¿Es seguro utilizar una tarjeta de crédito en internet? Si conocemos los conceptos subyacentes a la tecnología que utilizamos, responder esas preguntas se vuelve más sencillo, y nos permite tomar mejores decisiones como ciudadanos digitales.

Muchos especialistas afirman que dentro de pocos años en el futuro, desconocer estos conceptos será equivalente a no saber leer y escribir en la sociedad actual. Por eso este libro apunta a brindar un panorama introductorio, pero lo suficientemente amplio como para que el lector obtenga un panorama general de los conceptos principales de la informática generalmente desconocidos.



Computadoras

| | | |
|----------|---|-----------|
| 1 | Computadoras | 15 |
| 1.1 | Hardware (parte física) | |
| 1.2 | Software (parte lógica) | |
| 1.3 | Actividades | |
| 2 | Historia de las computadoras | 31 |
| 2.1 | Los precursores de las computadoras modernas | |
| 2.2 | El despertar de la computación | |
| 2.3 | La modernidad de las computadoras | |
| 2.4 | La actualidad y el futuro | |
| 2.5 | Actividades | |



1. Computadoras

Placa base de una computadora con sus circuitos impresos.

Fotografía de [Blickpixel](#).

Las utilizamos todos los días, rigen nuestra vida cotidiana, y el mundo moderno dejaría de funcionar sin ellas. Las computadoras son omnipresentes en nuestro día a día, ya sea que trabajemos directamente con ellas o no. Pero, ¿Qué es exactamente una computadora?, y más aún ¿Pará que sirve?.

Este capítulo intenta responder todas esas interrogantes y generar un glosario de los términos más comunes en informática. No es el fin de este capítulo centrarnos en detalles, sino obtener un panorama general de una computadora y como esta funciona. Capítulos siguientes retomarán este contenido y complementarán aportando detalles más específicos.

Comencemos entonces por definir qué es una computadora:

Definición 1.1 Una **computadora** es una máquina electrónica o electromecánica que recibe datos, los analiza, procesa y transforma, convirtiéndolos en información conveniente y útil para el posterior uso por seres humanos.¹

Es decir, una computadora es un dispositivo cuya única función es la de procesar datos. Esos datos, también suelen ser llamados “información”, y podrían ser procesados por un ser humano sin ningún problema. Lo que hace a una computadora realmente interesante es que **es capaz de procesar gran cantidad de datos en muy poco tiempo**. Así, una tarea que implique procesar grandes volúmenes de información, o procesarla de formas complejas (tarea que le podría llevar a una persona meses, o incluso años) a una computadora le puede llevar pocos segundos.²

Toda computadora está formada físicamente por numerosos componentes electrónicos y electromecánicos. Además, toda computadora cuenta con algún dispositivo que permite ingresar datos a la misma (por ejemplo, un teclado o un mouse), y algún otro dispositivo que permite ver los datos que la computadora procesó (por ejemplo, un monitor o una impresora).

Cuando se habla de computadora, generalmente pensamos en la típica **computadora de escritorio** (con su voluminoso gabinete, un monitor, teclado y mouse). Sin embargo, las computadoras

tienen hoy en día formas cada vez más diversas, como por ejemplo, las **notebooks**, y sus primas pequeñas las **netbooks**. Las **tablets**, los **celulares** e incluso los **relojes inteligentes** son también ejemplos de computadoras. Cada una tiene sus particularidades, por ejemplo, la forma de ingresar datos en una notebook no es la misma que en un celular. Sin embargo, todas son computadoras y comparten características comunes.



Vieja Apple II.
Exhibición del Musée
Bolo, Lausana, Suiza.

De hecho, la definición de computadora que utilizamos es tan amplia, que abarca una serie de dispositivos que en general no clasificaríamos como computadoras, pero que, por sus características, lo son. Además de las anteriormente mencionadas podríamos incluir también:

- **Consolas de videojuegos**
- **Juguetes electrónicos**
- **Sistemas de domótica (IoT)**
- **Computadoras de abordo (en autos, barcos y aviones)**
- **Robots**
- **Calculadoras graficadoras programables**
- **Microcontroladores**

Muchos otros dispositivos también podrían ser consideradas computadoras, o incluyen computadoras como un componente fundamental para poder funcionar.³

En realidad

Técnicamente cualquier dispositivo electrónico que realice cálculos (cómputos) es una computadora. Sin embargo, en general nos referimos como computadora a aquellos dispositivos que son “programables”, es decir, que pueden configurarse para llevar a cabo distintas tareas por parte del usuario. Esta distinción no nos resulta relevante.

Una computadora está constituida por dos partes fundamentales: El **hardware**, que comprende a la **parte física** de una computadora, y el **software**, es decir, la parte **parte lógica**. A continuación trataremos más en detalle ambas partes.

1.1 Hardware (parte física)

Comencemos por definir a qué nos referimos cuando hablamos de hardware:

Definición 1.2 El **hardware** comprende todos los elementos físicos que componen a la computadora. Es decir, el conjunto de circuitos, cables, perillas, palancas, botones, luces, displays, dispositivos de impresión, motores, imanes, placas metálicas, etc.¹

Una definición más pragmática sería: **si no funciona y lo puedo patear, es hardware**.

Los componentes físicos que tenga la computadora, serán dependientes del tipo de computadora con la que contemos. Así por ejemplo, algunas tendrán un teclado, mientras que otras contarán con una pantalla táctil; unas tendrán un monitor, mientras otras tendrán un display indicador; etc.

Muchos de estos dispositivos trabajan con elementos que no son puramente eléctricos, sino físicos o mecánicos. A estos se los denomina **dispositivos analógicos**. En general, los datos que emiten o reciben tienen forma de onda (son valores continuos sobre un rango).

Los dispositivos que no emplean partes mecánicas, sino solamente electrónicas, suelen emitir o

recibir datos discretos (valores puntuales, que no conforman un espectro). Este tipo de dispositivos se denominan **dispositivos digitales**.

La mayoría de las computadoras de hoy en días (aunque no siempre es así, ni tampoco fue así siempre) contienen una serie de partes relativamente estándar entre las que destacan:

Fuente de alimentación (Fuente de poder) Se trata de un transformador de electricidad que modifica la corriente que le llega (por ejemplo, desde un enchufe en la pared, con 220V, o desde una batería de litio), al voltaje que requiere la máquina para funcionar (por ejemplo, 12V, 5V, etc.). Como las computadoras son aparatos electrónicos, todas requieren electricidad de alguna forma. Este dispositivo se encarga entonces de darle energía a todos los otros componentes de la máquina.

Motherboard (Placa base o tarjeta madre) Consiste en una tarjeta o placa, que contiene los circuitos principales de la computadora impresos en un material conductor (cobre, oro, etc.) sobre su superficie. La placa cuenta con ranuras (la mayoría estandarizadas) en donde se conectan los diversos componentes del equipo (aunque a veces los mismos pueden estar soldados directamente a la placa). Sus circuitos se encargan de conectar y a los diversos componentes entre sí, permitiendo que la electricidad fluya de uno a otro componente.

CPU (Central Processing Unit o Procesador) Es lo que se conoce como un circuito integrado. Está compuesto de millones de transistores microscópicos, que son capaces de realizar miles de millones de cálculos por segundo. Además el CPU se encarga de coordinar al resto de los componentes, manejar los datos que entran, los que salen etc.

Memoria RAM Otro importante circuito integrado, pero que no realiza cálculos, sino que almacena información. Es el lugar donde el equipo almacena los datos temporales de las cuentas y acciones que va realizando (no debe confundirse con el lugar en donde se guardan datos a largo plazo). La memoria RAM solamente almacena información mientras la computadora esté encendida. Una vez la máquina se apaga, la información se pierde.

Estos componentes son lo que en general llamamos **computadora** en sí misma. El resto de los componentes son conocidos como periféricos. Los periféricos se conectan a la **computadora** para agregar capacidades de entrada y salida de información.⁴

En el lenguaje coloquial, usamos el término **computadora** para referirnos tanto a los componentes principales como a todos los periféricos conectados a los mismos. En general utilizamos el término con esta última acepción pero al hablar de periféricos el término computadora suele referir solo a los componentes principales.

1.1.1 Periféricos

Los **periféricos** son componentes de hardware adicionales que se acoplan a los componentes principales de la computadora. El nombre deriva de que los mismos se colocan en torno a estos (en la periferia).

Definición 1.3 Un **periférico** es un dispositivo de hardware que se acopla a los componentes centrales de una computadora y que permiten el ingreso y/o egreso de información a la misma.⁵

Los periféricos solo tienen por finalidad el permitir ingresar datos o información a los componentes principales de la computadora, o por el contrario, extraer la información que los componentes principales hayan generado para presentársela al usuario. Algunos de estos dispositivos cumplen ambas funciones. El almacenamiento a largo plazo de información también puede considerarse como entrada y salida de datos, y por eso los dispositivos que almacenan información son considerados



Diversos componentes de hardware de una computadora de escritorio.
Ilustración de GregoryJCL.

también periféricos.

Los periféricos se **clasifican en 4 categorías**, según si permiten ingresar datos, extraerlos, ingresarlos y extraerlos al mismo tiempo, o almacenar los datos a largo plazo.

Periféricos de Entrada

Los periféricos de entrada son aquellos que utilizamos para ingresar datos a la computadora. El dispositivo lee lo que el usuario ingresa de forma analógica (por ejemplo, que palancas accionó, o qué botones presionó) y traduce esa información a impulsos eléctricos que serán enviados a la computadora para que esta procese la información ingresada.⁵

Ejemplos comunes de este tipo de periféricos son:

- | | |
|--|---|
| <ul style="list-style-type: none"> ■ Teclado ■ Mouse ■ Trackpad ■ Micrófonos | <ul style="list-style-type: none"> ■ Cámaras Digitales ■ Webcams ■ Joysticks ■ Scanners |
|--|---|

Fuera de los tradicionales, hoy en día los dispositivos móviles cuentan con una amplia cantidad de periféricos de entrada en forma de sensores. Además, computadoras que comprenden sistemas más específicos, como de automatización industrial, pueden contar con sensores más especializados aún. Algunos ejemplos incluyen:

- | | |
|--|--|
| <ul style="list-style-type: none"> ■ Giroscopios ■ Potenciómetros ■ GPS ■ Sensores de luz ■ Sensores de temperatura ■ Detectores de humo/CO_2 | <ul style="list-style-type: none"> ■ Perillas, palancas y botones ■ Sensores de movimiento ■ Sensores de humedad ■ Lectores de tarjetas magnéticas ■ Lectores RFID ■ Lectores de tarjetas perforadas |
|--|--|

La gran mayoría de los periféricos de entrada suelen ser sensores analógicos cuya señal es transformada en digital por algún componente para ser luego enviada al CPU. También existen por supuesto casos de periféricos completamente digitales.

Periféricos de Salida

Los dispositivos de salida incluyen todos aquellos dispositivos mediante los cuales la computadora nos muestra indicaciones de los datos que procesó, o que se encuentra procesando.⁵ Algunos ejemplos de estos dispositivos incluyen:

- Pantallas o monitores
- Impresoras
- Displays digitales
- Luces (LEDs)
- Parlantes
- Auriculares

Nuevamente podemos encontrar sistemas con casos más específicos, como puede ser:

- Proyectores
- Dispositivos de lectura braille
- Fax
- Plotter
- Motores
- Impresoras 3D

Al igual que en los periféricos de entrada, dentro de los periféricos de salida podemos encontrar algunos completamente digitales y otros analógicos. El detalle de cada dispositivo queda sujeto a la curiosidad del lector y no pertinente a este libro.

Periféricos de Entrada/Salida

En esta categoría entran todos los dispositivos que son mixtos, es decir, que permiten tanto el ingreso de datos, como la salida de los mismos. Algunos autores catalogan también aquí los dispositivos de almacenamiento.⁵ Entre este tipo de dispositivos encontramos:

- Pantallas táctiles / multitáctiles
- Impresoras multifunción
- Cascos virtuales



Un teclado Ascom BEG 100 de Ascom Bankensysteme AG, creado específicamente para funcionar con el sistema de información financiera Reuters Dealing 2000. El teclado en este caso es un periférico de entrada y de salida.

Fotografía de DAFlippers.

La realidad es que muchos dispositivos que tradicionalmente eran considerados como solo de entrada, o solo de salida, actualmente están entrando cada vez más en esta categoría. Pueden tomarse como ejemplo los joysticks de consolas de videojuegos. Inicialmente considerados dispositivos de

entrada, hoy la mayoría incluye sistemas de estímulo al jugador, como luces o vibraciones. Las tabletas graficadoras son otro ejemplo de dispositivo que ahora incluye en forma de pantalla táctil, o en forma de tinta electrónica, selectores de herramientas que varían dependiendo de la aplicación, o muestras del dibujo realizado.

Sabías qué

Los **Joysticks** de la consola de videojuegos **Nintendo 64** permitían acoplarle un dispositivo llamado **Rumble Pack**, que transmitía vibraciones a las manos del jugador de acuerdo a lo que estuviera sucediendo en el juego. Sony incluyó la idea en sus controles **DualShock** de la consola **PlayStation** y pronto se transformó en un estándar en la industria. Así, la mayoría de los joysticks actuales ya no son solo dispositivos de entrada, sino de entrada y salida, algo a lo que los jugadores de videojuegos ya se han acostumbrado.

Periféricos de Almacenamiento

Por último, los periféricos de almacenamiento son dispositivos que permiten almacenar datos para su uso en un futuro. Algunos permiten escribir datos múltiples veces, y leer en muchas ocasiones. Otros permiten una única escritura, y muchas lecturas. Algunos ejemplos incluyen:

- Discos rígidos magnéticos
- Discos de estado sólido
- Discos ópticos (CDs/DVDs/Bluerays)
- Discos magnéticos extraíbles (Floppys)
- Memorias flash (pendrives/tarjetas de memoria)
- Cintas magnéticas (cassettes/VHSs)
- Tarjetas perforadas

En general, muchos de estos dispositivos se pueden separar en dos partes, el medio de almacenamiento en sí, y dispositivo que oficia de lector y/o escritor de los datos sobre el medio. En algunas otras ocasiones el medio y el dispositivo que lee o escribe están acoplados en un mismo elemento, no pudiendo separarse, y por tanto considerado un único elemento.

Como mencionamos, muchos de estos medios requieren de un lector especial para poder leer su información. Estos dispositivos pueden ser considerados dispositivos de entrada. También suele ser necesario un dispositivo para escribir en dichos medios, considerado en general como un dispositivo de salida. Cuando el mismo dispositivo se utiliza tanto para leer como para escribir información (por ejemplo una **lecto-grabadora de CDs**) estos suelen considerarse de entrada y salida. El medio en sí (por ejemplo, el CD) no suele ser considerado un periférico.

Algunos autores separan el medio de almacenamiento de su dispositivo de lectura y escritura, y catalogan solo a estos últimos como periféricos de entra, de salida, o de entrada y salida, según el caso. Otros prefieren incluir al medio como una parte necesaria del dispositivo, y agrupan a ambos como un dispositivo de entrada y salida. Finalmente, muchos autores han optado por colocar dichos dispositivos en una categoría completamente propia, como se ha hecho en este caso.

1.1.2 Redes

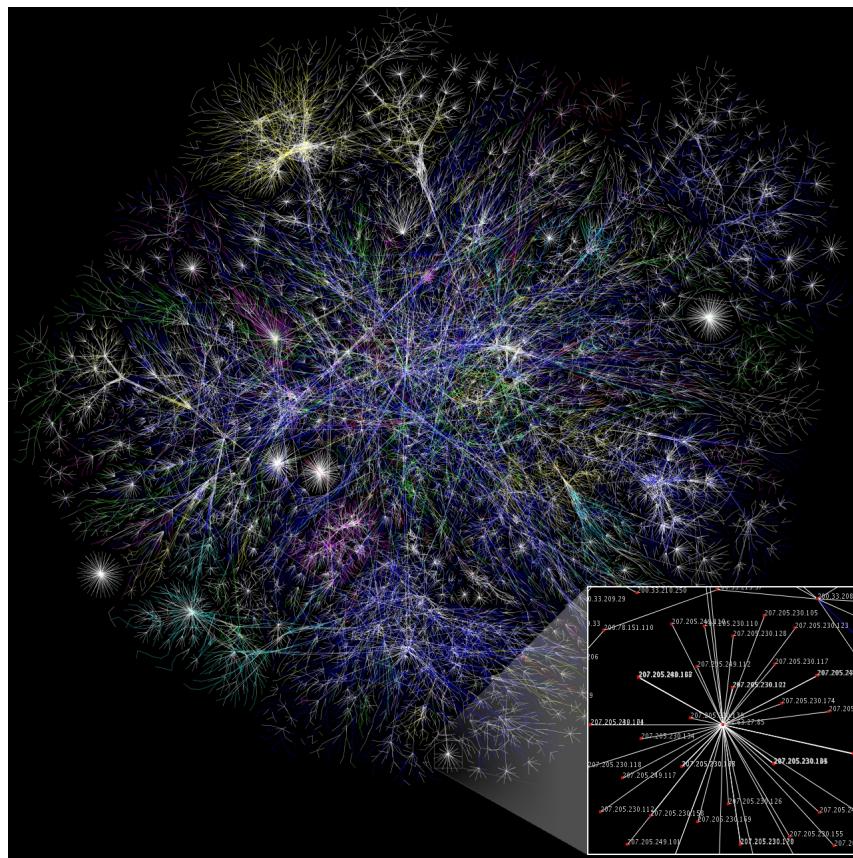
Ahora que sabemos qué es una computadora, resulta interesante saber que dos o más computadoras pueden conectarse entre sí, formando una **red**.

Una **red** no es más que una serie de computadoras conectadas entre sí a través de algún periférico, por ejemplo, una **tarjeta de red cableada**, o una **tarjeta de red inalámbrica**.

Definición 1.4 Una **red de computadoras** es un conjunto de equipos conectados entre sí por medio de dispositivos físicos o inalámbricos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios.⁶

Los equipos pueden conectarse directamente unos con otros, o utilizar alguna computadora particular que oficia de gestor de comunicaciones. En general, la computadora utilizada suele ser una que esté específicamente diseñada para tal fin, como un **router** o un **switch**.

Cuando una serie de computadoras se comunican entre sí, es posible que desde una máquina se acceda a los recursos de otra (archivos, programas, periféricos, e incluso al procesador y memoria). La forma en la que se configura la red permite que esta actúe de formas diversas, por ejemplo, como una única gran computadora, o como diversas máquinas que trabajan de forma independiente pero utilizando una serie de recursos en común, entre otras opciones.



Un mapa de Internet a principios de 2005. Cada línea representa la unión entre dos equipos, y el largo expresa el tiempo medio de comunicación entre ellos. Los colores representan la ubicación de los elementos.

Mapa creado por The Opte Project.

A su vez **una red puede estar conectada a otra red**, formando redes más grandes. Por ejemplo, una computadora puede estar conectada mediante una red pequeña a los dispositivos que tiene cerca, como impresora inalámbrica, celular y otros, utilizando por ejemplo, tecnología Bluetooth (red conocida como **red de área personal** o **PAN**). A su vez, esa red puede formar parte de una red más amplia que conecta todas las computadoras de la casa, y a la que se conectan también televisores y

otros dispositivos, generalmente mediante cables o WiFi, utilizando un router como dispositivo que maneja las comunicaciones (red conocida como **red de área local** o **LAN**). A su vez el router puede estar conectado con el servidor de un **proveedor de servicios de Internet** (conocido como **ISP** por las siglas en inglés), que permite conectarnos a servidores a redes que están más allá de nuestro router. Estos servidores conforman así una red muy amplia (conocida como **red de área ancha** o **WAN**) que a su vez se conecta a otras para formar una red aún más grande, **Internet**.

Para que las máquinas puedan conectarse, estas deben entender como comunicarse entre sí. Para eso, existen protocolos específicos que dictaminan como deben hacerlo, y existen diversos protocolos para diversas funcionalidades. Por ejemplo, el protocolo IP determina como una computadora puede identificar a otras (y a sí misma) en la red, el protocolo HTTP dictamina como dos equipos pueden compartir recursos de hipertexto, por mencionar solo algunos de ellos.

Definición 1.5 Internet (a veces llamado el internet o la internet) es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos de internet (a veces llamadas TCP/IP), lo cual garantiza que las redes físicas heterogéneas que la componen, formen una red lógica única de alcance mundial.⁷

1.1.3 Relación del hardware y el software

Lo que hemos aprendido hasta ahora es que el hardware corresponde a toda la parte física de la computadora, es decir, a todos los circuitos electrónicos, chips, cables, etc.

Sin embargo, no hemos mencionada nada sobre la electricidad que fluye por dichos circuitos. Y es así porque el hardware, efectivamente no incluye a la electricidad, la cual, es considerada parte del software. Es decir, que **el hardware no sirve para nada sin software**. Si no hay electricidad corriendo por los circuitos de la computadora, la misma es solo una pila de metales y plástico.

Cabe destacar que, en general, no pensamos al software como electricidad, pero, la realidad es que todo lo que no sea físico en la computadora (en general denominado lógico) termina siendo nada más que eso, un flujo de electrones que se mueven con cierta energía por los diferentes elementos del hardware.

1.2 Software (parte lógica)

Mientras que el hardware corresponde a toda la parte física, el **software** corresponde a toda la **parte lógica**. Es decir, **toda señal eléctrica que recorre los circuitos**. Aunque en general no nos referimos al software como electricidad, sino que hablamos de **programas, archivos, directorios, etc.**

Definición 1.6 El **software** es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación.¹

Nuevamente una definición pragmática sería **si no anda y solo lo puedo insultar pero no golpear, entonces es software**.

Una computadora sin software no sirve para nada, pues sin electricidad, simplemente no funciona. Más aún, no solo debe haber electricidad, sino que debe haber electricidad en los lugares correctos en el momento correcto para que el equipo funcione de la forma esperada.

Si bien los programas son software, el software no son solo programas, como ya mencionamos. Sin embargo, en ocasiones la palabra software se utiliza muchas veces por diversos autores o en el uso cotidiano como sinónimo de “programa”, omitiendo los otros elementos que están incluidos en el software. El lector debe ser cuidadoso al consultar diversa bibliografía para identificar con qué significado se está utilizando el término.

1.2.1 Firmware

Como una computadora no hace nada sin software, **toda computadora viene de fábrica con algún software mínimo que permite al menos encender la misma y determinar cual es el tarea que la misma debe realizar.** Este sistema se conoce como **firmware**.⁸

En las computadoras de escritorio y notebooks, el **firmware** incluye al sistema **BIOS**, que permite al usuario del equipo configurar cosas como:

- Determinar si, al iniciar, el equipo debe ejecutar lo que encuentre en el disco rígido, en la lectora de CD/DVD o un pendrive
- Optar por sobreexigir al procesador para que funcione más rápido (a pesar de que esto puede dañar el equipo)
- Visualizar datos de control como la temperatura interna de la máquina.
- Elegir si se desea habilitar o no los puertos USB del equipo.

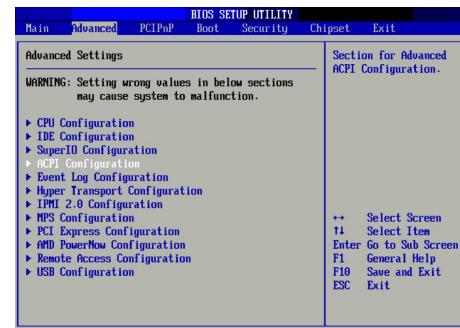
Estas son solo algunas de las muchas tareas que permite realizar, las cuales varían además dependiendo del fabricante y de equipo a equipo.

Además ese sistema realiza un chequeo de todos los componentes internos cada vez que se inicia la computadora, y notifica a los usuarios si algo está fallando, ya sea mediante mensajes en la pantalla, o mediante una serie de notificaciones sonoras.

Definición 1.7 El **firmware** es un programa informático que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Está fuertemente integrado con la electrónica del dispositivo, es el software que tiene directa interacción con el hardware, siendo así el encargado de controlarlo para ejecutar correctamente las instrucciones externas.⁵⁷

El firmware puede venir de fabrica en un circuito que no puede modificarse, conocidos como **ROM**, o venir en circuitos que son “re-escribibles” conocidos como **EEPROM**, lo cual posibilita actualizar el firmware por nuevas versiones del mismo (que en general corrigen errores o agregan nuevas características).

No solo las computadoras de escritorio traen un firmware. También muchos periféricos como **tarjetas gráficas** o **tarjetas de sonido** suelen incluir su propio firmware. Otros elementos como **routers** o **modems** suelen traer un firmware complejo (o incluyen también un pequeño sistema operativo al que consideran parte del firmware) que se encarga de administrar las computadoras conectadas en red a estos dispositivos.



Pantalla que muestra la configuración avanzada de la BIOS AMI.
Fotografía de Richard Masoner.

Algunos grupos de desarrolladores incluso crean **firmware alternativo** (en oposición al oficial que es ofrecido por el fabricante del dispositivo). Este firmware alternativo puede incluir características adicionales a las ofrecidas por el fabricante, aumentando en algunos casos de forma significativa las posibilidades del aparato al exprimir todas las características del hardware.

Sabías qué

DD-WRT y **OpenWrt** son firmwares alternativos y libres para diversos routers inalámbricos. En general el sistema amplifica las posibilidades ofrecidas por los fabricantes. Diversas marcas y modelos de routers son soportados, como Linksys, Buffalo, Belkin, ASUS, Mitsubishi, Motorola, Siemens y D-Link, entre otros.

El proyecto es lo que se conoce como **software libre**, lo que permite a cualquier programador contribuir al proyecto, y está basado en **Linux**, uno de los sistemas operativos libres más populares con una gran base de usuarios.

1.2.2 Sistemas Operativos

El firmware hace muy poco, casi nada a la vista de un usuario actual de una computadora. Salvo en máquinas muy específicas, como pueden ser sistemas de automatización industrial, automotores, u otros, el usuario final requiere de un programa adicional que se encargue de manejar la mayor parte del equipo, y permita al mismo realizar tareas comunes de forma sencilla. Este programa se conoce como **sistema operativo**.

El sistema operativo se encarga de coordinar los diversos componentes del equipo para que todos funcionen, así como de proveer una capa común a los programas que utilizará el usuario final para que estos realicen sus tareas de forma segura y eficiente. El sistema operativo es el encargado de determinar como se lee un CD, como se guardan datos en el disco rígido, o que programa debe ejecutar en que momento, y por cuanto tiempo, entre otras varias tareas. También tiene la potestad de determinar que aplicaciones pueden acceder a que archivos, o terminar programas cuando lo considere necesario, con la finalidad de evitar que una aplicación pueda realizar acciones maliciosas en el equipo. Si bien el sistema operativo es también un programa, este tiene mayores “privilegios” que el resto de las aplicaciones.

Definición 1.8 Un **sistema operativo** es el software principal o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software, ejecutándose en modo privilegiado respecto de los restantes.^{9,10}

La mayoría de los sistemas operativos modernos, proveen adicionalmente una serie de herramientas que posibilitan al usuario utilizar los periféricos del equipo sin demasiada complejidad, o realizar tareas comunes, como crear y editar archivos. Aunque esas herramientas no son parte en sí mismo del sistema operativo, suelen mencionárselas como tales. Por eso, se suele hablar del **núcleo del sistema operativo** (es decir, la parte importante encargada de manejar el hardware), y del resto del sistema (las herramientas, iconos, imágenes, sonidos, etc. que vienen junto con el sistema para permitir una mejor experiencia al usuario). La distinción es sutil, y la mayoría de las veces no es necesaria. Sin embargo, pasa a ser importante en sistemas operativos como Linux, donde precisamente, Linux refiere solo al núcleo, mientras que el sistema en su conjunto se conoce como distribución. Así, existe un único “sistema operativo” Linux, pero cientos de “distribuciones” (o distros) Linux.

Existen muchos sistemas operativos, algunos para uso general en computadoras de escritorio, otros pensados para dispositivos móviles, otros pensados con fines más específicos, como servidores

de internet, o router, robots, etc. También hay sistemas operativos que son desarrollados y vendidos por empresas, y otros que son hechos por comunidades de desarrolladores autoconvocados. Algunos tienen fines comerciales, otros educativos, otros de investigación, y algunos son simplemente por entretenimiento.

En las computadoras de escritorio y notebook actuales es común encontrar instalado alguna versión del sistema operativo **Windows**. Windows es desarrollado y vendido por **Microsoft Corporation**, una de las empresas dedicadas al desarrollo de software más grandes del mundo. El sistema se volvió el más popular en las computadoras de escritorio gracias a las prácticas monopólicas que la empresa lleva adelante, practicando acuerdos con fabricantes de hardware para que el sistema venga pre-instalado, y utilizando lobby político para conseguir acuerdos con instituciones educativas y entidades gubernamentales para la enseñanza de sus sistemas como currícula informática. En Argentina es común que la enseñanza de informática solamente incluya en el programa escolar las nociones del uso de Windows y del paquete de aplicaciones de oficina **Microsoft Office**. Casos análogos se dan en muchos países, no solo latinoamericanos, sino también africanos, europeos y asiáticos.¹¹

Linux es un sistema operativo que merece una mención aparte. Hoy en día Linux ha avanzado mucho desde sus inicios en 1991, cuando un estudiante finés, **Linus Torvalds**, decidió crear su propio sistema operativo y compartirlo con el mundo. Linus adhirió prontamente a las ideas de **Richard Stallman**, un desarrollador que, cansado de un modelo de software en donde los usuarios quedan prisioneros de las políticas y reglas que imponga la empresa, comenzó un movimiento alternativo en donde son los usuarios lo que tienen el poder y las decisiones sobre el software, el movimiento de **software libre**. Stallman también desarrolló una fundación para la promoción del software libre, e inició el **Proyecto GNU**, un proyecto destinado a crear herramientas de software totalmente libres. Por ser software libre, Linux puede ser modificado por cualquier usuario con los conocimientos técnicos necesarios para adaptarse a una gran variedad de necesidades y situaciones.

Así, diferentes **distribuciones** de Linux existen para solucionar diversos problemas de múltiples usuarios, o para satisfacer diferentes gustos y preferencias. Algunas de las distribuciones más populares para computadoras de escritorio son **Ubuntu**, **Linux Mint**, **Debian**, **Fedora**, **RedHat Enterprise Linux**, **SuSe Linux**, **Arch**, **Manjaro** y **Zorin OS**, pero la lista es inmensa. Existen además distribuciones que se enfocan en casos muy específicos: sistemas para servidores de internet, routers, celulares, smart TVs, etc. Como gran cantidad de distribuciones se ofrecen de forma gratuita, es cada vez más común encontrarlo instalado de fábrica en equipos de escritorio o notebooks, ya que el fabricante minimiza costos de venta final al no tener que pagar el software de Microsoft. Además, Linux se ha convertido en un sistema robusto y confiable, tanto así que el 70% de todos los servidores de internet utilizan Linux como su sistema operativo.

Y habla de distribuciones Linux, la más popular hoy en día es casi sin lugar a dudas una versión modificada y adaptada para funcionar sobre teléfonos celulares inteligentes y tablets, **Android**. Android tiene una cuota de mercado del 80%; es decir, 8 de cada 10 celulares o tablets utilizan Android. El sistema surgió de la mano de una pequeña empresa respaldada por **Google** en 2007, realizando acuerdos con la Open Handset Alliance (un conjunto de empresas de hardware, software



Escritorio del sistema operativo Huayra Linux 3.2
Fotografía de Conectar Igualdad.

y telecomunicaciones). Al ser libre, modificable y altamente personalizable, los distintos fabricantes pueden modificar la apariencia del sistema para adaptarlo a su marca y estilo, pero permitiéndolo interactuar con el gran ecosistema de desarrolladores y aplicaciones de la plataforma, y gozando de los avances generales que se llevan adelante en el sistema.

1.2.3 Programas



Pantalla donde se listan las aplicaciones del equipo en un sistema Windows 8.1.

Fotografía de India7 Network.

Los programas o aplicaciones que utilizamos a diario en la computadora son también parte del software. Programas como **Word**, **Excel** o **PowerPoint** para ofimática, **Photoshop**, **GIMP** para diseño gráfico, **iTunes** o **Windows Media Player** para reproducción de música y video, **Super Mario Brothers**, **Pacman**, **Angry Birds** u otros juegos, **todos son software**.

Llamamos a este software **programa o aplicación**. Estos programas son los que brindan funcionalidades específicas al usuario final, como una planilla de cálculo o un reproductor de video. Se “paran” sobre el sistema operativo, y por tanto, quienes lo desarrollan pueden abstractarse del hardware subyacente del equipo. Por ejemplo, el programa de reproducción de música simplemente le dice al sistema operativo “quiero reproducir este sonido”, pero desconoce completamente si el sonido va a salir por los parlantes de la computadora, los auriculares u otro; es el sistema operativo quien se encarga de ello.

Definición 1.9 Una **aplicación** es un programa informático diseñado como herramienta para permitir a un usuario realizar uno o varios tipos de tareas.¹²

El desacoplar el sistema operativo de las aplicaciones permite a los desarrolladores de software generar más aplicaciones, con más funcionalidades, en menos tiempo, pues no tienen que preocuparse por detalles que, a los fines del programa no son relevantes. No solo eso, sino que al desentenderse del hardware subyacente, las aplicaciones pueden correr en cualquier equipo, sin importar el hardware que posea, por lo que son más genéricos y menos susceptibles a cambios en el hardware.



Banner publicitario de CorelDRAW Graphic Suite 2018 con su precio de lista
Captura de pantalla del sitio oficial de Corel.

Las aplicaciones son una parte fundamental del mercado informático, tanto en aplicaciones de escritorio como en celulares y tablets. Tanto es así que las tiendas de aplicaciones de Android y iOS facturaron en 2018 un total de 34.400 millones de dólares. Las posibilidades de este mercado en las plataformas móviles generan un círculo virtuoso en donde los desarrolladores generan constantemente nuevas y mejores aplicaciones, ampliando la oferta, mientras que los usuarios esperan de forma constante nuevas apps, descartando las viejas. Mientras que en el mercado de celulares y tablets hay una idea de “aplicaciones para lo que sea”, en los sistemas operativos de escritorio la oferta suele estar más enfocada a aplicaciones para profesionales y de alta calidad. Así, aplicaciones como Photoshop o Microsoft Office se venden a cientos de dólares.

También hay un gran mercado para aplicaciones gratuitas que basan sus ganancias en publicidad o en venta de datos y patrones de uso de los usuarios, como Google o Facebook. Muchas veces los usuarios desconocen el verdadero costo de las aplicaciones y servicios que utilizan, no entendiendo que sus datos pueden ser vendidos a terceros por la empresa que les ofrece la aplicación de forma gratuita.

Las aplicaciones de software libre son también una moneda corriente, habiendo en general una alternativa libre a casi cualquier aplicación comercial, las cuales, en general se ofrecen de forma gratuita. La calidad final de estas aplicaciones varía enormemente, habiendo desde aquellas en donde la versión libre supera incluso a las versiones comerciales (Un ejemplo es el reproductor de video VLC), y casos en donde la versión comercial está tan por arriba de la libre que optar por esta última se vuelve muy difícil si uno utiliza el software con motivos comerciales (Por ejemplo Adobe Illustrator o CorelDraw están muy por arriba de sus contrapartidas libres).

Finalmente hay un gran mercado de aplicaciones a medida, generadas específicamente para solucionar el problema puntual de una persona u organización. Este software brinda una solución ad-hoc. Estas aplicaciones suelen ser en ocasiones mucho más costosas que aplicaciones genéricas, pues solo se venden a un único usuario, quien debe cargar con la totalidad de los costos de desarrollo.

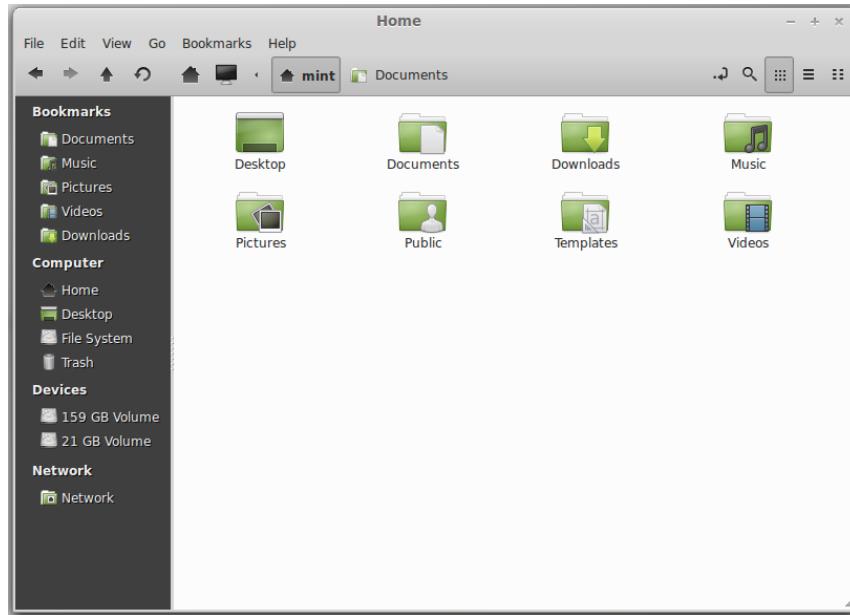
Definición 1.10 Ad-hoc es una locución latina que significa literalmente “para esto”. Generalmente se refiere a una solución específicamente elaborada para un problema o fin preciso y, por tanto, no generalizable ni utilizable para otros propósitos.

1.2.4 Archivos y Directorios

Un **archivo** digital no es más que la representación lógica de un archivo físico (Por ejemplo, una hoja con texto escrito, una foto, una tarjeta, etc.). El término surge en base a una analogía con los elementos en papel que se manipulaban en una oficina previo a la era de la digitalización. Sin embargo, un archivo también puede representar elementos que uno no encontraría en dichas oficinas (o al menos no como papel), como música, videos, etc. En las computadoras modernas, los archivos son guardados a bajo nivel como **bits**, un término que discutiremos en la Unidad II.

Cada archivo posee un **nombre**, que lo identifica, así como un **tamaño** (la cantidad de bits que se utiliza para almacenarlo), y un **tipo** (que dictamina al sistema operativo de que forma deberá procesar ese archivo, por ejemplo, si representa una foto, o un video, etc.). Además, dependiendo del sistema operativo y otros detalles técnicos pueden o no tener más características, como permisos (quien puede acceder al archivo y para hacer qué cosa), marcas temporales (por ejemplo, la fecha en la que fue creado el archivo, la fecha de la última modificación, etc.), por mencionar algunas.

En general (aunque depende del sistema operativo) los archivos no se encuentran “sueltos” en el equipo, sino que se organizan en **directorios** (llamadas también carpetas). Un directorio no es más que la representación virtual de una carpeta física (nuevamente basado en la analogía de la



Captura de Pantalla de Nemo, el manejador de archivos de Linux Mint.

Imagen de Gaba P.

oficina), en donde uno puede agrupar varios archivos. Generalmente, un directorio pueden contener a su vez, otros directorios, formando lo que se conoce como una **jerarquía de directorios**.

Para manipular un archivo informático es importante saber también en que carpeta se encuentra almacenado, ya que su nombre y la carpeta son lo que lo identifica de forma única en todo el sistema.

Definición 1.11 Un **archivo informático** es un conjunto de bits que son almacenados en un dispositivo. Un archivo es identificado por un nombre y la descripción de la carpeta o directorio que lo contiene. Un archivo puede representar cualquier tipo de dato, ya sea texto, imágenes, audio, video, programas u otros.¹

Toda la información que se almacena en los discos rígidos, CDs, y otros periféricos de almacenamiento suele estar organizada en carpetas (en la mayoría de los sistemas operativos), y suelen haber carpetas estándar en donde se almacenan determinados archivos que el sistema requiere para funcionar.

Otras carpetas, en cambio, son manejadas por los usuarios, pudiendo crear nuevas, agregar o quitar elementos a la misma, modificar su nombre, moverlas a otra carpeta, copiarlas a otro dispositivo, etc. La carpeta le sirve al usuario para organizar sus archivos por algún criterio que considere apropiado para sus necesidades.

Definición 1.12 Un **directorio** es un contenedor virtual en el que se almacenan una agrupación de archivos informáticos y otros subdirectorios, atendiendo a su contenido, a su propósito o a cualquier criterio que decida el usuario.¹

Sabías qué

La mayoría de los sistemas operativos modernos incluyen algún programa que permite ver cuales son los archivos y carpetas del sistema. Este programa se conoce en general como **Navegador de Archivos** o **Explorador de Archivos**.

Los sistemas operativos más antiguos también lo hacían, aunque de una forma menos intuitiva, utilizando la **terminal o consola**.

1.2.5 Otros elementos de software

Mencionamos algunos de los elementos más relevantes que son considerados software, como son el **firmware**, el **sistema operativo**, las **aplicaciones**, los **archivos** y los **directorios**.

Existen sin embargo otros elementos que podrían mencionarse, como las **utilidades**, que consisten en programas diseñados para realizar tareas de mantenimiento del sistema, **antivirus**, que consisten en un programa que verifica que los archivos y programas de la computadora no contengan código perjudicial para el usuario, **herramientas de desarrollo**, que consisten en programas pensados para programadores, para que estos puedan crear nuevos programas, etc.

Como ya mencionamos, a fin de cuentas estas categorizaciones son solamente desde un punto de vista del usuario, pues para el hardware, un programa se almacena en un archivo, al igual que una foto o un video, y todo termina siendo información pasando por cables, o para ser más específico, electricidad. Como usuarios de una computadora hablamos por supuesto de archivos, carpetas y aplicaciones, y no de cables y electricidad, realizando un proceso de abstracción. Retomaremos este último concepto sobre el final de la Unidad II.

Vale la pena recordar que, el hardware no sirve para nada sin software que lo maneje, mientras que el software no existiría si no hubiera hardware. Ambos elementos forman una simbiosis, y si bien uno puede centrarse en el estudio de uno u otro con mayor o menor intensidad, es necesario comprender ambos elementos para tener una idea clara del funcionamiento de una computadora.

1.3 Actividades

A continuación se presentan una serie de actividades orientadas a reforzar los conceptos vistos en este capítulo. Realice las actividades investigando en Internet si lo cree conveniente.

Ejercicio 1.1 Mire el cartel a continuación

Determine.

¿Cuáles de los elementos mencionados corresponden a software?

¿Cuáles de los elementos mencionados corresponden a hardware? ■

Ejercicio 1.2 ¿Conoce otros sistemas operativos además de los mencionados? ¿Conoce algo más sobre los sistemas mencionados?

Investigue en internet para descubrir nuevos sistemas y ampliar su conocimiento de los vistos. ■

Ejercicio 1.3 ¿Se le ocurren otros dispositivos de hardware que use de forma cotidiana?

¿Son periféricos de entrada, de salida, de entrada y salida, o de almacenamiento? ■



2. Historia de las computadoras

Museo de Historia de las Computadoras en Mountain View, California, EE.UU..

Fotografía de Michael Kappel.

La historia de las computadoras está directamente ligada a las historias de la matemática, la lógica, la gramática y la ingeniería mecánica y electrónica. Así, los seres humanos han utilizado dispositivos que los ayuden a computar (calcular) diversas cosas desde hace milenios, así como desarrollado técnicas que hoy en día se utilizan ampliamente en ciencias de la computación.

Este capítulo busca realizar un pequeño recorrido histórico por el desarrollo de las ciencias de la computación, no solo desde una perspectiva del hardware, sino también del software y de las teorías que sustentan a la disciplina. En el transcurso de este viaje se espera que el lector no solo se lleve una serie de anécdotas, sino que pueda apreciar el estado de las ciencias de la computación en la actualidad como parte de un proceso evolutivo que continúa en desarrollo, así como descubrir algunos conceptos que verá de forma recurrente si opta por desarrollarse en la disciplina.

Pero no solo veremos el pasado de la computación, también haremos un pequeño análisis del presente, para cerrar el capítulo con un vistazo a lo que depara el futuro.

2.1 Los precursores de las computadoras modernas

Pensar que la computación es un desarrollo que se ha realizado solamente en los últimos años del siglo XX es una idea ingenua. Los seres humanos hemos tenido la necesidad de contar y calcular (**computar**) cosas desde hace mucho, mucho tiempo; tanto tiempo como la prehistoria.

Desde el desarrollo de pequeños utensilios para asistir en los trabajos de computo, pasando por el desarrollo de conceptos matemáticos y trabajos manuales de trabajadores y trabajadoras cuyo esfuerzo nunca fue reconocido ni remunerado, hasta el desarrollo de maquinarias con complejos engranajes que permitieran asistir en el trabajo computacional. Todos terminan por ser precursores, de una u otra forma, de un proceso mental que se fue generando en la humanidad; la necesidad de contar con computadoras.

Y es que, a medida que avanza la ciencia y la técnica, también avanza la necesidad de realizar cálculos cada vez más complejos, y cada vez, en menor tiempo. Las computadoras no son una

casualidad, son una necesidad intrínseca al proceso de desarrollo científico-tecnológico que ha desarrollado la humanidad.

2.1.1 La prehistoria de la computación

Los registros cuentan que el ser humano parece haber comenzado a contar desde al menos el año 20.000 AC. Por ejemplo, varios científicos, antropólogos e historiadores nos comentan sobre la utilidad del **hueso de Ishango**, un utensilio de hueso que data del paleolítico superior (20.000 AC.), y que habría servido para realizar conteos y cuentas. Incluso parecería que ya en esa época, los seres humanos tenían alguna noción sobre números primos. Obviamente al no haber registros escritos, es imposibles saberlo con seguridad.¹³

Pero los verdaderos avances en materia de cuentas comenzaron en la antigua Mesopotamia con la invención del **ábaco sumerio**, al rededor del 2500 AC. El ábaco consiste en un cuadro de madera con barras paralelas por las que corren una serie de bolas móviles. Este aparato permitía realizar cálculos sencillos (sumas y restas). Los chinos pronto en el 190 AC. mejorarían la idea con la creación del **suanpan**, conocido también como el **ábaco chino**. Este incluía las características del ábaco sumerio, pero las ampliaba permitiendo realizar también multiplicaciones y divisiones gracias a su innovador diseño.¹⁴

Sabías qué

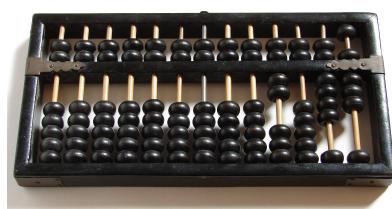
El ábaco todavía se utiliza hoy en día para enseñarle operaciones sencillas a los niños, e incluso el suanpan es utilizado ampliamente por mercaderes, comerciantes y contadores en áreas donde el uso de tecnología es poco práctico.

En el 910 AC se crea también en China el **carruaje que apunta al sur**, un dispositivo que actúa de forma similar a una brújula, y que permitía a los mercaderes orientarse en los caminos. Este dispositivo es el primero en utilizar un engranaje, una de las piezas fundamentales de las **computadoras mecánicas**.

Pero si de engranajes hablamos, el **Mecanismo de Anticitera**, datado del 150 AC, es ya una completa **computadora mecánica**. Habría sido construido por científicos griegos con la intención de calcular eventos astronómicos y calendáricos, así como determinar la fecha exacta de los juegos olímpicos y otros certámenes deportivos de la antigüedad. La complejidad de este aparato no fue equiparada hasta siglos después, con la invención de los primeros relojes mecánicos.¹⁵

En el área de las matemáticas, los avances de los chinos, utilizando el número cero, y los números negativos, así como los aportes del matemático Indio Pingala desarrollando el **sistema de numeración binario**, el cual es la base de todas las computadoras modernas sentarían las bases para la matemática moderna y por tanto para las ciencias de la computación.

El matemático Abu Abdallah Muḥammad ibn Mūsā al-Jwārizmī, conocido en general como al-Juarismi, quien fue un prestigioso matemático, astrónomo y geógrafo persa alrededor del año 800. Dentro de sus numerosas obras se le atribuye el haber propuesto por primera vez la idea de que, mediante una especificación clara y concisa de pasos a realizar, se podían calcular sistemáticamente, y por tanto se podrían desarrollar dispositivos mecánicos similares al ábaco que realizarán trabajaran sobre este para realizar las cuentas en lugar



Un suanpan. Su diseño permite contar en decimal y hexadecimal.
Archivo de David R. Tribble.

de tener el usuario que emplear las manos. Este concepto es el precursor de lo que hoy en día se conoce como **algoritmo** (llamado así como una deformación del nombre al-Juarismi).

Por supuesto que solamente mencionamos algunos de los hitos más destacados del mundo antiguo, el cual estuvo lleno de avances científicos y matemáticos que aportaron de alguna forma u otra a las ciencias de la computación.

2.1.2 Las computadoras humanas



Imagen de una mujer operando una de las primeras computadoras, una IBM 704, en el Jet Propulsion Laboratory de la NASA.

Imagen de NASA/JPL-Caltech.

A medida que la ciencia avanzaba, se volvía necesario muchas veces realizar gran cantidad de cálculos para lograr comprender el universo que nos rodeaba. Por ejemplo, las áreas de astronomía y de física, solían requerir una gran cantidad de cuentas para arribar a conclusiones sobre las observaciones realizadas sobre diversos fenómenos.

Así, era común que los científicos contrataban a personas que realizaban los cálculos, mientras ellos se centraban en otros experimentos u observaciones. En ocasiones en los que los cálculos tardarían semanas o meses para realizarse, contrataban varias personas trabajando día y noche.

Las personas contratadas debían poseer sin embargo una serie de conocimiento (saber leer y escribir, ser capaz de realizar cálculos matemáticos, etc.) que no eran comunes en la población general. Esto y la necesidad de abaratar costos llevó a que se contraten en general mujeres, generalmente de clases sociales acomodadas y que hayan recibido educación. En un contexto donde la mujer no trabajaba (salvo en el hogar) la remuneración ofrecida podía ser baja.[16, p23]

A estas personas se las conocía como **computadoras**. El término que hoy usamos para los dispositivos, deviene de esas mujeres que, en la mayoría de los casos, la historia ha omitido de sus páginas.

Incluso luego de la invención de las primeras computadoras electromecánicas, las computadoras humanas se seguían empleando, pues eran consideradas más fiables, e incluso más económicas en algunos casos que comprar una computadora.

Sabías qué

Alexis Clairaut, Jérôme Lalande, y Nicole-Reine Lepaute, los astrónomos que calcularon la fecha del retorno del cometa Halley en 1750, habían contratado a un equipo de computadoras humanas para realizar el cálculo. En ese equipo se encontraba Nicole-Reine Etable de la Brière Lepaute, una de las primeras mujeres de las cuales se tiene registro que oficiaron de computadoras.

2.1.3 Las calculadoras mecánicas

En 1642 **Blaise Pascal**, matemático y físico francés, cansado de realizar cálculos manualmente, desarrolló la primera calculadora mecánica de la cual se tengan registros, la **Pascalina**.

Era una pequeña caja de madera que tenía en la tapa una hilera de discos numerados, con los agujeros para introducir los dedos y hacerlos girar. Funcionaba con engranajes, de forma que cuando una rueda daba una vuelta completa, avanzaba la otra rueda situada a su izquierda. Esta calculadora solo podía realizar sumas y restas. En 1672, **Gottfried Leibniz** perfeccionaría el dispositivo, creando el **Cilindro de Leibniz**, que permitía además multiplicar y dividir.¹⁷

Sabías qué

El Cilindro de Leibniz fue la base de numerosas calculadoras mecánicas creadas hasta la década de 1970, cuando finalmente se reemplazaron por calculadoras electrónicas.

Pero el invento tal vez más ambicioso fue el ideado por el matemático británico **Charles Babbage**. Babbage creó la **Máquina Diferencial**, una calculadora mecánica capaz de calcular gran cantidad de polinomios. Su máquina funcionaba con engranajes metálicos conectados, como una gran pieza de relojería. Babbage consiguió fondos de la Royal Astronomical Society para la construcción de su máquina. Sin embargo nunca logró hacerla funcionar. Los engranajes de la época no eran lo suficientemente buenos, y los materiales se rompían o generaban vibraciones que desarmaban el equipo. Desesperado, Babbage modificaba constantemente su diseño. Finalmente, la Royal Astronomical Society cortaría el financiamiento y el proyecto sería abandonado.

2.1.4 La primera computadora mecánica

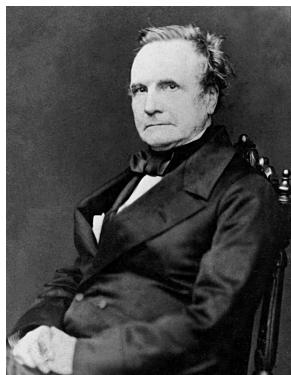
Años después, en 1833, Charles Babbage volvería a intentarlo, pero esta vez, con una idea todavía más radical, la **Máquina Analítica**. Este equipo tenía la característica de ser **programable**. Es decir, se podía configurar para realizar distintas tareas en distintos momentos. Los planos diseñados por Babbage definían un complejo equipo capaz de realizar diversos cálculos, almacenar los resultados en una memoria, y hasta imprimirloros con un dispositivo similar a una impresora.

Sobre hombros de gigantes

Babbage no inventó todo de cero. Su sistema de engranajes se basaba en gran parte en el Cilindro de Liebniz. Como característica adicional su máquina permitía ser programada mediante tarjetas perforadas.

El sistema fue copiado de un telar creado en 1801 por **Joseph Marie Jacquard**, un tejedor y comerciante francés. El telar de Jacquard era **programable** a través de un sistema de **tarjetas perforadas**, y permitía a una persona diseñar complejos patrones en el tejido. Esta idea revolucionaria permitía reducir costos al permitir diseñar tejidos complejos con un muy bajo esfuerzo y reducido personal.

El invento de Babbage hubiera estado dotado de la mejor tecnología de la época.



Charles Babbage (1860).

Dominio Público.

Charles Babbage buscó desesperadamente financiamiento para su dispositivo, pero tras el fracaso anterior, el gobierno y los inversores privados eran escépticos. Sin embargo, una joven aristócrata y matemática se acercó a Babbage por sus diseños. Se trataba de Augusta Ada Byron, Condesa de Lovelace, conocida más simplemente como **Ada Lovelace**, hija del famoso poeta Lord Byron. Ada reconoció el potencial en los diseños de Babbage, y comenzó a diseñar programas para el dispositivo. Por este motivo **Ada Lovelace se considera la primer programadora de la historia.**¹⁸

Los deficientes materiales y técnicas de construcción de la época hicieron que todos los esfuerzos de Babbage por construir la máquina analítica sean un fracaso. Así, el proyecto cayó en el olvido. Sin embargo, muchos años después, su máquina sería recreada utilizando los planos originales, y se demostraría que esta funcionaba correctamente. Por esto, **Charles Babbage es considerado el padre de la computadora moderna, así como de la impresora.** Además, realizó numerosas contribuciones a las áreas de la matemática y de la criptografía.

2.2 El despertar de la computación

La computación se empieza a plantar como una realidad cada vez más inminente con llegada del nuevo siglo (1900). Las ideas de Babbage serían reflotadas, ya en una época en donde los avances en la técnica metalúrgica permitía crear mejores dispositivos mecánicos y en donde los descubrimientos en materia de electricidad planteaba nuevas posibilidades.

Muchos científicos y matemáticos comienzan a plantearse las posibilidades y los límites de dispositivos que computen, o realicen pruebas matemáticas automatizadas, y grandes avances se realizan en estos términos, dando lugar a las ciencias de la computación como una disciplina nueva dentro de la rama de las matemáticas, pero que interactúa en gran medida con la electrónica, la mecánica y la ingeniería.

La Primera Guerra Mundial, y el advenimiento de una segunda gran guerra, en donde contar con una superioridad tecnológica podía ser clave para la victoria, llevó a que gobiernos tanto del eje como aliados financiaran serios desarrollos e investigaciones en el área.

2.2.1 La computación como teoría

En 1920 la matemática atravesaba una fuerte crisis. Los nuevos conceptos y abstracciones desarrollados en la disciplina a finales del siglo XIX traían aparejadas una serie de paradojas e inconsistencias en los fundamentos básicos de la matemática. En ese contexto, **David Hilbert**, un matemático alemán, propuso un proyecto de investigación conocido como “programa de Hilbert”, que intentaba solucionar el problema mediante la reformulación de la matemática sobre bases sólidas y completamente lógicas. Hilbert creía que, en principio, esto podía lograrse, mostrando dos cosas:

1. Toda la matemática se sigue de un sistema finito de axiomas escogidos correctamente.
2. Se puede probar que tal sistema axiomático es consistente.

En 1931, **Kurt Gödel** publicaría su **teorema de la incompletitud**, demostrando que no se podía probar la completitud de ningún sistema formal no contradictorio que fuera suficientemente amplio para incluir al menos la aritmética, sólo mediante sus propios axiomas. Así probó que el

ambicioso plan de Hilbert era imposible tal como se planteaba.¹⁹

La necesidad de comprender las implicancias del trabajo de Gödel llevaron pronto al desarrollo de la **teoría de la computabilidad** y de allí a la **lógica matemática** como disciplina autónoma en los años 30. Esto sería la base para los desarrollos teóricos en ciencias de la computación realizados por Alonzo Church y Alan Turing.

La **teoría de la computabilidad** comenzó a estudiar que problemas pueden ser resueltos mediante un **algoritmo** lo cual permite de alguna forma saber que problemas pueden ser resueltos mediante una computadora, y cuales no.

Definición 2.1 Un **algoritmo** es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permiten llevar a cabo una actividad mediante pasos sucesivos que no generen dudas a quien deba hacer dicha actividad.

En Inglaterra, en 1936, **Alan Turing** desarrolla un modelo matemático conocido como **Máquina de Turing**. Si bien se conoce como “máquina”, no hay ninguna construcción física involucrada, sino que se trata de un dispositivo hipotético, capaz de llevar adelante un algoritmo con el fin de solucionar un problema.

El dispositivo imaginado por un Turing consiste en un cabezal que manipula símbolos sobre una tira de cinta que se extiende infinitamente, basándose en una tabla de reglas. A pesar de su simplicidad, una máquina de Turing puede ser adaptada para simular la lógica de cualquier algoritmo de computadora y es particularmente útil en la explicación de las funciones del CPU dentro de una computadora.

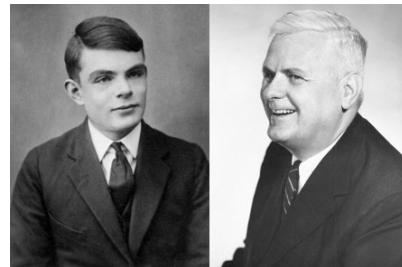
El modelo formal de computadora de Turing le permitió demostrar que **existían problemas que una computadora no podía resolver, sin importar que tan poderosa fuera la misma**. Demostró que es imposible resolver el “Entscheidungsproblem”, un problema matemático planteado por Liebniz en el siglo XVII, y retomado por Hilbert al plantear su propuesta de refundar la matemática.¹⁹

Al mismo tiempo pero del otro lado del Océano Atlántico, en Estados Unidos, **Alonzo Church** desarrolla un modelo matemático conocido como **cálculo lambda**, un sistema formal diseñado para investigar la definición de función, la noción de aplicación de funciones y la recursión.

Se puede considerar al cálculo lambda como el lenguaje universal de programación. Consiste en una regla de transformación simple (sustitución de variables) y un esquema simple para definir funciones. Hoy en día este modelo teórico es la base para el análisis formal de problemas computacionales, así como la creación de nuevos lenguajes de programación.

El cálculo lambda permitía probar, de forma similar a las máquinas de Turing, que **hay problemas que no pueden solucionarse con una computadora**. Así, de forma independiente y mediante métodos distintos, dos científicos habían llegado a la misma conclusión.

Posteriormente se formularía la **tesis de Church-Turing**, la cual formula hipotéticamente la equivalencia entre los conceptos de función computable y máquina de Turing, que expresado en lenguaje corriente vendría a ser “todo algoritmo es equivalente a una máquina de Turing”. Esto quiere decir que el cálculo lambda y las máquinas de Turing, así como otros formalismos creados para analizar las posibilidades de la computación, son todas equivalentes.¹⁹



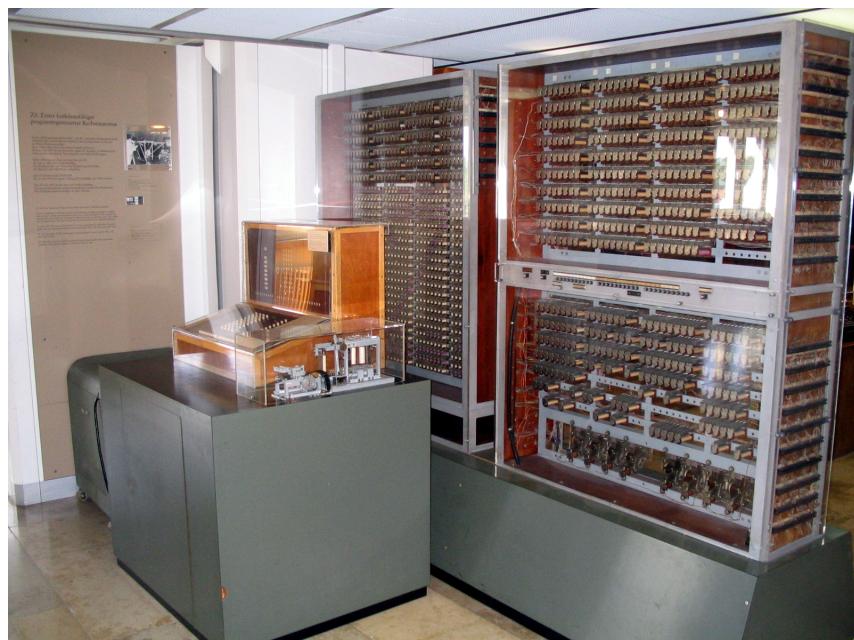
Alan Turing (izq.) y Alonzo Church (der.).
Fotografía de Dominio Público.
Universidad de Princeton.

Lo interesante de estos desarrollos es que por primera vez se demostraron, no las utilidades prácticas de contar con computadoras, sino los límites de las mismas, pero también sus enormes posibilidades. Los formalismos desarrollados por estos científicos siguen presentes actualmente y son la base sobre la que se sustentan muchos de los avances en ciencias de la computación.

2.2.2 Primeras computadoras electromecánicas

Ya llegada la mitad de la década de 1930 nos encontramos en un contexto en donde los trabajos de Babbage comienzan a hacer soñar a muchos científicos e ingenieros. Además, los nuevos trabajos teóricos de Turing y Church auguraban grandes posibilidades para la computación.

Considerado hoy el inventor de las computadoras modernas, Konrad Zuse diseñó la **Z1** en 1935, y terminó de construirla en 1938. La Z1 era una máquina completamente mecánica, era sumamente lenta y solo funcionaba unos pocos minutos cada vez sin necesidad de reajustes. Sin embargo, era la primer computadora realmente funcional. Más tarde realizó una segunda versión de su computadora utilizando **relés** (una especie de interruptor operado mediante magnetismo), bautizada la **Z2**.



Réplica de la Z3 de Zuse, la primer máquina completamente automática y digital (electromecánica) en el Deutsches Museum en Alemania.

Archivo del Deutsches Museum.

Esperando conseguir financiamiento para sus proyectos, presentó la Z2 al Deutsche Versuchsanstalt für Luftfahrt (Laboratorio Alemán para la Aviación) en 1940, siendo la presentación una de las pocas veces en que la Z2 funcionaría realmente. De esta forma consiguió el financiamiento del gobierno para crear la **primer computadora digital, automática y programable del mundo, la Z3**. Completada en 1941, la Z3 fue construida utilizando 2.600 relés y operaba a una velocidad de 4 o 5 Hz. Se debía inicializar la computadora manualmente, y ejecutaba programas que se almacenaban en cintas perforadas. La Z3 no fue considerada de vital importancia por el gobierno alemán durante la Segunda Guerra Mundial, por lo que operó poco tiempo antes de terminar destruida tras los bombardeos aliados a Berlín. Por haberse desarrollado en Alemania bajo el gobierno de Hitler, la historia omitió durante años el trabajo de Zuse.³

En el bando de los aliados, en Estados Unidos, comienzan a aparecer proyectos de desarrollos de computadoras, apoyadas por el estado (generalmente por la rama militar), universidades o empresas privadas.

Los avances en la tecnología y la reducción de costos permitían reemplazar relés (que contenían piezas mecánicas) con **tubos de vacío**, un dispositivo que actuaba de la misma forma, pero completamente electrónico, sin piezas móviles. Como contrapartida los tubos de vacío se iluminaban y generaban gran cantidad de calor, y en sus primeras versiones requerían de gran cantidad de espacio.

Ya en 1944 aparece **ENIAC**, financiada por el Ejército de los Estados Unidos, y utilizada para el cálculo de trayectorias balísticas. La computadora ocupaba un sótano entero en la Universidad de Pensilvania. Físicamente, la ENIAC tenía 17468 tubos de vacío, 7200 diodos de cristal, 1500 relés, 70000 resistencias, 10000 condensadores y cinco millones de soldaduras. Pesaba 27 Toneladas, media $2,4mx0,9mx30m$ ($167m^2$), requería la operación manual de unos 6000 interruptores, y su programa o software, cuando requería modificaciones, demoraba semanas de instalación manual.²⁰

No poco después aparece la primer computadora construida por **IBM**, una de las empresas que más aportaría a la computación mundial. La computadora en cuestión, el **IBM Automatic Sequence Controlled Calculator (ASCC)**, más conocido como **Harvard Mark I**, fue una computadora electromecánica, construida por IBM y enviado a Harvard en 1944. Tenía 760.000 ruedas y 800 kilómetros de cable y se basaba en la máquina analítica de Charles Babbage. Medía unos 15,5 metros de largo, unos 2,40 metros de alto y unos 60 centímetros de ancho, pesaba aproximadamente unas cinco toneladas y poseía unas cubiertas de cristal que dejaban que se admirara toda la maquinaria de su interior.²⁰

Sabías qué

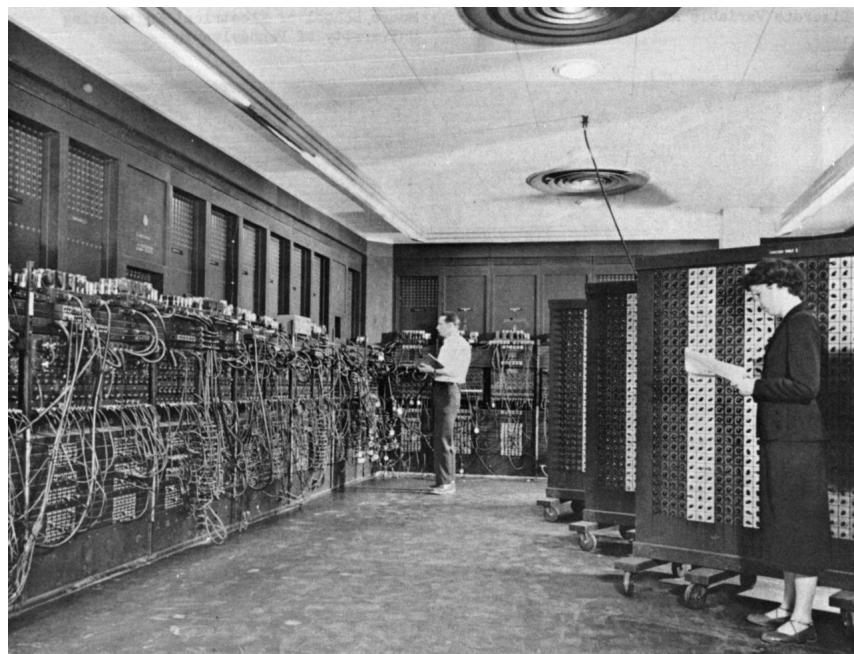
ENIAC fue creada por **John Presper Eckert** y **John William Mauchly**, quienes se llevaron el crédito principal, pero su programación fue realizada exclusivamente por 6 mujeres que fueron generalmente omitidas de los libros de historia: **Betty Snyder Holberton**, **Jean Jennings Bartik**, **Kathleen McNulty Mauchly Antonelli**, **Marlyn Wescoff Meltzer**, **Ruth Licherman Teitelbaum** y **Frances Bilas Spence**.

Al ENIAC le sucedió **EDVAC**, utilizado también para balística, mientras que los ingleses desarrollaron las máquinas **Colossus** (creadas con intención de descifrar los códigos alemanes). Otras como **UNIVAC** (la primera en ser vendida en los Estados Unidos) y la **Z4** de Zuse les sucedieron. Las computadoras eran todavía artefactos muy costosos (EDVAC por ejemplo tuvo un costo aproximado de unos US\$ 100.000) y utilizados en principio solo con fines militares, pero pronto expandiéndose hacia la investigación científica y los negocios.

Sabías qué

Los tubos de vacío atraían insectos que se quedaban en ellos, causando el mal funcionamiento de los equipos. Lo técnicos debían entrar y remover los insectos para que la computadora vuelva a funcionar. A este proceso se le conoce como **debuggear** (del inglés quitar los bichos). El término quedó, y hoy en día se conoce con ese nombre al proceso de arreglar los errores en un programa.

A medida que aparecían más y más computadoras, también se comenzó a desarrollar una idea más seria sobre **arquitectura de computadoras**. Es decir, se comenzaron a desarrollar modelos y descripciones funcionales de los requerimientos y las implementaciones de diseño para varias partes de un computadora, en especial en la forma en que el procesador trabaja internamente y



Glen Beck y Betty Snyder programan la ENIAC en el edificio 328 de la Ballistic Research Laboratory (Filadelfia, Pensilvania).
Fotografía del Ejército de los Estados Unidos.

accede a las direcciones de memoria.

Definición 2.2 La **arquitectura de computadoras** es el diseño conceptual y la estructura operacional fundamental de un sistema de computadoras.

La Mark I sentó las bases para lo que hoy se conoce como **arquitectura de Harvard**, que involucra una computadora en donde las pistas de almacenamiento y de señal se encuentran físicamente separadas para las instrucciones y para los datos. Por otro lado **John Von Neumann**, un matemático húngaro-estadounidense desarrolló el primer borrador sobre la arquitectura de EDVAC, describiendo detalles arquitecturales que simplificaban la arquitectura de Harvard, además de optimizar costos. La arquitectura pasó a ser conocida como **Arquitectura de Von Neumann**, y se transformó en un estándar, al punto de que casi todas las computadoras modernas la utilizan.²¹

2.2.3 IBM, Bell y grandes computadoras

Ya terminada la guerra, el boom de las computadoras estaba en el aire, y el potencial de estos dispositivos en otras áreas distintas a la guerra comenzaban a vislumbrarse.

Una empresa venía ya vendiendo dispositivos electrónicos para grandes compañías y fábricas, como máquinas tabuladoras y relojes de fichada. Se trataba de **IBM**. Con la experiencia del desarrollo del **Harvard Mark I**, y tras terminada la guerra, IBM crea la **IBM 701** (1953), considerada la primer computadora comercial basada en tubos de vacío. Se siguieron pronto otros modelos, como la **IBM 702** y la **IBM 650**.

Sabías qué

En 1956, **Arthur L. Samuel**, del laboratorio de IBM en Poughkeepsie, Nueva York, programó un IBM 704 para jugar a las damas utilizando un método por el que la máquina podía “aprender” a partir de su propia experiencia. Se cree que este es el primer programa de “auto-aprendizaje”, una demostración del concepto de **inteligencia artificial**.

IBM se transformó rápidamente en la empresa líder en todo lo referente a computadoras. Invirtiendo gran cantidad de dinero en investigación y desarrollo (I+D). En 1957 IBM desarrolló el primer sistema de almacenamiento informático basado en disco, llamado el IBM 305 RAMAC. El **RAMAC ATOGA** es el predecesor de los discos duros actuales y estaba formado internamente por cincuenta discos.²²

Sabías qué

IBM también desarrolló el lenguaje de programación científico **FORTRAN** (FORmula TRANslation) para facilitar la creación de programas. También fueron los inventores de otras tantas tecnologías ampliamente utilizadas hoy en día, como, **la memoria RAM, las tarjetas magnéticas y los códigos de barras**.

IBM no fue la única empresa que incursionó en tecnología. Otra de las grandes empresas que lo haría sería **AT&T** quien controlaba los teléfonos de Estados Unidos de forma monopólica, y principalmente su subsidiaria, **Bell Labs** (que gracias al monopolio contaba con uno de los presupuestos más altos del mundo en términos de investigación en tecnología).

Entre las patentes y descubrimientos más importantes de Bell Labs destacan **la libreta de un solo uso, el transistor, el láser, la fibra óptica, la tecnología DSL, la telefonía móvil, los satélites de comunicaciones, el sistema operativo Unix, el lenguaje de programación C y el lenguaje de programación C++**. Además, once de sus investigadores han sido galardonados con Premios Nobel.²³

Las computadoras sin embargo, seguían siendo costosas, grandes, pesadas, y requerían de personal altamente capacitado para su instalación y armado. Por eso solamente se encontraban disponibles en universidades, dependencias del estado o grandes empresas privadas, que podían solventar los costos de estos aparatos.

Cabe destacar que la operación de las computadoras era tarea compleja. **Programarlas requería de expertos, que supieran como estaba armada la computadora (como estaban diagramados los cables, los circuitos, etc)** y por tanto, solamente ingenieros electrónicos o matemáticos eran capaces de operarlas. Adicionalmente, **como cada computadora era distinta, un programa escrito para una computadora no funcionaba en otra**, y un profesional que supiera operar una computadora tenía que volver a aprender todo de cero para poder operar otra. Comprar una computadora era realizar un compromiso a largo plazo con el fabricante, y por eso las pequeñas empresas no podían competir contra compañías como IBM, que podían ofrecer profesionales capacitados, mantenimiento y muchos otros servicios al comprador de sus equipos.

2.2.4 Los lenguajes de programación

Las primeras computadoras no venían con software para el usuario final, sino que solamente permitían ser programadas. Es decir, todo usuario de la computadora debía primero transformarse en un programador. El proceso de programar implicaba conocer la arquitectura del equipo, y pasar las ideas que tuviera el usuario a complejos códigos en tarjetas perforadas (es decir, había que aprender en qué lugares perforar para lograr el comportamiento deseado del equipo, algo que



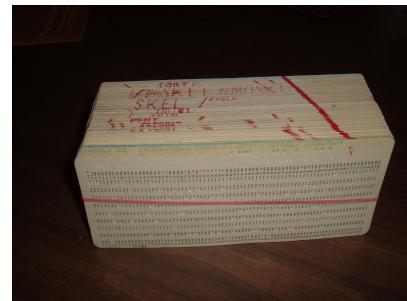
Ken Thompson (sentado) y Dennis Ritchie (parado), creadores de UNIX, trabajando en Bell Labs en una PDP-11.
Fotografía de Peter Hamer.

además variaba de máquina a máquina).

Los primigenios programadores, debían lidiar constantemente con la incompatibilidad de los programas que escribían. Así, un científico en Harvard no podía simplemente darle su programa a alguien en Princeton, pues las computadoras eran distintas, y por tanto, también los programas que se escribían para ellas. Tampoco era sencillo actualizar una computadora simplemente comprando una nueva, pues los programas ya escritos dejaban de funcionar.

Con la intención de solventar estos problemas, y reducir adicionalmente los costos asociados a la programación de las computadoras, es que aparecieron los primeros **lenguajes de programación**. Estos permitían programar la computadora mediante un conjunto de instrucciones fáciles de recordar por los programadores.³

A medida que avanzaba la técnica, también avanzaban los lenguajes de programación. Comenzaron a crearse lenguajes que funcionaban en múltiples computadoras. Así, el usuario podía escribir un programa en FORTRAN o COBOL, que funcionara en todos los equipos que soportaran dicho lenguaje de programación. Esto era un gran avance, no solo en términos de expansión del conocimiento, sino también en la reducción de tiempos, y por tanto costos, en los procesos de desarrollo. Adicionalmente, el desarrollo de software dejaba gradualmente de ser un trabajo solo de científicos y matemáticos, para pasar a ser algo que



Un programa escrito en tarjetas perforadas. Las marcas en la parte superior de la pila de cartas permitía identificar el programa y visualizar sus diversas partes.

Fotografía de Arnold Reinhold.

podía realizar cualquier persona con algo de formación.

Los lenguajes más avanzados permiten a los programadores expresar mejor sus ideas, y cuanto menor el trecho entre la idea y la aplicación en el equipo, menor el tiempo que toma llevar la idea adelante.

2.2.5 La revolución de los transistores y los circuitos integrados

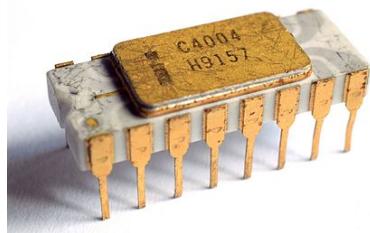
La verdadera revolución en el campo de las computadoras surge con la invención del **transistor**. Este dispositivo reemplazaba al tubo de vacío, siendo mucho más fiable, económico, duradero, pero por sobre todo, pequeño que el tubo.

Los transistores permitían disminuir significativamente el tamaño, el peso y la complejidad de una computadora. Además, a medida que avanzaba la técnica, los mismos se podían hacer más y más pequeños.

Sabías qué

El transistor moderno que utilizamos en nuestras computadoras ronda los 14nm (nanómetros). Además, se han creado exitosamente transistores más pequeños, de tan solo 5 y 6 nm, e incluso de 1nm (aunque son todavía experimentales).

Para tener una comparación, el virus del VIH mide 100nm, un glóbulo rojo aproximadamente 1000nm, y un grano de arena mide entre 63000nm y 200000nm.



Un procesador Intel 4004.
Fotografía de Thomas Nguyen.

En 1959 el ingeniero **Jack S. Kilby** mientras trabajaba para la empresa de tecnología **Texas Instruments** desarrolló el primer **circuito integrado** (también conocido como **chip** o **microchip**). Consistía en la integración de seis transistores en un mismo dispositivo. En el año 2000 Kilby fue galardonado con el Premio Nobel de Física por la enorme contribución de su invento al desarrollo de la tecnología.²⁴

La capacidad de producción masiva de circuitos integrados, su confiabilidad y la facilidad de agregarles complejidad, llevó a su estandarización, reemplazando circuitos completos con diseños que utilizaban transistores discretos, y además, llevando rápidamente a la obsolescencia a las válvulas o tubos de vacío.

Definición 2.3 Un **chip** o circuito integrado (CI) o microchip, es una estructura de pequeñas dimensiones de material semiconductor, normalmente silicio, sobre la que se fabrican circuitos electrónicos mediante fotolitografía y que está protegida dentro de un encapsulado de plástico o de cerámica.

Gracias a este avance, las computadoras ya no requerían enormes cuartos para su instalación, ni pesaban toneladas. Asimismo, el bajo costo de los chips volvía a la computadora un dispositivo asequible por “cualquier” persona (seguían siendo equipos caros, pero el costo ahora era equiparable al de un vehículo automotor).

2.3 La modernidad de las computadoras

En la década de 1970 surge un quiebre en la historia de las computadoras. La invención del chip, como elemento de bajo costo, hace que entusiastas y empresas sean capaces de armar sus propios equipos.

Con la reducción de costos, la estandarización del código gracias a lenguajes de programación, y una economía en auge, decenas de empresas comenzaron a producir computadoras (o en su defecto componentes y software para computadoras). Muchas de esas empresas hoy son líderes en la industria informática.

Por otro lado, el desarrollo de ideas democratizadoras de tintes socialistas llevó al desarrollo de software por parte de comunidades auto-convocadas, sin fines lucrativos, sino sociales.

Todos estos elementos son los que llevan al estado actual de las computadoras y de la informática, transformando a estos dispositivos en una parte fundamental de las sociedades modernas.

2.3.1 Las primeras microcomputadoras

A mediados de la década de 1970, comenzaron a aparecer las primeras **microcomputadoras**, equipos de tamaño sumamente reducido en comparación a las grandes computadoras que dominaban la escena de la época (aunque grandes para los estándares actuales).

La primer microcomputadora fue la japonesa **SMP80/08** de Sord Computer Corporation (1972), a lo que siguió la francesa **Micral N**. En 1974 aparece en los Estados Unidos el **Altair 8800**, que era vendido como un conjunto que debía ser ensamblado por el usuario.

Sabías qué

La Altair 8800 solamente contaba con unos pocos interruptores e indicadores lumínicos, y no tenía ni pantalla, ni teclado, ni nada parecido.

La revolución de los circuitos integrados y sus reducidos costos hizo que comenzaran a aparecer clubes de entusiastas y amantes de la electrónica y la tecnología, que se juntaban para compartir piezas, intercambiar planos, conocimiento, diseños y software. Aparece en esta época el **microporcesador**. Sumamente económicos, consistían en un complejo circuito integrado que cumplía múltiples funciones, antes realizadas por varios chips. Los **Intel 8080**, los **Z80** y los **Motorola 6800** serían los procesadores que potenciarían toda una generación de computadoras y consolas de videojuegos, transformándose en emblemas de la era.³

Definición 2.4 Un **microporcesador** es el circuito integrado central más complejo de un sistema informático. Es el encargado de ejecutar los programas, desde el sistema operativo hasta las aplicaciones de usuario. Puede contener una o más unidades centrales de procesamiento (CPU).

El **Homebrew Computer Club**, uno de estos clubes de entusiastas más influyentes y conocidos surgió en 1975 en **Silicon Valley, California**. Fue considerado como “el crisol de una industria entera”, pues de allí surgieron influyentes personajes, e incluso empresas. En particular, **Apple** surge de los trabajos realizados en ese club por parte de **Steve Jobs** y **Steve Wozniak**.

También sería el lugar a donde **Bill Gates**, fundador de **Microsoft** enviaría una famosa carta justificando la privatización y comercialización del software mediante el actual modelo de licencias. Gates alegaba que el software no debía ser compartido (práctica habitual en esa época) sino que



Steve Jobs (parado) y Steve Wozniak (sentado) miembros del Homebrew Computers Club diseñando su primer computadora, la Apple I.
Fotografía de Apple Computer.

los usuarios debían pagar un permiso (licencia) para tener derecho a su uso. La carta fue tomada con gran aceptación y permitió que el desarrollo de software se transformara en una industria millonaria.

En los 80s llega el boom de las computadoras comerciales y decenas de microcomputadoras aparecen en el mercado, como la **Apple II**, la **Commodore 64**, la **BBC Micro**, o **TRS 80**. También aparecen en la escena las primeras consolas de videojuegos, como la **Atari 2600**, la **ColecoVision**, la **Intellivision** o la **Magnavox Odyssey**.

El público comenzaría a acostumbrarse a estos dispositivos, y a demandar mayor producción y más características. Las computadoras pasarían a ser un elemento fundamental en muchas industrias y negocios, así como un valioso recurso en instituciones educativas.

2.3.2 Los primeros sistemas operativos

La mayoría de las primeras microcomputadoras no permitían al usuario hacer otra cosa que no fuera de programar. La mayoría traía incluido un “manual de usuario” que enseñaba a los usuarios a programar en el equipo, el cual, generalmente incluía el lenguaje de programación BASIC. Esto hacía que los usuarios requirieran aprender una serie de conocimientos para nada sencillos solo para poder operar la unidad. Eso si, los conocimientos aprendidos permitían realizar cualquier cosa con el equipo, librado solo a la imaginación del usuario y a las capacidades técnicas del hardware.

Sin embargo, este modelo traía aparejado la problemática de un bajo número de usuarios, y una resistencia por parte de empresas e instituciones de adoptar computadoras debido al alto costo de capacitación del personal. Con la intención de solventar ese problema surge la idea de los **sistemas operativos**.



CP/M corriendo sobre un monitor de fósforo verde en una Sanco 8001.
Fotografía de Mspetch.

Un **sistema operativo** consistía en un programa que corría de forma constante en la computadora, y permitía al usuario escribir instrucciones que requeriera de la máquina, y esta las llevaría adelante. Así, un usuario podía solicitarle que realice un cálculo, ejecute un juego, o lo que requiriera del equipo, sin necesidad de aprender a programar. Esto requería por supuesto que la computadora incluyera software capaz de llevar adelante muchas tareas genéricas. Así, aparecen los primeros programas como procesadores de documentos, planillas de cálculo y bases de datos.³

En las grandes computadoras que existían en las universidades y entidades gubernamentales, el sistema operativo por defecto era **UNIX**, creado por **Bell Labs (AT&T)**. El sistema se distribuía de forma libre en un inicio, pero luego AT&T decidió comenzar a vender el producto. Esto llevó a que algunas universidades buscaran reemplazar UNIX con otra cosa.

En las microcomputadoras aparecen sistemas como **CP/M**, creado por **Gary Kildall**, y **DOS** (Disk Operating System), que en realidad no es un sistema operativo, sino un conjunto de sistemas, pues cada fabricante solía crear el suyo propio, y comercializarlo junto con su computadora (Y no eran necesariamente compatibles entre si). Tal vez el más popular sea de estos últimos sea **MS-DOS** de **Microsoft**. **MS-DOS** supo ganarse al mercado gracias a que era el sistema por defecto de las computadoras **PC** (Personal Computer). PC era una gama de modelos de computadoras vendidas por **IBM**, de gran éxito comercial por poseer **partes fácilmente intercambiables**, y por licenciar el diseño de componentes a diversas empresas que creaban clones económicos de muchos componentes.

Apple lanzaría en 1984 la **Apple Macintosh**, un modelo de computadora que fue el primero en venir de serie con un sistema operativo que incluía un entorno gráfico y mouse. Si bien no fue el primer sistema operativo gráfico (ya se habían hecho varios experimentos en diversos sistemas), al incluirlo de serie, puso la idea en el aire de que ese era el futuro, y pronto fue copiado por otros. Por su parte, la primera versión de **Windows**, el sistema operativo de **Microsoft** que hoy sigue siendo comercializado, vería la luz en 1985.

Sabías qué

Apple tomó la idea del mouse y de la interfaz gráfica de experimentos que se realizaban en el Centro de Investigación de Palo Alto de Xerox, conocido como **Xerox PARC**. Este centro fue la cuna de la computadora personal como hoy la conocemos, además de otros inventos como el mouse, la interfaz gráfica, la metáfora de escritorio, la impresión láser, las redes ethernet, la programación orientada a objetos, la computación ubicua, entre otras tantas cosas.²⁵

Para mediados de los 80 prácticamente no se comercializaban computadoras que no tuvieran un sistema operativo pre-instalado. Algunas venían incluso con más de uno, permitiendo al usuario elegir su sistema de preferencia. Ningún sistema mantenía la hegemonía total, por lo que todavía había un mercado para el desarrollo de nuevos sistemas operativos. Así surgen sistemas como **Amiga OS**, o **BeOS**, entre otros.

2.3.3 El Software Libre

Una consecuencia directa de la masificación del uso de computadoras fue la **privatización del software**. El software pasa a considerarse una obra intelectual, de forma similar a un libro, una canción o una película. Es decir, el **software pasa a tener derechos de autor (copyright)**.

Ademas, por las características técnicas del software, tales como la posibilidad de replicar el mismo en tantas máquinas como se quiera, o de utilizarlo con fines que al autor no considera apropiados, hacen que el software no se comporte como los libros, música o películas de la época.

Así, el **software no se vende, se licencia**. Es decir, uno compra la posibilidad de utilizar el software de una forma determinada (una cantidad específica de equipos, usarlo con determinados fines, compartirlo de determinada forma, etc.).

Definición 2.5 Una **licencia de software** es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribución) y el licenciatario (usuario consumidor, profesional o empresa) del programa informático, para utilizarlo cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.

Esta práctica pasa a ser el estándar en el desarrollo de software de la época. Sin embargo, en 1985, un programador llamado **Richard Stallman**, cansado de problemas asociados al copyright del software, inicia un movimiento que involucra tanto ideas de izquierda como cuestiones técnicas asociadas al software, el **movimiento de software libre**. Poco tiempo después fundaría la **Free Software Foundation** y el **proyecto GNU**, con la intención de promover la creación y el uso del software libre.²⁶

Stallman acuña el término **copyleft**, como una alternativa al **copyright**. En este caso, una obra intelectual sigue teniendo derechos de autor, pero el autor manifiesta al momento de su creación de forma explícita su voluntad de permitir a otros utilizar su obra para lo que sea, siempre y cuando esta se mantenga libre (es decir, que los derivados de dicha obra comparten la misma característica de poder ser utilizados como se desee).

Esto supone un cambio radical en la industria del software. Involucra el compartir el **código fuente** de los programas, y permitir que cualquiera trabaje sobre él. Stallman sostiene que esto permite que la sociedad en su conjunto se beneficie de las mejoras que se vayan realizando.

Las empresas en principio juzgaron fuertemente al movimiento, pero las bondades del mismo pronto se hicieron evidentes. Al tener cientos de programadores que participan de forma voluntaria en un producto, el mismo termina por contar con mayor calidad y siendo más seguro. Así, aparece la **Open Source Initiative**, una iniciativa que comparte las prácticas del movimiento del software libre, pero, cuyos fines no son los mismos; mientras el primero persigue fines altruísticos, el segundo solo busca una mejora en la rentabilidad y beneficios comerciales.

Se dice en general que el software es **libre** si sus creadores adhieren a las políticas propuestas por la **Free Software Foundation**, y que es **open source** si simplemente abren su código pero no comparten la filosofía. El resto del software se denomina **privativo**. En muchas ocasiones no se realiza la distinción entre los términos libre y open source, considerando a todo el software de código abierto como libre.

Hay que tener en cuenta que el software libre es general, pero no necesariamente, gratuito. Esto es uno de los principales incentivos para los usuarios, pero a esto se le debe adicionar la confiabilidad en el software, pues su código ha sido visto por muchos desarrolladores que no tienen interés comercial en el mismo de forma directa, y que denunciarían cualquier acción que pudiera perjudicar al usuario final.



Richard Stallman en una charla en Madrid (2016).
Fotografía de Rubén Ojeda.

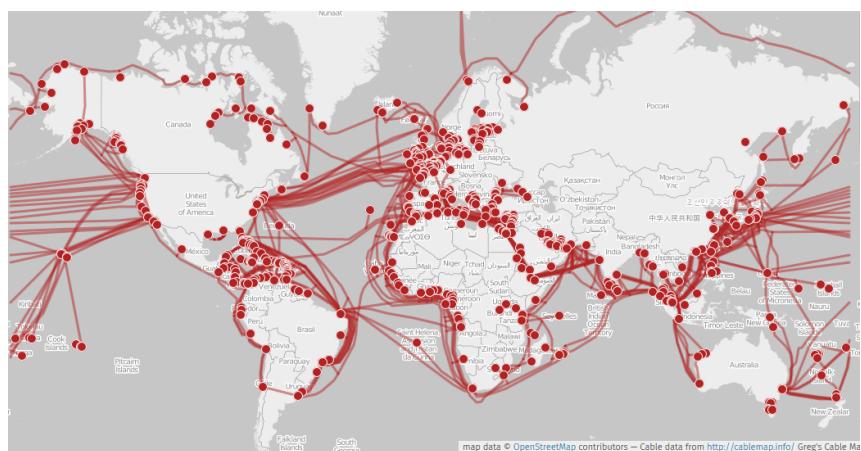
Hoy en día, la mayoría de los programas comerciales cuentan con una alternativa “libre”. En muchas ocasiones, la alternativa libre es superior a la “privativa”, pero en otras ocasiones, la falta de financiamiento y prácticas monopólicas hacen que las alternativas libres no estén a la altura de sus contrapartes comerciales.

Este movimiento se ha vuelto tan grande que incluso empresas como Microsoft o IBM ahora desarrollan y apoyan el software libre u open source. Muchas empresas se dedican incluso a la venta de servicios de soporte o desarrollo de soluciones de software libre, por lo que el software libre no representa una amenaza a la industria del desarrollo de software.

2.3.4 Las redes, Internet, y la globalización

No tardó mucho tiempo en que los usuarios quisieran compartir información entre computadoras ubicadas en distintos lugares físicos. Ya en 1950, el sistema de radares militares de Estados Unidos, conocido como **SAGE**, se puso en línea, conectando computadoras de diversos puntos del país. Por su parte, en la Unión Soviética en 1959 se propone un plan de defensa que incluye una red de computadoras ubicadas en lugares estratégicos.

Sin embargo, el verdadero hito en materia de redes se produciría en 1969, con la creación de **ARPANET** (Advance Research Projects Agency Network). Nacida como un proyecto del Departamento de Defensa de los Estados Unidos, ARPANET sería la primer red en utilizar un protocolo estandarizado para comunicar diversas computadoras. Empezó uniendo los equipos de diversas universidades en la costa oeste, para expandirse velozmente hacia la costa este y centro del país.



Mapa de las conexiones submarinas de Internet.
Datos de Greg Mahlknecht sobre mapa de OpenStreetMap.

En Europa, diversas organizaciones generaban sus propias redes, algunas de las cuales colaboraban con la ARPANET adoptando sus protocolos, como **NORSAR** en Noruega, o redes en las universidades de Inglaterra. ARPANET evolucionaría rápidamente para convertirse en lo que hoy llamamos **Internet**.²⁷

Sabías qué

Internet se basa en una red de computadoras conectadas por cables. Hay cables submarinos que conectan todos los lugares del mundo, así como cables enterrados bajo la tierra en diversos países que no limitan con algún océano.

Muchas personas creen que Internet funciona mediante conexiones satelitales. Si bien es cierto que existen conexiones de estas características, las velocidades de conexión y los altos costos de mantener ese servicio, lo vuelven inviables para un servicio tan masivo.

2.4 La actualidad y el futuro

Avanzamos mucho desde esas primeras microcomputadoras. Hoy, es normal que en los países desarrollados y emergentes la mayoría de la población cuente con una computadora. Además, la penetración de internet en la población de dichos lugares suele ser elevada o encontrarse en constante aumento, tal es el caso de Argentina.

Sabías qué

En algunos países el acceso a internet es considerado un derecho constitucional. En esos lugares entienden que el acceso a la información debe ser irrestricto, y por tanto, el estado debe garantizar alguna forma de que las personas accedan a Internet.

Los dispositivos se han reducido en tamaño y aumentado en potencia, pero siguen basados en los mismos principios que las microcomputadoras de los 70. Un celular moderno es incluso cientos de veces más potente que esas primigenias computadoras que, no obstante, llevaron al hombre a la luna.

Nuevas tecnologías como el GPS, las redes inalámbricas, las conexiones de datos móviles, entre otras, nos posicionan en mundo donde la tecnología es ubicua. Además, nuevos avances, aún en desarrollo, pero ya una realidad, nos plantean oportunidades, a la vez que desafíos de cara a un futuro donde la tecnología estará cada vez más integrada en nuestro día a día.

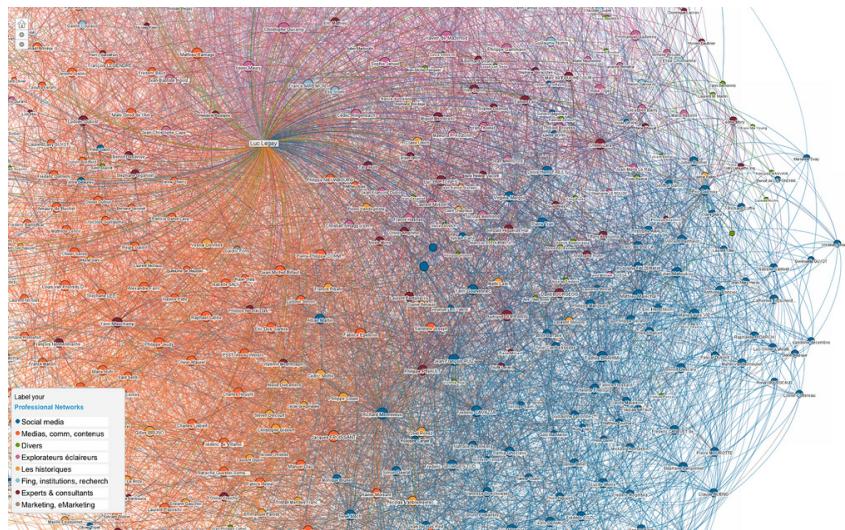
La concepción de la información como un elemento fundamental del universo, inherente a la física o a la biología, plantea nuevos modelos computacionales que presentan inicialmente cambios radicales en la forma en la que pensamos en computación, así como también auguran avances en áreas como la física, la astronomía o la medicina.

2.4.1 La internet de las cosas y Big Data

Uno de los escenarios en donde hay cada vez más desarrollo y más trabajos en ciencias de la computación tiene que ver con el concepto de “**Internet de las cosas**” (IoT). Es un concepto que se refiere a una interconexión digital de objetos cotidianos con internet.²⁸

Esto implica que virtualmente todo objeto tenga algún tipo de identificador de radiofrecuencia que le permita al dispositivo conectarse a Internet. Con objeto nos referimos a autos, heladeras, tostadoras, lámparas, libros, paquetes, envoltorios, etc. Virtualmente todo. Kevin Ashton, quien acuñó el término expresa los siguientes:

“Las computadoras actuales —y, por tanto, internet— son prácticamente dependientes de los seres humanos para recabar información. Una mayoría de los casi 50 petabytes de datos disponibles en internet fueron inicialmente creados por humanos, a base de teclear, presionar un botón, tomar una imagen digital o leer un código de barras. Los diagramas convencionales de internet, dejan fuera a los routers más importantes de todos: las personas. El problema es que las personas tienen un tiempo, una atención y una precisión limitados, y no se les da muy bien conseguir información sobre cosas en el mundo real. Y eso es un gran obstáculo. Somos cuerpos físicos, al igual que el medio que nos rodea. No podemos comer bits, ni quemarlos para resguardarnos del frío, ni meterlos en tanques de gas. Las ideas y la información son importantes, pero las cosas cotidianas



Visualización de las conexiones del usuario Luc Legay en la plataforma LinkedIn.

Imagen de Luc Legay.

tienen mucho más valor. Aunque, la tecnología de la información actual es tan dependiente de los datos escritos por personas que nuestras computadoras saben más sobre ideas que sobre cosas. Si tuviéramos computadoras que supieran todo lo que tuvieran que saber sobre las 'cosas', mediante el uso de datos que ellas mismas pudieran recoger sin nuestra ayuda, nosotros podríamos monitorizar, contar y localizar todo a nuestro alrededor, de esta manera se reducirían increíblemente gastos, pérdidas y costes. Sabríamos cuándo reemplazar, reparar o recuperar lo que fuera, así como conocer si su funcionamiento estuviera siendo correcto. El internet de las cosas tiene el potencial para cambiar el mundo tal y como hizo la revolución digital hace unas décadas. Tal vez incluso hasta más.” Ashton [29]

La internet de las cosas no es un escenario futurista lejano, pues distintas estimaciones proyectan que para 2020 ya habrá aproximadamente unos 20 mil millones de dispositivos conectados, siendo la cifra actual de unos 9 mil millones.³⁰

Este tipo de interacciones con dispositivos tendría implicancias prácticas en áreas como la **domótica** (automatización de hogares), el **marketing**, y la **logística**. Sin embargo, a medida que la tendencia crece surgen diversas preocupaciones y problemáticas. Por un lado, aspectos relacionados a la seguridad de estos dispositivos, pues se ha comprobado que muchos de ellos pueden ser vulnerados con facilidad, algo que sucedió incluso con una cámara de monitoreo para bebés, pudiendo el atacante tener acceso a la filmación. Por otro lado, la cantidad de datos a manejar representa un volumen significativo, lo que trae aparejados desafíos tanto a nivel de ingeniería (servidores, energía, etc.) como a nivel de desarrollo de software (manejando grandes volúmenes de datos).

Así, esta área va de la mano con el desarrollo de **Big Data**, un concepto que hace referencia a conjuntos de datos tan grandes y complejos que requieren de aplicaciones informáticas no tradicionales de procesamiento de datos para tratarlos adecuadamente. El análisis de los datos permite elaborar complejas estadísticas o predicciones.³¹

Pero el Big Data no solo se produce a partir del internet de las cosas, sino que otros datos, por ejemplo, aquellos generados por seres humanos al utilizar tecnología también son sujetos de esta disciplina. Hoy en día empresas como Google o Facebook recolectan y venden una gigantesca

cantidad de datos de sus usuarios, que involucran sus gustos, sus conocidos, su ubicación en distintos momentos, entre otras muchas cosas, y que se utilizan para determinar la publicidad más adecuada para cada segmento, así como para hacer predicciones sobre su comportamiento y consumo. No solo es publicidad, sino que también son utilizados esos datos para manipular las elecciones en diferentes países.³²

2.4.2 La inteligencia artificial y la automatización

Otra de las áreas en donde hay un especial interés dado su amplio potencial es en la **inteligencia artificial**. Coloquialmente, el término se aplica cuando una máquina imita las funciones “cognitivas” que los humanos asocian con otras mentes humanas, como por ejemplo: “aprender” y “resolver problemas”. Técnicamente puede definirse como la capacidad de un sistema para interpretar correctamente datos externos, para aprender de dichos datos y emplear esos conocimientos para lograr tareas y metas concretas a través de la adaptación flexible.³³

La inteligencia artificial permite **automatizar** procesos previamente realizados por humanos. Dentro de las aplicaciones prácticas ya destacan actualmente la participación de robots automatizados en procesos industriales, o de inteligencias artificiales en el mercado de valores, con resultados que superan ampliamente a los que podrían obtener cualquier grupo de seres humanos. También se utilizan sistemas de inteligencia artificial en procesos de filtrado de spam en correos electrónicos, o generación de listas musicales, así como en videojuegos. Hoy en día incluso hay empresas que ofrecen para empresas de noticias, servicios de inteligencia artificial con autómatas capaces de redactar notas, por ejemplo, de eventos deportivos, basándose solamente en las estadísticas del juego. También existen sistemas que permiten eliminar puestos de gerencia media, al identificar el trabajo realizado por estos y reemplazarlos por sistemas automáticos. Ya existen también, autos capaces de manejarse solos, y de tomar la mejor decisión de conducción ante situaciones imprevistas. Se espera cada vez más y más intervención de este tipo de tecnologías, en transporte, en medicina, en telecomunicaciones, en política, etc. Su incorporación será tan grande que se habla de una “**cuarta revolución industrial**”, la cual cambiará significativamente el panorama económico y laboral del planeta, al punto de que se estima que el un alto porcentaje de la población quedará sin empleo, aunque también se pronostica la aparición de nuevos puestos de trabajo asociados a estas tecnologías.³⁴

2.4.3 La computación cuántica y la computación biológica

La última área de interés que mencionaremos está relacionado a nuevos modelos computacionales que vienen a romper el paradigma de computadora actual. En ese sentido hay dos grandes modelos que llenan de emoción a los científicos.

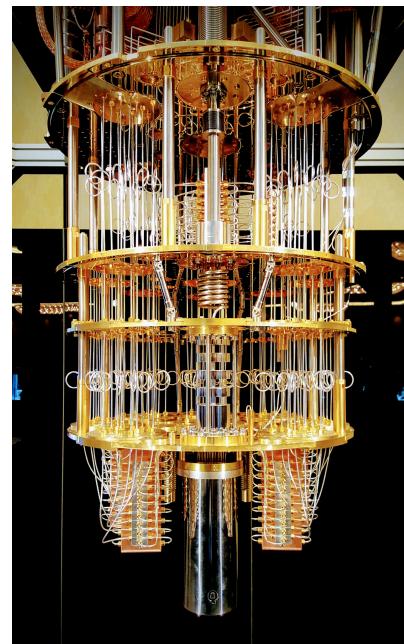
El primero, la **computación cuántica**, sería capaz de resolver de forma eficiente algunos problemas que no pueden resolverse con computadoras clásicas, permitiendo la implementación de nuevos algoritmos. Los avances tanto en la parte formal de esta ciencia, así como en la ingeniería de la misma, auguran un gran futuro para este tipo de computadoras. Las mismas utilizan principios de la física cuántica, tales como la superposición o el entrelazamiento cuántico para manipular información, la cual está codificada como alguna propiedad cuántica de una partícula (por ejemplo la polarización de un fotón, o el spin de un electrón).³⁵

Las primigenias computadoras cuánticas ya están entre nosotros. Compañías como IBM, Microsoft y Google ya tienen desarrollos sobre esta tecnología. Sin embargo, todavía son propensas a errores, y lo que ofrecen puede realizarse mediante simulaciones en computadoras clásicas, por lo que todavía no ofrecen mejoras sustanciales. Sin embargo, la tecnología avanza rápidamente y se

espera que en pocos años pasen a ser una realidad.³⁶

El segundo modelo es el de las **computadoras biológicas**. Este tipo de computadoras se hacen posibles gracias a la nanobiotecnología, y permitirían manipular proteínas y ADN para realizar cálculos computacionales. Este tipo de computadoras permitiría manipular elementos biológicos, generando complejas interacciones, lo cual podría resultar en nuevos medicamentos o tratamientos para enfermedades.³⁷ No solo eso, sino que la capacidad de los sistemas biológicos de autorePLICARse permitiría crear computadoras con un costo sumamente bajo, lo cual sería una ventaja económica sustancial frente a sistemas que requieren procesos manuales para su fabricación.

Estos nuevos modelos computacionales surgen de la combinación de las ciencias de la computación con otras disciplinas (física y biología respectivamente). Este enfoque nos demuestra que las ciencias de la computación están todavía en una etapa embrionaria, pues la interpretación de cualquier elemento como “información” da como resultado estas tecnologías que nos obligan a repensar seriamente las posibilidades y los límites de la computación. Por esto muchos científicos se emocionan con las posibilidades que auguran estas nuevas tecnologías, aunque muchas de ellas sean todavía teóricas.



Parte de la computadora cuántica Q de IBM.

Fotografía de Lars Plougmann.

2.5 Actividades

Esta sección presenta una serie de ejercicios y preguntas para que realice. En este caso, se pide investigue en Internet para poder contestar las preguntas a continuación:

Ejercicio 2.1 ¿Cuál fue la primer computadora en argentina? ¿Para que se usaba? ¿Quién la trajo al país?

Ejercicio 2.2 ¿Qué es la ley de Moore? ¿Se cumple actualmente?

Ejercicio 2.3 ¿De quién son los cables submarinos de Internet?

Ejercicio 2.4 Sabía que en Argentina se desarrollaron varias distribuciones de Linux. Averigüe el nombre y la historia de algunas de ellas.

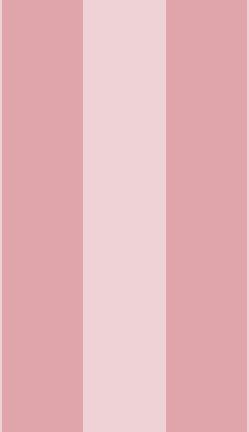
Ejercicio 2.5 ¿Conoce alguna de las microcomputadoras emblemática? ¿Cuál? Si no conoce ninguna, investigue que computadoras marcaron esa época.

Ejercicio 2.6 ¿Qué lenguajes de programación conoce? ¿Sabe cuales son los más populares?

Referencias

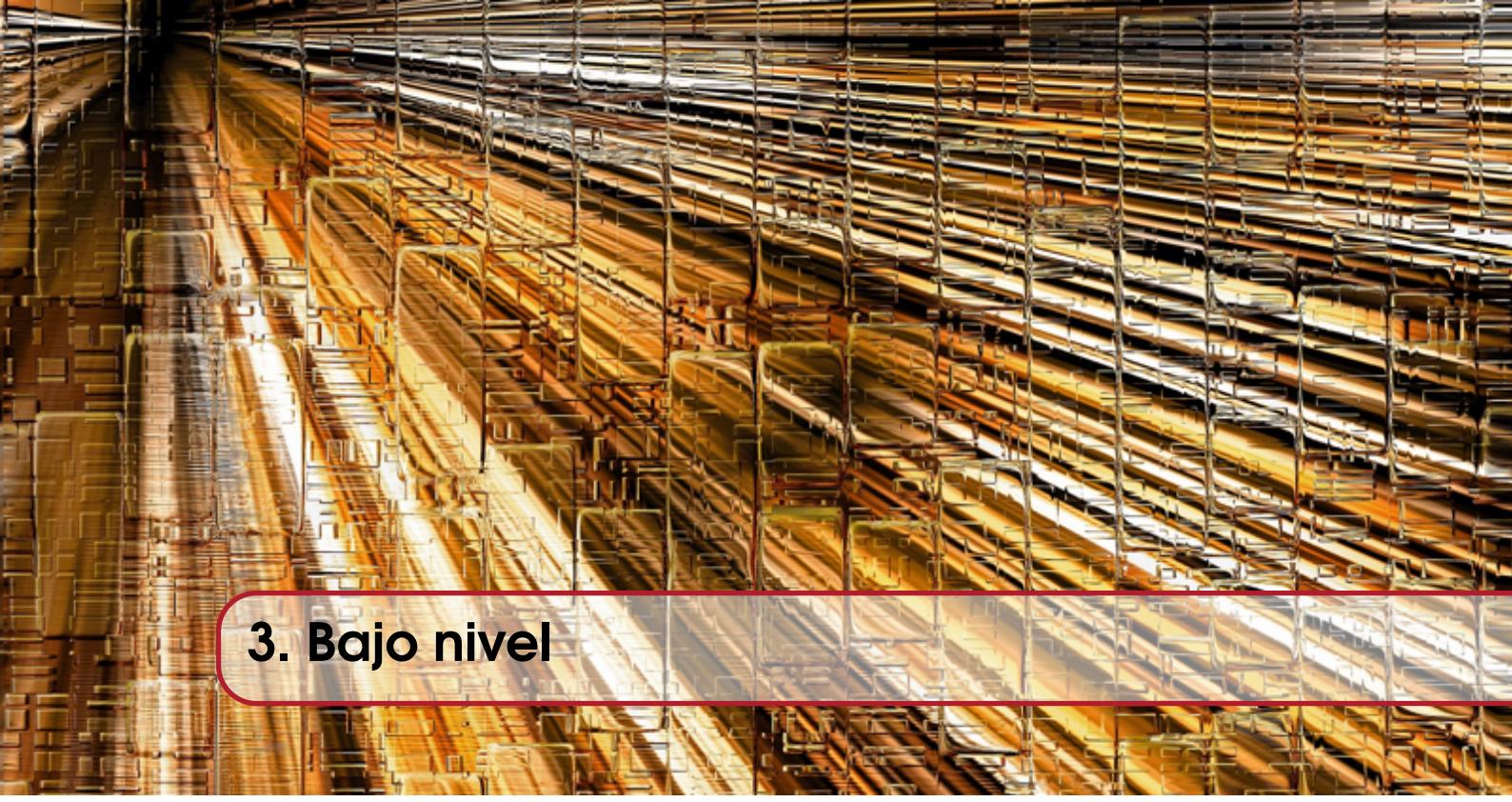
- [1] Dan Gookin. *PCs For Dummies*. 10^a ed. For dummies. Hoboken, Nueva Jersey: Wiley, 2005 (véanse páginas 15, 16, 22, 28).
- [2] J Clark Scott. *But How Do It Know? - The Basic Principles of Computers for Everyone*. John C. Scott, 2009 (véase página 15).
- [3] Paul E. Ceruzzi. *A History of Modern Computing*. 2^a ed. London, England; Cambridge, Massachusetts: MIT Press, 2003 (véanse páginas 16, 37, 41, 43, 45).
- [4] Dan Gookin. *Buying a PC For Dummies*. 2006 ed. For dummies. Hoboken, Nueva Jersey: Wiley, 2006 (véase página 17).
- [5] Philip A. Laplante. *Dictionary of Computer Science, Engineering and Technology*. CRC Press, 2000 (véanse páginas 17-19, 23).
- [6] Doug Lowe. *Networking for Dummies*. 7^a ed. For dummies. Wiley, 2004 (véase página 21).
- [7] Douglas Downing y col. *Dictionary of computer and Internet terms*. 10^a ed. Barron's business guides. Hauppauge, Nueva York: Barron's Educational Series, 2009 (véanse páginas 22, 23).
- [8] Firmware. *Merriam-Webster's dictionary*. <https://www.merriam-webster.com/dictionary/firmware>, 2018 (véase página 23).
- [9] Andrew S. Tanenbaum. *Modern Operating System*. 2^a ed. Prentice Hall, 2007 (véase página 24).
- [10] Andrew S. Tanenbaum y Albert S. Woodhull. *Operating Systems: Design and Implementation*. 3^a ed. Prentice Hall, 2006 (véase página 24).
- [11] Thomas M. Lenard. «Competition, Innovation and the Microsoft Monopoly: Antitrust in the Digital Marketplace». En: *Proceedings of a conference held by The Progress & Freedom Foundation in Washington, DC February 5, 1998*. Editado por Thomas M. Lenard y Jeffrey Eisenach. Springer Netherlands, 1999 (véase página 25).
- [12] Aplicación. *Diccionario de la lengua española*. Madrid, España, 2004 (véase página 26).
- [13] Jonas Bogoshi, Kevin Naidoo y John Webb. «The oldest mathematical artifact». En: *The Mathematical Gazette* 71.458 (1987), página 294 (véase página 32).
- [14] Carl B. Boyer y col. *A history of mathematics*. 2^a ed. Wiley, 1991 (véase página 32).
- [15] Jian-Liang Lin y Hong-Sen Yan. *Decoding the Mechanisms of Antikythera Astronomical Device*. Springer, 2016 (véase página 32).
- [16] Clair L. Evans. *Broad Band: The Untold Story of the Women Who Made the Internet*. Penguin, 2018 (véase página 33).
- [17] George C. Chase. «History of Mechanical Computing Machinery». En: *IEEE Annals of the History of Computing* 2.3 (1980), páginas 198-226 (véase página 34).
- [18] Fuegi J. y Francis J. «Lovelace & Babbage and the creation of the 1843 'notes'». En: *IEEE Annals of the History of Computing* 25.4 (oct. de 2003), páginas 16-26 (véase página 35).
- [19] Charles Petzold. *Annotated Turing*. Wiley, 2008 (véase página 36).
- [20] Paul E. Ceruzzi. *Computing: A Concise History*. Cambridge, Massachusetts: MIT Press, 2012 (véase página 38).
- [21] David A. Patterson y John L. Hennessy. *Computer Architecture: A Quantitative Approach*. 3.^a edición. Morgan Kaufmann, 2002 (véase página 39).

- [22] Emerson W. Pugh. *Building IBM: Shaping an Industry and Its Technology*. London, England; Cambridge, Massachusetts: The MIT Press, 1996 (véase página 40).
- [23] John Gertner. *The Idea Factory: Bell Labs and the Great Age of American Innovation*. Penguin, 2013 (véase página 40).
- [24] Reid T.R. *The Chip : How Two Americans Invented the Microchip and Launched a Revolution*. HarperBusiness, 2001 (véase página 42).
- [25] Michael A. Hatzik. *Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age*. HarperBusiness, 2000 (véase página 45).
- [26] Peter H. Salus. *The Daemon, the Gnu, and the Penguin*. Reed Media Services, 2008 (véase página 46).
- [27] Katie Hafner y Mathew Lyon. *Where Wizards Stay Up Late: The Origins Of The Internet*. Simon & Schuster, 1998 (véase página 47).
- [28] Christian Floerkemeier y col., edición. *The Internet of Things: First International Conference, IOT 2008, Zurich, Switzerland, March 26-28, 2008. Proceedings*. Lecture Notes in Computer Science 4952 : Information Systems and Applications, incl. Internet/Web, and HCI. Springer-Verlag Berlin Heidelberg, 2008 (véase página 48).
- [29] Kevin Ashton. *That 'Internet of Things' Thing*. <https://www.rfidjournal.com/articles/view?4986>. Jul. de 2009 (véase página 49).
- [30] Gartner Press. *Gartner Says 8.4 Billion Connected 'Things' Will Be in Use in 2017, Up 31 Percent From 2016*. <https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016>. Egham, U.K., feb. de 2017 (véase página 49).
- [31] Pete Warden. *Big Data Glossary*. O'Reilly Media, 2011 (véase página 49).
- [32] «Facebook: Cambridge Analytica habría manipulado las elecciones en Argentina». En: *Perfil* (mar. de 2018). URL: <https://www.perfil.com/noticias/internacional/facebook-cambridge-analytica-habria-manipulado-las-elecciones-en-argentina.phtml> (véase página 50).
- [33] Andreas Kaplan y Michael Haenlein. «Siri, Siri in my Hand, who's the Fairest in the Land? On the Interpretations, Illustrations and Implications of Artificial Intelligence». En: *Business Horizons* 62 (2019), páginas 15-25 (véase página 50).
- [34] Klaus Schwab. «The Fourth Industrial Revolution: what it means, how to respond». En: *World Economic Forum* (ene. de 2016). URL: <https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/> (véase página 50).
- [35] Michael A. Nielsen e Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th anniversary. Cambridge University Press, 2011 (véase página 50).
- [36] Will Knight. «Serious quantum computers are finally here. What are we going to do with them?» En: *MIT technology Review* (feb. de 2018). URL: <https://www.technologyreview.com/s/610250/serious-quantum-computers-are-finally-here-what-are-we-going-to-do-with-them/> (véase página 51).
- [37] Robert A. Freitas. *Nanomedicine Volume I: Basic Capabilities*. Austin, Texas: Landes Bioscience, 1999, páginas 349-351 (véase página 51).



Información

| | | |
|----------|---|-----------|
| 3 | Bajo nivel | 57 |
| 3.1 | Representación de Información | |
| 3.2 | Sistemas de Caja Negra | |
| 3.3 | Actividades | |
| 4 | Informática | 71 |
| 4.1 | Archivos y extensiones | |
| 4.2 | Organización de archivos, directorios y rutas | |
| 4.3 | Programas y Lenguajes | |
| 4.4 | Modelos de comercialización de software | |
| 4.5 | Actividades | |



3. Bajo nivel

Fotografía artística que representa información.

Fotografía de Pixabay.

En este capítulo analizaremos como la computadora transmite y almacena información, y como procesa los datos que tiene almacenados. También veremos como la concepción de “información”, independientemente del tipo de información que se trate y que tan compleja sea, puede abstraerse a una interpretación sencilla capaz de ser procesada rápida y eficazmente por una computadora.

También veremos como las computadoras solamente trabajan con electricidad, y como esta puede ser utilizada para representar todo tipo de información mediante una codificación.

3.1 Representación de Información

¿Cómo hacen las computadoras para hacer todo lo que hacen? ¿Dónde guardan las cosas? ¿Cómo saben? Son preguntas que en general el usuario promedio de computadoras, no versado en la temática, suele hacerse en algún momento.

En esta sección intentaremos responder algunas de esas preguntas, analizando en primer lugar la pregunta “¿Qué es la información?” desde la perspectiva de una computadora. Tras contestar eso veremos como estas complejas máquinas pueden procesar esa información para realizar diversas tareas.

Esto nos llevará a sumergirnos aún más en el mundo del software y en su interacción con el hardware en el nivel más sencillo, para luego ascender en un proceso espiral en los distintos niveles de abstracción que existen en el software.

3.1.1 Códigos y comunicación

Olvidemonos de las computadoras solo por un momento y empiecemos por analizar un poco la comunicación humana. Digamos que queremos comunicarnos con alguien, por ejemplo, hablando. Nosotros seríamos entonces el **emisor** de la comunicación, y la persona con la que hablamos el **receptor**. Además, la comunicación se produce siempre mediante un **canal**, es decir, un medio

físico capaz de transferir la información desde el emisor hasta el receptor (en el caso del habla, el canal es el aire, capaz de propagar las ondas sonoras). Por el canal circula el **mensaje**, que es el elemento de la comunicación que se transfiere desde el emisor hacia el receptor (en el caso del habla, las ondas sonoras).

Ahora bien, no haga trampa y no se adelante en el texto, lea solo lo que tiene inmediatamente a continuación, y responda ¿Puede descifrar el siguiente mensaje alienígena?

○ □ V K K L 1 L D N V K L 1 L K N

Lo más probable es que la respuesta sea que no. Aquí va una pista, cada símbolo que aparece en el mensaje corresponde a una letra. ¿Puede descifrarlo ahora?. ¿Todavía no? Muy bien, la respuesta era:

○ □ V K K L 1 L D N V K L 1 L K N
h o l a a m i g o s y a m i g a s

¿Lo intentamos una vez más? Intente descifrar ahora este mensaje:

K L 1 L K L 1 K

¿Lo logró? Debería haberle sido más sencillo. El mensaje era “amiga mía”. El truco estaba en buscar el mismo símbolo en el mensaje anterior, y luego ir reemplazando cada símbolo con la letra correspondiente.

K L 1 L K L 1 K
a m i g a m i a

Si colocáramos cada símbolo junto a su correspondiente letra, y lo hacemos para todas las letras del abecedario, entonces tendríamos una simple tabla que nos permitiría no solo leer cualquier mensaje que use esos símbolos, sino también escribir mensajes con los mismos.

Esta práctica es algo común que muchos realizamos cuando éramos chicos, como parte de un juego que nos permitía enviarle mensajes secretos a otra persona, sin que terceros (por ejemplo, nuestros padres, o las maestras) pudieran leerlos. El truco consistía en que la tabla de símbolos solo fuera conocida por nosotros y por la persona a quien le íbamos a enviar los mensajes.

Imaginemos ahora dos barcos que se cruzan en alta mar en la noche. Los barcos se encuentran lo suficientemente lejos como para que el sonido de la voz de sus capitanes viaje de un barco a otro, pero sin embargo están lo suficientemente cerca como para ver las luces del otro barco. Así, si los capitanes desean comunicarse, pueden usar la luz (por ejemplo, de una linterna) como canal de comunicación. Un capitán puede enviar mensajes mediante una serie de parpadeos lumínicos que genera con su linterna (los cuales pueden ser largos o cortos) y que el otro capitán podría entender. Pero, para poder comunicarse efectivamente, ambos capitanes deben tener una serie de reglas o convenciones en común que le permitan saber qué significan dichos parpadeos. Un parpadeo corto y uno largo, fácil, la letra “A”, uno corto, uno largo y luego dos cortos, la letra “L”.

Este sistema ideado por Samuel F.B. Morse en 1836, para ser utilizado en sistemas telegráficos.

En general las personas que utilizan código Morse no hablan de parpadeos lumínicos largos y cortos, sino de “rayas” y “puntos”, pues es una convención sencilla para escribir esos parpadeos. Si deseamos comunicarnos basta tener la siguiente tabla a mano:

| letra | código | letra | código | letra | código |
|-------|--------|-------|--------|-------|--------|
| A | ·— | J | ·—— | S | ··· |
| B | —… | K | —·— | T | — |
| C | —·—. | L | ·—.. | U | ··— |
| D | —.. | M | --- | V | ···· |
| E | . | N | —. | W | ·—— |
| F | ··—. | O | ——— | X | —··— |
| G | —··. | P | ·——. | Y | —··— |
| H | | Q | ——·— | Z | ——.. |
| I | .. | R | ·—· | | |

En el caso de los mensajes alienígenas y el juego que realizábamos cuando chicos era que el sistema de reglas que usábamos para leer y escribir el mensaje es secreto, solo nosotros y alguien más lo conocía. En el caso del código Morse en cambio, el código no debe ser secreto, sino todo lo contrario (todos los capitanes de barcos deberían saberlo para que podamos comunicarnos efectivamente).¹

Definición 3.1 Un **código** es un sistema de reglas que permiten convertir información dada por el emisor de un mensaje en alguna forma (letras, símbolos, gestos, parpadeos lumínicos, sonidos u otros) en información relevante para el receptor (probablemente en otra forma).

Pero, que tiene todo esto que ver con las computadoras. Para eso debemos mantener esta idea en la cabeza unos minutos mientras analizamos un poco el hardware de la computadora.

3.1.2 Electricidad y cables

Las computadoras modernas son dispositivos principalmente electrónicos, por tanto solamente pueden manipular la electricidad que fluye por sus cables (principalmente de cobre, como los de las instalaciones eléctricas, pero de tamaños mucho más reducidos y con mucha menos carga eléctrica). La computadora posee una serie de dispositivos que bloquean o no el paso de corriente eléctrica en determinados casos (similar a un interruptor eléctrico, pero que no requiere de intervención mecánica) llamados transistores. También existen en la computadora otra serie de dispositivos electrónicos como resistencias, capacitores, osciladores, etc. Estos se combinan de intrincadas formas para permitir o no el paso de electricidad en determinados cables ante ciertas condiciones, formando lo que conocemos como **circuitos**.

Definición 3.2 Un **circuito electrónico** consiste en un conjunto de componentes electrónicos, tales como resistencias, transistores, capacitores, inductores y diodos, conectados mediante cables o trazas conductivas a través de los cuales fluye corriente eléctrica.

Todas las cuentas y cálculos que la computadora realiza, todas las imágenes que se muestran en pantalla, todo es solamente electricidad. Así **los cables de la computadora son el canal por el que fluye la comunicación**, mientras que **la electricidad que fluye por los cables conforma el mensaje**.

Si queremos comunicarnos con una computadora debemos pensar en el proceso comunicacional

que ocurre. El emisor y el receptor alternan entre usuario y computadora, ya sea que el usuario ingrese datos a esta, esta le brinde información al usuario. Más importante es pensar en el canal y en el mensaje de dicha comunicación.

El único canal existente son los cables que mencionamos, y el mensaje solo puede estar dado por pulsos eléctricos. Muy similar al ejemplo de las linternas, donde solo teníamos “puntos” o “rayas”, aquí solo tenemos electricidad enviada por un cable (o no enviada). Es decir, en un único cable, hay solo dos posibles mensajes en un mismo momento.

Podemos decir que nuestros mensajes están limitados al canal en donde se transmite. Si consideramos más cables, veremos que podemos enviar electricidad por algunos de ellos, mientras que no lo hacemos por otros, dando lugar a una mayor cantidad de combinaciones, que aumenta exponencialmente a medida que agregamos cables. Esto permite enviar combinaciones más complejas, y por tanto mayor cantidad de mensajes en un mismo momento. Nos centraremos más en esto en las siguientes secciones, pero tenga en cuenta el lector que en general, no nos interesa analizar un único cable, sino varios de ellos, pues es cuando se combinan de diversas formas que efectivamente se tiene información interesante.

Siguiendo con los circuitos, los mismos **pueden tener formas sumamente complejas que permiten realizar operaciones de lo más variado, como sumas, restas, multiplicaciones y demás**. De esta forma, la información que ingresamos puede ser procesada para transformarse en información que la computadora nos brindará luego.²

Uno de los principales problemas que tienen los circuitos hechos con transistores y cables, es que solo funcionan mientras hay electricidad. Si por ejemplo, se cortara el suministro eléctrico todo lo que hay en la computadora se pierde. Sin embargo, es deseable almacenar información para un uso posterior (por ejemplo, guardar una imagen o un video digital). Así, existen dispositivos que utilizan otros elementos para almacenar información, conocidos como **periféricos de almacenamiento**. Todos estos dispositivos son capaces de tomar la electricidad que circula por un cable específico (conjunto de cables en realidad, aunque no nos centraremos aquí en el funcionamiento detallado de los dispositivos) y almacenarla de alguna forma tal que luego pueden leer y transformar nuevamente en pulsos eléctricos que viajan por dicho cable.

Los discos rígidos son un ejemplo de periférico de almacenamiento que utilizan una serie de imanes microscópicos que pueden estar orientados hacia alguna de dos direcciones posibles. El disco contiene un cabezal que puede leer la dirección a la que apunta un imán particular, o alterarla. Si lee una cierta dirección, el dispositivo es capaz de mandar un pulso eléctrico (usando energía que proviene de la fuente de alimentación) por alguno de los cables relevantes.

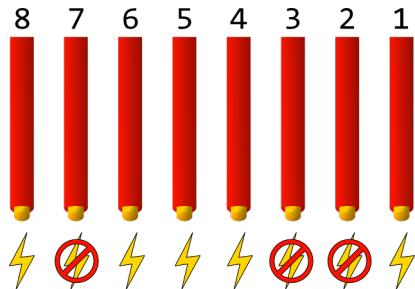
Los discos ópticos (CD/DVD) operan de forma similar, pero con una superficie que refracta la luz, un láser y sensores ópticos en lugar de imanes y un cabezal. Dependiendo del lugar hacia donde se refracte la luz, el dispositivo enviará pulsos eléctricos en determinados cables.

Las tarjetas perforadas hacen algo parecido con punciones sobre una superficie de cartón o papel, para que luego un cabezal intente pasar por un determinado lugar de la tarjeta, trasmitiendo electricidad si se encontraba perforada en ese lugar, pero bloqueando el paso de la misma si no lo estaba.

De esta forma, podemos decir que, **todo la información en una computadora (es decir, el software) no es más que electricidad pasando por cables, transistores y resistencias que se unen en formas extremadamente complejas, formando circuitos, que permiten procesar datos y comunicarse entre usuario y máquina.**³

3.1.3 Código Binario

Imaginemos un conjunto de ocho (8) cables. El pasaje o no pasaje de electricidad por dichos cables hace finalmente al mensaje. Probablemente nos interese mencionar en algún momento, por qué cables pasa electricidad y por cuales no, para determinar cuál es el mensaje que estamos, o bien enviando a la máquina, o bien recibiendo de ella. Podríamos graficar nuestros cables de la siguiente forma:



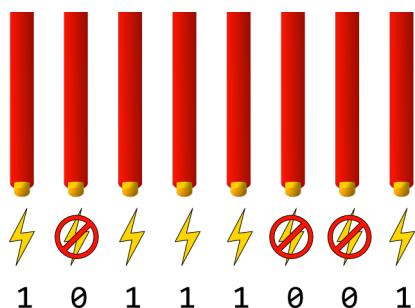
En este caso numeramos los cables del 0 al 7 de derecha a izquierda (podríamos haber numerado del 1 al 8 y de izquierda a derecha). La elegida es una convención arbitraria, pero útil en términos prácticos y por tanto, muy utilizada). Como se puede predecir, cuando la cantidad de cables aumenta de unos pocos a cientos, a miles, realizar un dibujo para indicar en qué cables pasa electricidad y en cuáles no deja de ser viable.

En realidad

En realidad, los cables siempre tienen electricidad, pero el voltaje que pasa por los mismos es distinto. Los circuitos se “activan” cuando el nivel de voltaje supera un umbral (por ej. 1.5 voltios), por lo que todo cable por el que pase menos voltaje se considera “apagado” y por el que pase más “encendido”.

Así es como surge la idea de utilizar alguna especie de **código** para poder hablar del mensaje que se transmite por esos cables, sin tener que realizar complejos dibujos o diagramas. Esto nos permite además abstraernos de los cables y circuitos, es decir, del canal, para centrarnos en el mensaje.

El código elegido para representar el estado de los cables internos de la computadora es el **código binario**, o lo mismo el **sistema binario**. El **sistema binario** es un sistema numérico en el que solamente se utilizan dos símbolos, **cero** y **uno** (nuevamente, una convención, cualquier otro par de símbolos, ej. “A” y “B” hubiera sido perfectamente válido). Así, un **número binario** se compone de una secuencia de ceros y unos (Por ej. 10100111). Como el sistema solo permite dos números, es ideal para representar los estados de los cables en la computadora que trabajan con presencia o ausencia de electricidad. Por lo tanto, podríamos representar el estado de nuestros ocho cables con un número.

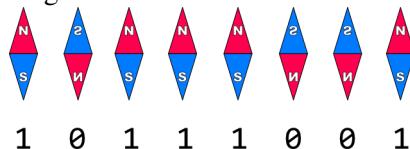


En este caso, elegimos que el dígito 1 representa la presencia de electricidad, mientras que el 0

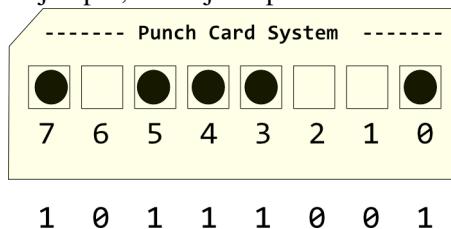
representa su ausencia, pero podría haberse elegido de forma inversa, es irrelevante, pues es solo una convención.

Definición 3.3 El **sistema binario** es un sistema numérico posicional de dos dígitos, los cuales son en general representados mediante cero y uno (0 y 1).⁴

Como ya vimos en la sección anterior, la electricidad que fluye por los cables puede eventualmente almacenarse en un dispositivo adicional, como un disco rígido, que utiliza imanes para representar la presencia o ausencia de electricidad. Si almacenamos en un disco, podríamos representarlo gráficamente de la siguiente forma:



Como vemos, el número binario que representa al valor almacenado es el mismo que el que representa a la electricidad en los cables. Podríamos ver que esto aplica también en otros dispositivos de almacenamiento, como por ejemplo, las tarjetas perforadas:



El sistema binario es lo suficientemente versátil para permitirnos hablar acerca de lo que vamos a almacenar o transmitir por un cable, sin importar el medio físico subyacente. Así, nos permite abstraernos del dispositivo y pensar acerca de la información que fluye en la computadora de forma matemática.

El sistema binario fue estudiado por primera vez en el siglo III A.C. por el matemático indio Pingala, coincidiendo con el descubrimiento del concepto del número cero. El matemático alemán Leibniz documentó en su totalidad el sistema en el siglo XVII, en su artículo “Explication de l'Arithmétique Binaire”. En 1854, el matemático británico George Boole publicó un artículo que marcó un antes y un después, detallando un sistema de lógica que terminaría denominándose “Álgebra de Boole”. La elección de este sistema para representar los datos responde entonces también a la gran cantidad de documentación y trabajos previos sobre dicho sistema numérico.

3.1.4 Codificación de información

Vamos a realizar un juego, similar al que hicimos dos secciones atrás. ¿Se anima a descifrar este mensaje?

72 79 76 65 32 77 85 78 68 79

Por supuesto que a priori es imposible identificar el mensaje, pero si seguimos la misma lógica de antes, podríamos asignar un número a cada letra, y obtendríamos una tabla que nos permita realizar la conversión de uno a otro. Es decir, sabemos que para poder entender ese mensaje, necesitamos un **código**. Dado el código en la siguiente tabla:

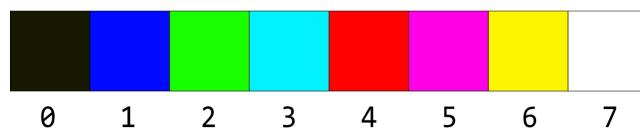
| letra | número | letra | número | letra | número |
|-------|--------|-------|--------|---------|--------|
| A | 65 | J | 74 | S | 83 |
| B | 66 | K | 75 | T | 84 |
| C | 67 | L | 76 | U | 85 |
| D | 68 | M | 77 | V | 86 |
| E | 69 | N | 78 | W | 87 |
| F | 70 | O | 79 | X | 88 |
| G | 71 | P | 80 | Y | 89 |
| H | 72 | Q | 81 | Z | 90 |
| I | 73 | R | 82 | espacio | 32 |

¿Ahora puede identificar el mensaje dado? Claro que si, es cuestión de reemplazar uno a uno los número por las letras dadas.

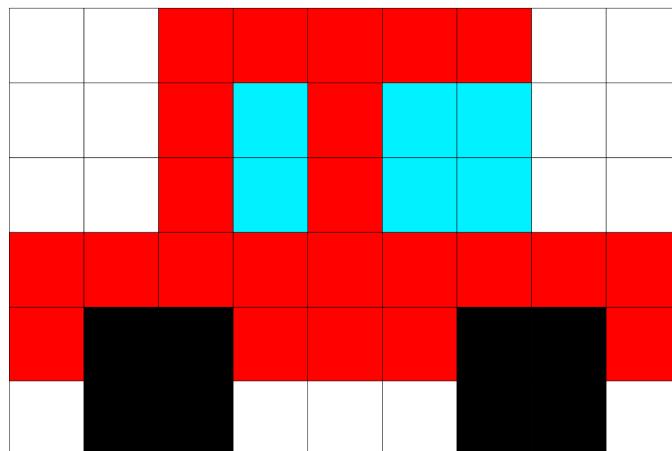
| | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|
| 72 | 79 | 76 | 65 | 32 | 77 | 85 | 78 | 68 | 79 |
| H | O | L | A | | M | U | N | D | O |

Imagínese ahora que la tabla no la tenemos solo dos personas, sino que es una tabla que todo el mundo conoce, y a la que todo el mundo tiene acceso. Si ese fuera el caso, entonces cualquier persona podría escribir un mensaje en forma de números, y cualquiera podría leerlo.

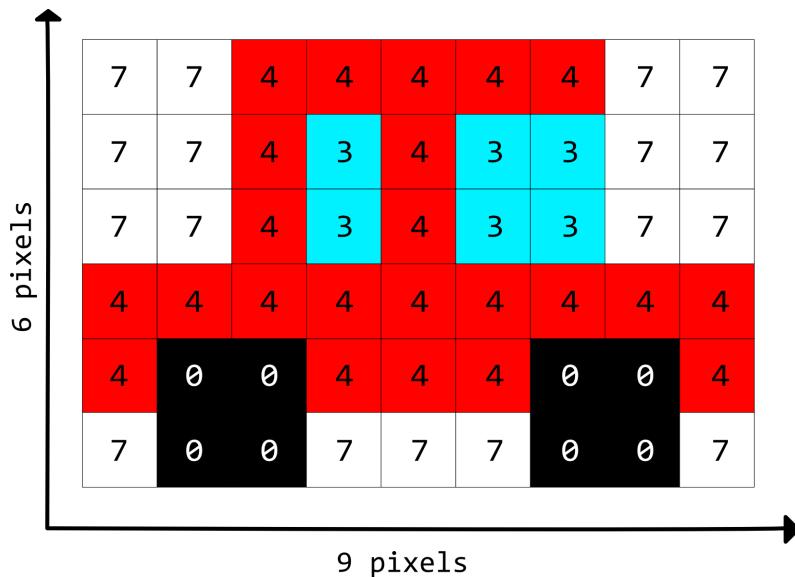
Más interesante aún, podríamos representar otras cosas que no sean letras como números, o como grupos de números, por ejemplo, colores:⁵



Teniendo colores, podríamos representar una imagen como una secuencia de números.



Para representar una imagen, la misma debe ser dividida en una grilla cuadriculada en donde cada celda tiene un único posible color. Luego podemos determinar el color de cada celda haciendo un reemplazo uno a uno con la representación de colores que vimos anteriormente, para obtener una secuencia de números. Además deberíamos indicar el alto y el ancho de la imagen, y elegir el orden en el que vamos a leer la secuencia (por ejemplo, de izquierda a derecha y de arriba a abajo).



Si tomamos los dos primeros dos dígitos como ancho y alto respectivamente, vemos que se indica que la imagen tiene 9 celdas de ancho, por 6 de alto. El resto corresponde a cada uno de las celdas de la grilla, indicando para cada una, que color tiene.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 9 | 6 | | | | | | | |
| 7 | 7 | 4 | 4 | 4 | 4 | 4 | 7 | 7 |
| 7 | 7 | 4 | 3 | 4 | 3 | 3 | 7 | 7 |
| 7 | 7 | 4 | 3 | 4 | 3 | 3 | 7 | 7 |
| 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 4 | 0 | 0 | 4 | 4 | 4 | 0 | 0 | 4 |
| 7 | 0 | 0 | 7 | 7 | 7 | 0 | 0 | 7 |

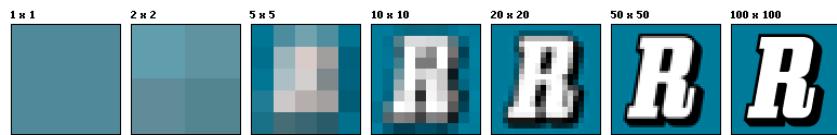
Dispusimos los números en forma de filas y columnas, similar a la imagen, para que sea fácil de leer, pero esto no es necesario. Más aún, como cada color se compone solo de un dígito, podemos eliminar completamente los espacios entre un dígito y otro, transformando nuestra imagen en un gran número.

9677444447777434337777434337744444444400444004700777007

Note que el ancho y el alto no deben ser superiores a 10, para que efectivamente sea entendible por el receptor.

Definición 3.4 Se conoce como **codificación** al proceso mediante el cual una cierta información (por ejemplo, una imagen, un color, una letra) es transformada en algún otro elemento que la representa mediante la utilización de un código. El proceso inverso se conoce como

decodificación.



Ejemplo de como la calidad de imagen aumenta a medida que aumenta su resolución.

Imagen de dominio público.

Sabías qué

Las pantallas de los monitores, celulares y otros dispositivos electrónicos están compuestas de cientos de miles de pequeñísimos puntos conocidos como **pixeles**. Cada **pixel** puede prenderse de un único color (o estar apagado y verse negro). Las imágenes digitales entonces se almacenan guardando la información de que color debe tener cada pixel.

Los sensores de las cámaras digitales funcionan igual. La cantidad de sensores se mide en **megapixeles** y tiene que ver con la cantidad de puntos que van a conformar la imagen.

Las imágenes digitales entonces se almacenan basados en los pixeles, aunque como la cantidad de sensores de la cámara y la cantidad de puntos en la pantalla no siempre coincide, a veces un único pixel de una imagen puede corresponder a varios puntos de la pantalla. La cantidad de pixeles que tiene una imagen es lo que se conoce como **resolución** de la imagen. A mayor resolución, más nítida y de mejor definición, a costa de que ocupa mayor espacio de almacenamiento.

Es decir, cualquier tipo de elemento del que podamos querer hablar es susceptible de ser codificado mediante números.³ Pero eso es solo el principio de un proceso que va más allá, pues todo número puede ser representado mediante binario.

3.1.5 Representando números con binario

El poder representar todo con números es una excelente idea, pero surge un problema al querer guardar esos números en una computadora. Como ya vimos, la computadora solo maneja unos y ceros (o electricidad y no electricidad), no números naturales como los que necesitamos para representar nuestras letras, colores e imágenes.

Por suerte, los números naturales pueden expresarse en binario con poco esfuerzo. El truco consiste en elegir una representación binaria distinta para cada número, de forma similar a lo que hicimos con las letras. Debemos para ello partir de la cantidad de cables que vamos a tener, por ejemplo, ocho cables. Luego, basta analizar todos los posibles estados en los que puede llegar a estar ese conjunto de cables (por ejemplo, todos sin electricidad, todos con electricidad, todos menos el último sin electricidad, todos menos los dos últimos sin electricidad, y así siguiendo). Es decir, analizamos todos los posibles números binarios que representarían a nuestro conjunto de cables. Luego, basta asignar un número a cada binario. Por ejemplo, si tuviéramos cuatro cables tendríamos:

| número | binario | número | binario |
|--------|---------|--------|---------|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | 10 | 1010 |
| 3 | 0011 | 11 | 1011 |
| 4 | 0100 | 12 | 1100 |
| 5 | 0101 | 13 | 1101 |
| 6 | 0110 | 14 | 1110 |
| 7 | 0111 | 15 | 1111 |

Note como la cantidad de cables que tenemos limita la cantidad de números que podemos representar, siendo mayor los números a mayor cantidad de cables. En particular:

$$2^{\text{cant. cables}} = \text{cant. números distintos} \quad (3.1)$$

Por supuesto no es la única representación posible, podríamos haber elegido asignar los números en otro orden. Existen diferentes formas estandarizadas para representar un número en binario, que incluso sirven para números negativos o racionales, algo que no analizaremos pues escapa a los alcances del presente.

Lo importante es que **a toda representación numérica, se puede asignar una representación binaria. Así toda la información digital, ya sea texto, colores, o imágenes, pueden representarse con binario.**³

Una secuencia de números se traduce entonces directamente a una secuencia de números binarios, es decir, a una larga tira de unos y ceros. Y por tanto, cualquier información es posible de ser representada como datos binarios.

3.1.6 Bits, Bytes y otros

Cuando se utilizan los unos y ceros para representar información se habla de **bits**.

Un bit es un dígito en el sistema de numeración binario. Es decir, corresponde a un único cero o uno en una posición determinada. **Si tenemos un dato que está representado por ocho cables (o imanes, o lo que fuera) decimos que está representado por 8 bits.**

Era muy común que las antiguas computadoras trabajaran con 8 bits. Como era muy común la medida de 8 bits para representar datos, surge la palabra **octeto** para indicar precisamente esa cantidad. Adicionalmente surge la palabra **byte**. El **byte** suele representar 8 bits, pero en algunas computadoras antiguas el término se utilizaba con otra acepción (por ejemplo, 6, 7 o incluso 9 bits). Hoy en día la mayoría de las computadoras hablan de **byte para referirse a 8 bits**.

También es muy común escuchar los términos **kilobit** o **megabit**. El prefijo corresponde a una escala adicional que indica una gran cantidad de bits. Así de forma similar a como un kilómetro representa mil metros, un kilobit representa mil bits.

| Nombre | Prefijo | Equivalencia |
|---------|---------|---|
| Kilobit | kbit | 1000 bits |
| Megabit | Mbit | 1000 Kbits = 1.000.000 bits |
| Gigabit | Gbit | 1000 Mbits = 1.000.000.000 bits |
| Terabit | Tbit | 1000 Gbits = 1.000.000.000.000 bits |
| Petabit | Pbit | 1000 Tbit = 1.000.000.000.000.000 bytes |
| Exabit | Ebit | 1000 Pbit = 1.000.000.000.000.000 bytes |

Los bytes también son sujeto de los mismos prefijos, siendo un Kilobyte equivalente a 1000 bytes.⁴

| Nombre | Prefijo | Equivalencia |
|----------|---------|---|
| Kilobyte | kB | 1000 bytes = 8000 bits |
| Megabyte | MB | 1000 KB = 1.000.000 bytes |
| Gigabyte | GB | 1000 MB = 1.000.000.000 bytes |
| Terabyte | TB | 1000 GB = 1.000.000.000.000 bytes |
| Petabyte | PB | 1000 TB = 1.000.000.000.000.000 bytes |
| Exabyte | EB | 1000 PB = 1.000.000.000.000.000.000 bytes |

Para tener una idea de lo importante que son estas medidas, basta con acercarse a un local de hardware para comprar algunos dispositivos. Las memorias de computadoras (conocidas como RAM) tienen su capacidad de almacenamiento expresadas en Gigabytes (GB), siendo hoy en día comunes las de 2, 4 y hasta 8 GB, aunque hay incluso de más. Los discos rígidos también tienen su capacidad expresadas en GB, siendo comunes los de 250, 500, 750, e incluso hay algunos que llegan a ser de 1 TB.

Al contratar un servicio de conexión a internet, la velocidad de conexión se expresa en Megabits por segundo (Es decir, cuantos bits se pueden descargar en un segundo) siendo comunes hoy en día conexiones de 2, 4, 10, 20 y hasta 50 Mbits por segundo.

A continuación se deja una idea de cuanta información (comparada con información impresa) representa cada medida.

| Número de bytes | Múltiplo | Equivalencia aproximada |
|---------------------------|----------|---|
| 1 | 1 B | Una letra. |
| 10 | 10 B | Una o dos palabras. |
| 100 | 100 B | Una o dos frases. |
| 1.000 | 1 kB | Una historia muy corta. |
| 10.000 | 10 kB | Una página de enciclopedia con un dibujo simple. |
| 100.000 | 100 kB | Una fotografía de resolución mediana. |
| 1.000.000 | 1 MB | Una novela. |
| 10.000.000 | 10 MB | Dos copias de la obra completa de Shakespeare. |
| 100.000.000 | 100 MB | Un estante de un metro lleno de libros. |
| 1.000.000.000 | 1 GB | Una camioneta llena de páginas con texto. |
| 1.000.000.000.000 | 1 TB | Páginas con texto elaboradas de 50.000 árboles. |
| 10.000.000.000.000 | 10 TB | La colección impresa de la biblioteca de EE.UU. |
| 1.000.000.000.000.000 | 1 PB | Los datos que maneja Google cada hora. |
| 1.000.000.000.000.000.000 | 1 EB | Todos los datos en Internet para finales de 2001. |

3.2 Sistemas de Caja Negra

En muchas ocasiones es útil pensar los sistemas como una **caja negra**. Es decir, un elemento o proceso del cual no nos importa el funcionamiento interno, pero si sabemos que, ante cierta entrada, producirá cierta salida.

Definición 3.5 Se denomina **caja negra** a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno.

Puede pensarse una analogía al proceso de caja negra con un auto. Cualquier persona puede conducir un auto. Basta comprender que, ante la presión del pedal del acelerador, y habiendo colo-



Ilustración de un proceso de Caja Negra.

Ilustración propia.

cado la palanca de cambios en la posición correcta, el auto irá hacia adelante. Los funcionamientos internos del vehículo, tales como los detalles sobre el funcionamiento de un motor de combustión interna, la batería, o la caja de cambios, no son necesarios desde el punto de vista del conductor.

En informática este proceso se debe llevar a cabo todo el tiempo, y en varios niveles. Pensemos por ejemplo en el CPU, uno de los componentes de hardware más complejos presentes en las computadoras modernas. El mismo está compuesto de millones de transistores colocados e interconectados de intrincadas formas. Si lo que nos interesa es diseñar microprocesadores, probablemente sea relevante conocer los detalles sobre como un procesador está estructurado y compuesto. Sin embargo, si lo que nos interesa es componer los grandes componentes de hardware, el procesador será un elemento más, y su interior no es relevante. Más aún, los componentes de hardware no son relevantes si lo que interesa es analizar el software de la computadora. Un último ejemplo podría ser la forma en la que los datos están codificados, como binario, que no interesa si lo que queremos es utilizar el equipo, por ejemplo, editando una imagen o escribiendo un texto.

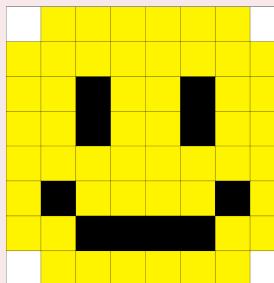
El concepto de caja negra utilizado en sistemas no debe confundirse con el “cajanegismo”; éste es un concepto vinculado a la sociología que hace referencia al hecho de que las personas suelen olvidar el funcionamiento interno de las cosas (generalmente nuevos dispositivos tecnológicos) a medida que se familiarizan con ellos y terminan por asimilarlos como de uso cotidiano.

La principal diferencia entre ambos conceptos radica en que el **estudio de un sistema como una caja negra es un proceso de abstracción**, mientras que el “cajanegismo” es más bien un proceso de olvido.

Por lo tanto, **concebir al hardware o al software de la computadora como una caja negra** puede ser útil desde el punto de vista de un programador o un usuario. **La concepción de caja negra es una idea que aplica de forma constante en las disciplinas informáticas**, y es un concepto recurrente ya que se deben realizar abstracciones de forma constante.

3.3 Actividades

Ejercicio 3.1 Dada la imagen a continuación, exprésala como un único número utilizando la misma codificación que se aplicó en este capítulo.



Ejercicio 3.2 La tabla mostrada en este capítulo para representar letras como números corresponde a la codificación ASCII. Utilizando esa tabla se pide que codifique la siguiente frase como una serie de números.

SOMOS LO QUE PROGRAMAMOS



Ejercicio 3.3 Nuevamente, usando ASCII, se pide ahora que decodifique el siguiente mensaje expresado como una serie de números.

80 82 79 71 82 65 77 79 32 76 85 69 71 79 32 69 88 73 83 84 79



Ejercicio 3.4 Si contamos con una computadora con 16 cables para representar nuestros datos, ¿Qué cantidad de números distintos pueden representarse con ellos?



Ejercicio 3.5 Un chiste de informáticos reza lo siguiente:

“Solo hay 10 tipos de personas en este mundo, los que entienden binario y los que no”

En qué radica la gracia del chiste.





4. Informática

Placa base de una computadora con sus circuitos impresos.

Fotografía de [Blickpixel](#).

En este capítulo veremos como los conceptos del capítulo anterior se aplican a los conceptos que utilizamos día a día como usuarios de computadoras. Veremos qué son los archivos, las carpetas y los programas, y veremos como se crean los programas utilizando lenguajes de programación.

Esto nos llevará a analizar qué es exactamente un lenguaje y a ver diversos tipos de lenguajes que existen en informática. Nos centraremos luego en los lenguajes de marcado, como una forma de representar datos mediante texto.

4.1 Archivos y extensiones

Cuando utilizamos la computadora sabemos bien donde está guardada nuestra información. Archivos, ubicados en carpetas como “Mis Documentos” poseen nuestros documentos de texto y planillas de cálculo. Otros, ubicados en “Mis Imágenes” mantienen nuestros recuerdos en forma de imágenes digitales.

Pero ¿Qué es exactamente un archivo informático? ¿Y qué es un directorio? En esta sección analizaremos esas preguntas desde dos puntos de vista. En una primer instancia, lo veremos desde el punto de vista de la computadora en bajo nivel. Luego, veremos cuales son los detalles importantes desde el punto de vista del usuario final para poder administrar y manejar mejor los archivos.

También analizaremos diversos tipos de archivos, y veremos como el sistema es capaz de identificar esos tipos de archivos, y como opera con y sobre ellos.

4.1.1 Archivos informáticos

Un archivo informático no es más que la representación digital de un archivo físico que se tiene en papel. Puede contener texto, imágenes, videos, modelos tridimensionales, o cualquier otro contenido que uno quiera almacenar y manipular en la computadora.

En última instancia, un archivo informático se reduce solo a una serie de bits que la compu-

tadora es capaz de almacenar, transmitir, y manipular mediante los circuitos que posee, como ya mencionamos previamente. Pero surge la pregunta: “¿Qué es lo que hace que una serie de ceros y unos sea una imagen, y otra sea un video, audio o un modelo 3D?”. La respuesta radica en el **formato de archivo**.

Cada archivo tiene un formato, que es lo que determina como la computadora debe leer e interpretar los bits que componen al mismo. El formato está relacionado a la codificación que se haya elegido. Por ejemplo, la codificación que elegimos para las imágenes en la sección anterior, es una, de muchas posibles codificaciones.

No hay un único formato para cada tipo de archivo, por ejemplo, no hay un único formato para todo lo que son imágenes. Cada formato tiene sus ventajas y desventajas, y está pensado para solucionar problemas puntuales que tienen que ver con representar ese archivo en la computadora en alguna forma particular, por ejemplo, con mayor calidad, con menor espacio posible, etc.

La información se **codifica** como binario y se **decodifica** como información (de binario a información). Esta interpretación de los datos en ida y vuelta está dada entonces por el **formato de archivo**. Para que los archivos puedan compartirse y leerse como se espera por cualquier computadora, la forma en la que interpretamos esos ceros y unos debe ser la misma para todos, por lo que los formatos están **estandarizados**.

Definición 4.1 Un **archivo informático** es la representación digital de un archivo físico, el cual se encuentra almacenado en un dispositivo de almacenamiento como una cadena de bits la cual es la codificación de lo representado utilizando algún código determinado, determinando su formato. Un archivo además tiene un nombre y un tamaño (que depende de la cantidad de bits que tiene el archivo).⁶

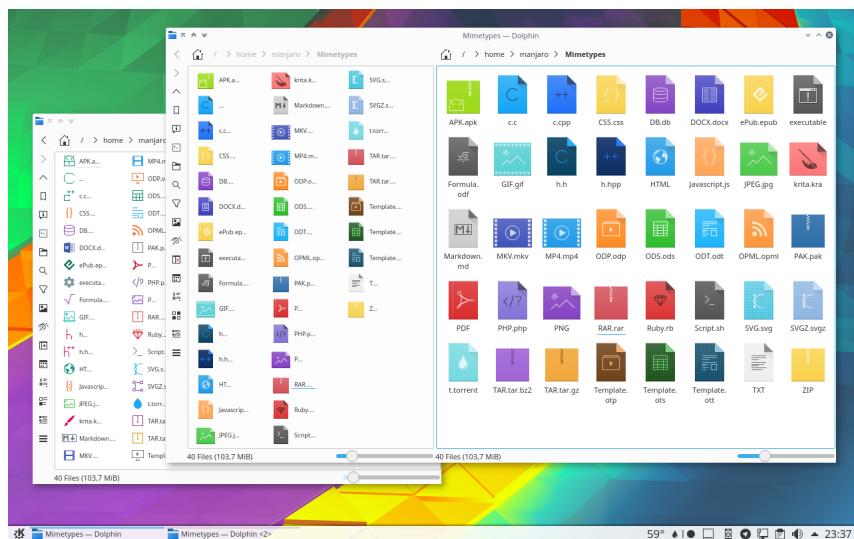
Para tener una mejor idea, analicemos un ejemplo simple. Una fotografía o imagen digital puede almacenarse en alguno de los muchos formatos disponibles para ello. Tres de los más utilizados son los formatos RAW, JPEG y PNG.

- Las imágenes en formato RAW son aquellas en donde se almacena toda la información para cada pixel de la imagen. Es decir, se divide la imagen en una grilla gigante de pequeños puntos (pixeles), y el archivo consiste en la información de que color debe tener cada pixel. Esto otorga la mejor precisión de los colores y la mejor calidad de imagen posible, fiel al dispositivo que capturó la imagen. Como contrapartida, una sola imagen en buena calidad puede llegar a ocupar mucho tamaño en dispositivos de almacenamiento.
- Las imágenes JPG utilizan una técnica mediante la cual pueden eliminar la información de varios pixeles, los cuales pueden deducirse a partir de la información de los pixeles circundantes. Adicionalmente, se almacena la información de cada pixel mediante una técnica que permite reducir el tamaño de la información. Así, la calidad de imagen baja significativamente, pero el tamaño se reduce de forma acorde. Este formato es ideal para fotografías digitales y el estándar usado por la mayoría de las cámaras fotográficas.
- Las imágenes PNG utilizan técnicas que permiten minimizar el tamaño que ocupa la información de cada pixel, al pre-anotar los colores que aparecen en la imagen. Adicionalmente, marca grandes bloques de imagen anotando su posición relativa y tamaño, marcándolo con un único color. Este formato es ideal para imágenes que tienen pocos colores, o grandes bloques de un mismo color, por ejemplo, dibujos digitales.

Incluso los programas de la computadora son solamente archivos, cuyo formato le indica a la compu que debe ejecutarlo, y como hacerlo, algo que veremos más adelante.

Es importante saber que todo archivo tiene un **nombre**, un **tamaño** (dado en bytes) y una ubicación (un directorio en donde se encuentra). Además, dependiendo del sistema operativo y otros detalles puede contener otra información. Estos datos no componen a lo que representa el archivo en si mismo, sino que es información adicional que el mismo posee, y que se conocen como **metadatos**.

4.1.2 Extensiones de archivos



Muchos sistemas operativos utilizan iconos distintos para cada formato de archivo al mismo tiempo que ocultan la extensión del nombre de archivo. Esto permite al usuario identificarlo visualmente, a pesar de no leer el nombre completo. En la foto, una imagen del escritorio de un sistema con KDE plasma.

Imagen de KDE.org

Ahora bien, surge otra pregunta; ¿Cómo hace la computadora para saber que formato tiene cada archivo?. Si bien en realidad existen varias formas de hacerlo, lo más común es que se determine el formato mediante el nombre del archivo.

El nombre que vemos de los archivos, no es en general el nombre completo del mismo. El nombre completo se compone del nombre, y de una **extensión**, que no es más que un conjunto de letras que se anexan al final del nombre, separados del mismo por un punto.⁷

Así, por ejemplo, si tenemos una imagen “de vacaciones”, lo más probable es que el nombre completo del archivo sea “de vacaciones.jpg”. La extensión del archivo es entonces “jpg” (o a veces, según la bibliografía también “.jpg”)

En realidad

Como ya se dijo, “la computadora”, no hace nada por si sola, es el software el que hace. En particular, el que detecta el formato de archivo y elige la forma de interpretarlo es el sistema operativo. Esta es, entre otras varias, una de las tareas que tiene el mismo.

Además, algunos sistemas operativos utilizan otras técnicas adicionales a mirar la extensión de archivo para determinar el formato, como por ejemplo, la lectura de la primer parte del archivo en búsqueda de “números mágicos” (un número que, en general, es distinto para distintos formatos de archivo).

Existen miles de extensiones de archivo, algunos relativamente comunes (por ejemplo JPG para fotografías, o MP3 para música), y otros que son bastante más exóticos, pues son usados por formatos de archivo que sirven solo para una aplicación en particular (por ej. DWG, formato de archivo de AutoCAD).

A continuación se presenta un listado con algunos de los formatos de archivo más comunes que uno puede encontrar en el uso cotidiano:

| Fotos e imágenes: | Audio: | Video: |
|-------------------|-----------------------|--------------|
| ■ JPG | ■ MP3 | ■ MP4 |
| ■ JPEG | ■ OGG | ■ AVI |
| ■ PNG | ■ WAV | ■ DIVX |
| ■ APNG | ■ 3GP | ■ XVID |
| ■ BMP | ■ M4A | ■ MOV |
| ■ TIFF | ■ FLAC | ■ WMV |
| ■ GIF | ■ AIFF | ■ FLV |
| ■ SVG | | ■ MKV |
| Documentos: | Archivos comprimidos: | Texto plano: |
| ■ DOC | ■ ZIP | ■ TXT |
| ■ DOCX | ■ ZIPX | ■ MD |
| ■ ODT | ■ RAR | ■ XML |
| ■ XLS | ■ TAR | ■ HTML |
| ■ XLSX | ■ GZ | ■ JSON |
| ■ ODS | ■ ISO | ■ JS |
| ■ PPT | | ■ C |
| ■ PPTX | | ■ CSS |
| ■ ODP | | ■ JAVA |
| ■ PDF | | |

Una categorización muy amplia y genérica, pero sumamente útil es la que agrupa los archivos en los siguientes tres grupos:

Archivos ejecutables Son los programas de la computadora. Programas como Word, Excel, Paint, aplicaciones de celulares, editores de fotos, etc.

Archivos de datos binarios Son los archivos que solamente pueden ser leídos por programas específicamente diseñados para tal fin: documentos de Word, imágenes, videos, audio, etc. Su codificación puede ser estándar, o puede estar restringida a ser solo conocida por la aplicación que maneja el archivo.

Archivos de texto plano Son archivos que usan una codificación estándar y en donde su contenido representa pura y exclusivamente texto. Pueden ser leídos por un **editor de texto**. La gracia de los archivos de texto es que la interpretación de sus bits representa siempre lo mismo, texto.

Un detalle más

Más allá de que los archivos representen lo mismo (por ej. una imagen) cambiar la extensión de archivo no hace que cambie el formato en el que está codificado, por lo que transformar una foto en .jpg en una en .png requiere del uso de un programa que sea capaz de cambiar todos los unos y ceros del archivo en otros que representen lo mismo, pero con otra codificación. Los archivos de texto pueden ser cambiados de extensión tantas veces como se quiera, y siguen pudiendo abrirse con el mismo programa, pues siguen representando siempre texto, y la codificación es la misma para todos los archivos de texto.

Dentro de cada uno de estas categorías se encuentran a su vez sub-categorías que terminan por definir de forma más granular la forma en la que se interpreta cada archivo. Por ejemplo, las imágenes pueden estar codificadas de diversas formas (diferentes a las que vimos), cada una con ventajas y desventajas, y por tanto la interpretación de los ceros y unos del archivo, cambia según el formato.

4.2 Organización de archivos, directorios y rutas

Todos los archivos se deben almacenar en algún lado para poder utilizarlos cuando se deseé. Esto puede ser, en un disco rígido, en un disco extraíble (CD, DVD, Pendrive, Tarjeta de memoria, etc). La forma en la que se estructuran los archivos en el disco se conoce como **sistema de archivos**.

Uno de los primeros sistemas de almacenamiento para guardar archivos era el disco flexible (En inglés Floppy), un disco magnético extraíble, recubierto de una carcasa plástica de protección. Los primeros sistemas operativos que soportaban estos discos como espacio de almacenamiento, solo permitían guardar en ellos unos pocos archivos. Las personas agrupaban archivos relacionados en un mismo disco, y en caso de que la cantidad de archivos fuera mayor a la del disco, agrupaban discos físicamente.

Sabías qué

Los primeros sistemas de archivos utilizaban una cantidad preestablecida de bits para representar el nombre y extensión de cada archivo. Por eso, en sistemas operativos antiguos, el nombre de un archivo no podía superar los 8 caracteres, y la extensión no podía superar los 3. Esta característica quedó, y por eso la mayoría de las extensiones hoy cuentan con solo 3 caracteres, más allá de que esa limitación ya no existe.

A medida que aumentaba la capacidad de los discos, y la cantidad de archivos que podían almacenar, el organizar archivos separándolos en distintos discos se vuelve ineficiente. Así surgen los primeros sistemas de archivos con **directorios** (también llamadas carpetas).

Los directorios son el equivalente lógico de una carpeta física, permitiendo agrupar archivos (en base a alguna cualidad que el usuario considere oportuna).

Además, una carpeta puede estar dentro de otra carpeta, generando una estructura de árbol. En estas estructuras siempre hay una carpeta que contiene a todas las otras, conocida como carpeta raíz.



Windows utiliza la analogía de archivadores físicos, carpetas y archivos para su sistema operativo.

Imagen propia.

Definición 4.2 Un **directorio** es la representación digital de una carpeta física. La misma se encuentra en alguna ubicación en algún dispositivo de almacenamiento, posee un nombre, y puede contener ninguno o varios archivos y/o directorios en su “interior”. El tamaño, en bytes, de un directorio es la suma del tamaño de todos los directorios y archivos que contiene.⁶

Cada sistema operativo tiene su propio sistema de archivos, y su forma particular de organizar las carpetas. Por ejemplo, Windows tiene una carpeta raíz por cada disco rígido que tenga conectada

la computadora. Linux y macOS utilizan una única carpeta raíz que contiene como sub-carpetas los diversos discos conectados.

Los sistemas de archivos basados en directorios permitían cosas muy interesantes, como por ejemplo, tener varios archivos con el mismo nombre (nombre completo, es decir, nombre y extensión) siempre y cuando se encuentren en distintas carpetas. Pero también traían aparejados otros problemas, como que ahora los archivos no son identificables mediante solo su nombre, sino mediante el nombre y la carpeta en la que se ubican.

Los sistemas de archivos que permiten directorios que contienen a otros directorios y archivos se denominan sistemas de archivos jerárquicos.

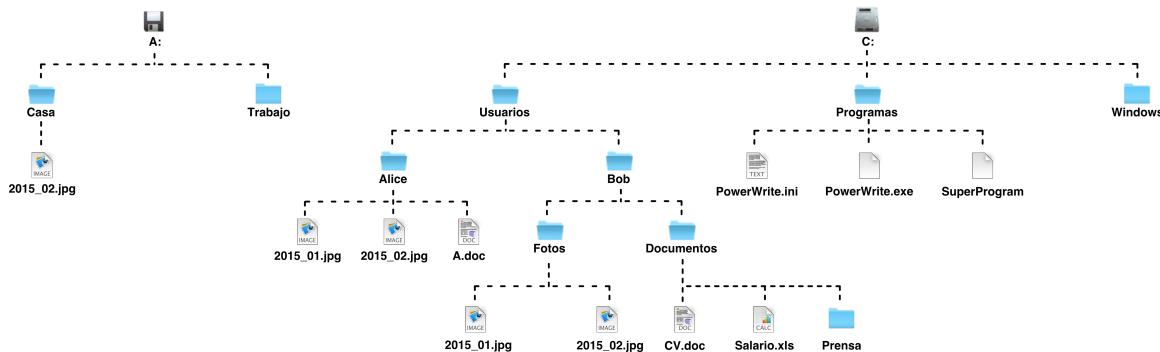
4.2.1 Rutas absolutas

En los sistemas operativos cuyo sistema jerárquico (lo cuál es hoy en día prácticamente la totalidad de los mismos, aunque hay excepciones) la ruta es la forma de identificar de forma única un archivo en el equipo. Así, todo archivo tiene asociada una ruta absoluta en el equipo.

Definición 4.3 Una **ruta absoluta a un archivo** es el camino de carpetas que hay que seguir en el árbol de directorios para llegar **desde el directorio raíz hasta el archivo en cuestión**.⁸

La ruta suele variar entre un sistema operativo y otro, pues como ya mencionamos, cada sistema tiene su propia forma de estructurar las carpetas, pero el concepto subyacente es el mismo.

A continuación se muestra un detallado ejemplo con algunos directorios en un sistema Windows para analizar rutas absolutas de varios archivos.



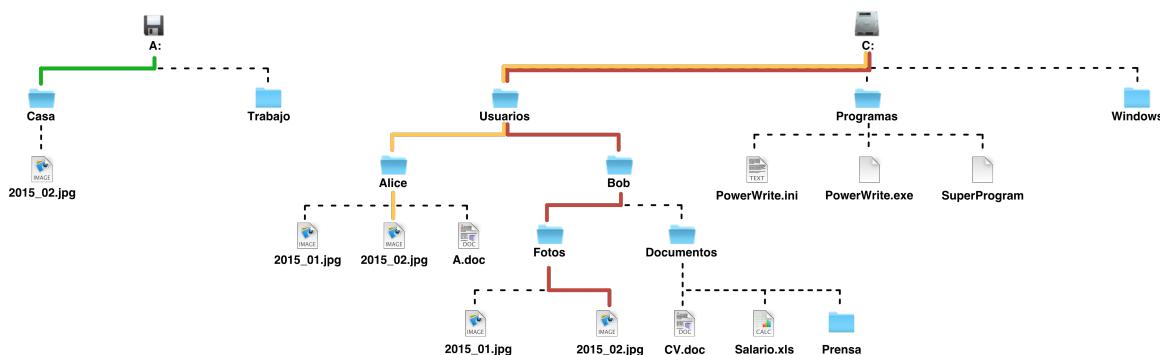
En este sistema hay dos dispositivos de almacenamiento. Un disco extraible “floppy” que Windows ha identificado con el nombre “A:”, y un disco rígido, el cual se identifica con “C:”. Es decir, “A:”, y “C:” son las carpetas raíces de cada uno de los dispositivos de almacenamiento presentes en el sistema.

Dentro del disco rígido hay tres carpetas, “Usuarios”, “Programas” y “Windows”. La carpeta “Windows” está **vacía**. Un directorio se denomina vacío cuando no contiene otras carpetas o archivos en su interior. La carpeta “Programas” contiene solamente tres archivos en su interior, “PowerWrite.ini”, “PowerWrite.exe” y “SuperProgram”. La carpeta “Usuarios” tiene dentro dos carpetas, “Alice” y “Bob”, las cuales a su vez contienen otras carpetas y archivos en su interior.

Surgen varias cosas de este ejemplo. En primer lugar podemos ver que hay más de un archivo con el nombre “2015_02.jpg”. ¿Significa esto que todos son el mismo archivo? La respuesta es no, solo son archivos con el mismo nombre, pero el contenido del mismo podría ser completamente distinto. Podemos así modificar uno de esos archivos sin afectar el contenido de los otros. También

surge la pregunta ¿Cómo sabe el sistema de cuál de los archivos estamos hablando? La respuesta es, mediante la ruta absoluta del archivo.

Cuando se trabaja sobre un archivo en el equipo, y aunque transparente para el usuario en la mayoría de los casos, se debe indicar la ruta completa al archivo, para que el sistema lo identifique de forma única. Tomemos el ejemplo del archivo “2015_02.jpg”; dicho nombre no es único, sino que existen tres archivos con el mismo en ubicaciones distintas en el equipo. Si deseamos trabajar con cualquiera de ellos, deberíamos dar al sistema la ruta absoluta de aquel archivo que nos interesa. Para obtenerla uno debe imaginar una especie de camino que parte desde la carpeta raíz (por ejemplo, “C:”) y luego continúa por las líneas punteadas hasta arribar en el archivo, pasando en el camino por una serie de directorios intermedios (siempre moviéndose hacia abajo y a los laterales, obteniendo así el camino más corto). En el gráfico siguiente se muestran tres caminos distintos que llevan a los distintos archivos con el nombre “2015_02.jpg”, marcados cada uno con un color distinto.



Cada línea corresponde entonces a la ruta absoluta para cada uno de los archivos con el nombre “2015_02.jpg”. La primera línea, la verde, parte desde el disco floppy “A:”, pasa por el directorio “Casa”, para luego arribar en el archivo. Así, vamos a decir que la ruta a ese archivo es:

■ Ejemplo 4.1

Ruta roja

A: » Casa » 2015_02.jpg

En el segundo caso, la ruta amarilla, parte desde el disco rígido “C:”, y pasa luego por la carpeta “Usuarios”, para continuar por “Alice” previo a arribar al archivo. Los pasos serían entonces:

■ Ejemplo 4.2

Ruta amarilla

C: » Usuarios » Alice » 2015_02.jpg

Finalmente, la ruta roja parte igualmente desde el disco rígido “C:”, y pasa luego por la carpeta “Usuarios”, pero continua ahora por “Bob”, luego en “Fotos” para arribar ahora si, al archivo:

■ Ejemplo 4.3

Ruta roja

C: » Usuarios » Bob » Fotos » 2015_02.jpg

Como vemos, tres archivos con el mismo nombre tienen rutas completamente distintas. Podemos obtener varias conclusiones de este ejemplo. Por un lado vemos que los sistemas de archivos con directorios permiten que múltiples archivos se llamen de la misma forma, ya sea que efectivamente sean el mismo archivo, o que no lo sean. Por otro lado, no pueden existir nunca dos archivos con el mismo nombre en un mismo directorio.

Miremos con atención el caso de “PowerWrite”. Existen dos archivos con ese nombre en la misma carpeta, sin embargo, tienen distinta extensión. Tenga presente el lector que siempre que hablamos del nombre de un archivo, en general nos referimos al nombre y su extensión.

De la experiencia que usted pueda tener con el uso de computadoras, sabrá que los archivos pueden copiarse, y moverse a diferentes directorios. ¿Qué ocurre si muevo un archivo de lugar? Lo que sucede es que su ruta absoluta cambia. Todo esto puede parecer algo menor, pero las rutas absolutas son utilizadas por los programadores para hacer referencia a archivos que deben estar presentes en el equipo para que un determinado sistema funcione. Mover un archivo de lugar puede hacer que todo un sistema falle, pues el archivo no se encuentra en la ruta que el sistema esperaba. Así también, modificar configuraciones de un sistema que involucran rutas absolutas sin mover de forma acorde los archivos puede ocasionar el mismo efecto. Por tanto, entender qué y cómo funcionan las rutas es una parte fundamental para cualquier persona que quiera comprender un poco más en profundidad los sistemas informáticos, cualquiera sea su fin.

Es menester mencionar algunos detalles sobre nomenclatura de rutas en diversos sistemas. Note que en los ejemplos utilizamos el símbolo “»” para separar cada uno de los elementos de una ruta. En realidad cada sistema operativo puede utilizar una símbolo distinto para separar los elementos de una ruta. Windows utiliza por convención la barra invertida (\). Linux, macOS, y otros sistemas (aquellos que utilizan la convención de los sistemas basados en UNIX, y que en general se denominan sistemas basado en UNIX, o sistemas POSIX) separan los elementos con barra simple (/). Así, las rutas de los ejemplos de esta sección se escribirían en Windows:

- A:\Casa\2015_02.jpg
- C:\Usuarios\Alice\2015_02.jpg
- C:\Usuarios\Bob\Fotos\2015_02.jpg

La convención en sistemas UNIX, es también la utilizada por muchos lenguajes de programación y de marcado, como veremos más adelante. Utilizando esta convención las rutas anteriores quedarían.

- A:/Casa/2015_02.jpg
- C:/Usuarios/Alice/2015_02.jpg
- C:/Usuarios/Bob/Fotos/2015_02.jpg

Como detalle final, cabe mencionar que los sistemas UNIX no utilizan una letra para cada uno de los dispositivos, sino que suelen tener un sistema de archivos que comienza en una carpeta virtual denominada “/” (“raíz” o “root”). Los diversos dispositivos son simplemente carpetas dentro de alguna de las carpetas que tiene “/”, por ejemplo, la carpeta “mnt” o “mount”. Las rutas absolutas en dichos sistemas comienzan entonces con la barra simple, para indicar que parten desde la raíz. Las rutas anteriores podrían verse algo como:

- /mount/floppy1/Casa/2015_02.jpg
- /mount/hd0/Usuarios/Alice/2015_02.jpg
- /mount/hd0/Usuarios/Bob/Fotos/2015_02.jpg

Las rutas absolutas son un concepto simple de entender, pero complejo de dominar sin práctica en su uso. Más difícil aún es el concepto de rutas relativas.

4.2.2 Rutas relativas

Históricamente, los primeros sistemas funcionaban indicándole a la computadora comandos a realizar, mediante la introducción de las mismas utilizando el teclado. El usuario solamente visualizaba una pantalla con texto, y el mouse no existía. En ese contexto, indicar la ruta absoluta de

forma constante para cualquier archivo que el usuario quisiera manipular era sumamente engorroso. Para solucionar esa molestia surgen dos ideas, la primera es el concepto de **directorio de trabajo actual**, y la segunda las **rutas relativas**.

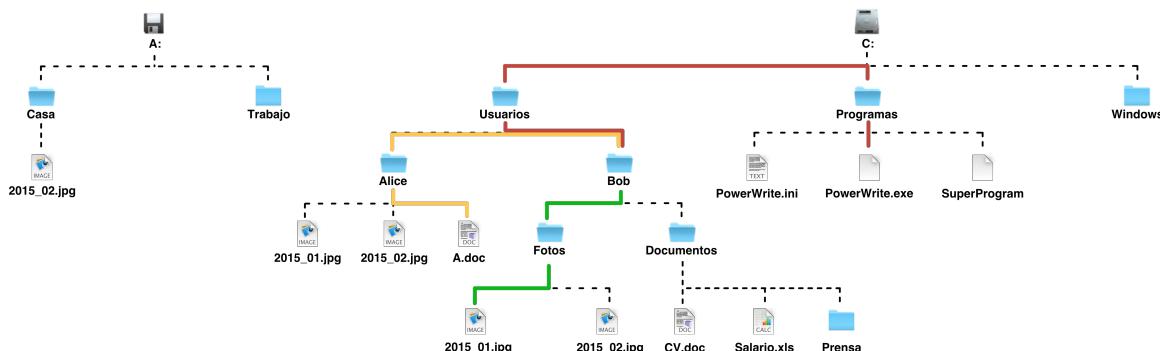
Definición 4.4 El **directorio de trabajo actual** es un directorio de un sistema de archivos jerárquico el cual se encuentra asociado de forma dinámica a un proceso o tarea del sistema.

En estos sistemas antiguos, el usuario se encontraba siempre trabajando en un directorio, el cual era denominado el directorio de trabajo actual. El sistema operativo mantenía un registro del directorio de trabajo actual, ya que el mismo podía cambiar a lo largo del tiempo, pudiendo ser modificado por el usuario mediante la utilización de uno o más comandos. Este concepto viene de la mano con la idea de **rutas relativas**.

Definición 4.5 Una **ruta relativa** hacia un archivo consiste en la ruta a realizar desde un directorio cualquiera para dar con dicho archivo. Si el directorio elegido es el directorio raíz del dispositivo en donde se encuentra el archivo, la ruta relativa y la ruta absoluta son la misma.⁹

Una ruta relativa puede indicar donde se encuentra un archivo sin tener que conocer todas las carpetas del equipo. En particular, en estos primeros sistemas operativos, uno podía mencionar un archivo con la ruta relativa desde el directorio de trabajo actual hacia el mismo. Así, sin importar en qué carpeta uno estuviera trabajando, si se deseaba hablar de un archivo en esa carpeta bastaba solo con el nombre, incluso si la ruta absoluta era larga y compleja.

Veamos un ejemplo en donde el usuario ha decidido que el directorio de trabajo actual sea la carpeta “Bob”, dentro de la carpeta “Usuarios” en el disco rígido “C:”. Es decir, todas las rutas serán relativas a la carpeta “C: » Usuarios » Bob”.



Analicemos primero la ruta verde. Bob quiere acceder al archivo en “C: » Usuarios » Bob » Fotos » 2015_01.jpg”. Como el directorio de trabajo ya es “C: » Usuarios » Bob”, basta con poner la ruta desde dicha carpeta (sin incluirla) hasta el archivo.

■ Ejemplo 4.4

Ruta verde

Fotos » 2015_01.jpg ■

Es decir, desde la carpeta “Bob” basta ir a la carpeta “Fotos” para encontrar allí el archivo “2015_01.jpg”. Note como la ruta siempre va hacia “abajo”, es decir, que nos adentramos más en la jerarquía de carpetas, siempre yendo hacia la carpeta que está dentro de la anterior, y nunca saliendo de la misma.

En el caso de la ruta amarilla esto no es así. Bob quiere acceder al archivo de Alice, pero dichos archivos no se encuentran dentro de una carpeta en “Bob”, es decir, se debe “subir” para poder acceder a ellos. Es decir, la ruta relativa debe dejar constancia de que, en primer lugar, hay que ir de la carpeta “C: » Usuarios » Bob” a “C: » Usuarios” en lo que podríamos denominar “subir un nivel”. La ruta quedaría algo como:

■ **Ejemplo 4.5**

Ruta amarilla

^ » Alice » A.doc



Es decir, partiendo de la carpeta “Bob”, vaya a la carpeta inmediatamente arriba (llamada **padre**). Luego, desde ahí, vaya a la carpeta “Alice”, una vez allí encontrará el archivo “A.doc”.

En el caso de la ruta roja el proceso es el mismo, aunque en este caso no basta con subir un nivel, sino que tenemos que subir varias veces.

■ **Ejemplo 4.6**

Ruta roja

^ » ^ » Programas » PowerWrite.exe



En este caso la ruta absoluta hubiera sido más corta y sencilla que la ruta relativa, sin embargo es interesante ver como una ruta relativa puede aplicar en cualquier punto de la jerarquía de carpetas, e independientemente de la carpeta.

Nuevamente el símbolo “^” que utilizamos para indicar “ir un nivel hacia arriba” se representa distinto en distintos sistemas. Sin embargo, los sistemas más populares usan todos la misma convención: “..”. Las rutas anteriores quedarían en Windows:

- Fotos\2015_01.jpg
- ..\Alice\A.doc
- ../../Programas\PowerWriter.exe

Y en los sistemas UNIX

- Fotos/2015_01.jpg
- ./Alice/A.doc
- ../../Programas/PowerWriter.exe

Una propiedad interesante de las rutas relativas es que, dado un directorio como directorio de trabajo actual, todas las rutas relativas desde el mismo hacia archivos o carpetas que este contenga son iguales independientemente de donde esté ubicado el directorio en el equipo. Esto resulta muy útil a las personas de sistemas, pues un archivo de configuración o programa que utilice esta técnica para hacer mención de sus archivos puede ser copiado o movido a otro lado sin perjudicar su funcionamiento, dado que no se copie solo el archivo, sino todos los archivos que están en la carpeta del programa. Veremos más este detalle en secciones futuras.

4.2.3 Universal Resource Identifier

Imaginemos ahora una serie de computadoras conectadas entre si. Es decir, una red de computadoras. Las redes permiten compartir recursos, los cuales, entre otras cosas, incluyen archivos y directorios. Por tanto, es importante poder identificar de forma única un archivo, ya no en un equipo, sino en toda una serie de equipos. Si la cantidad de máquinas es grande, nombrar con una única letra cada dispositivo de la red se vuelve imposible. Para solucionar este problema surgen el **identificador universal de recursos**, o en inglés **universal resource identifier (URI)**. Pero antes de entender que son los URIs, veamos un momento como se identifica un equipo en una red.

Cuando se cuenta con diversas computadoras conectadas a una red, se debe poder hacer mención a una máquina en particular, diferenciándola del resto de las de la red. Así, los sistemas de software de red (por ejemplo, aquellos incluidos en un sistema operativo o incluidos en dispositivos como routers) asignan a cada equipo un número único, conocido como **dirección IP**.

Otra técnica para hablar de un equipo único en una red consiste en asignarle un **nombre de dominio**. Los nombres de dominio son identificadores en texto, en lugar de numéricos, por lo que son más fáciles de recordar por los seres humanos. Un sistema especializado mantiene registro de los nombres asignados para que no existan dos equipos con el mismo nombre, además de recordar la asociación entre el nombre de dominio y la dirección IP a la que corresponde.

Tal vez haya visto en algún momento estas formas de identificar computadoras al navegar por Internet, pues Internet no es más que una gran red donde el usuario simplemente solicita archivo a diversos equipos. Una dirección IP se compone en realidad de 4 números entre 0 y 255 cada uno, los cuales se escriben separados por puntos (ej. “125.65.92.135”). En cambio los nombres de dominio suelen consistir en dos partes, de las cuales la primera suele ser un nombre que identifica a una empresa u organización, y la segunda, que consiste en un conjunto de 2 y/o 3 letras que identifican el tipo de organización (empresa, ONG, entidad educativa, etc.) y/o el país al que pertenece (ej. “google.com.ar”, “wikipedia.org”).

Ahora bien, una URI se compone entonces de diversas partes. En primer lugar un protocolo, que le indica a los equipos de que forma se espera que se transfiera el archivo de una máquina a la otra, seguido por alguna forma de identificador de equipo (ya sea la dirección IP o un nombre de dominio) y luego una ruta hacia el archivo en cuestión. Como por motivos de seguridad las computadoras no comparten en la red todos los archivos sino una carpeta específica del equipo, la ruta de la URI no es una ruta absoluta, sino relativa a la carpeta compartida.¹⁰



Un URI también puede ser llamado URL (Universal Resource Locator). Si bien hay pequeñas diferencias técnicas entre uno y otro concepto, los términos hoy en día se suelen utilizar de forma prácticamente intercambiable. Además, tanto los URI como los URL pueden contener otros elementos, como sub-dominios, puerto, queries, y fragmentos, que no analizaremos aquí.

Definición 4.6 Un **identificador de recursos universal** es una cadena de texto capaz de identificar un recurso en un equipo específico dentro de una red de computadoras.

El concepto de URI no es nada más que el mismo concepto de ruta expandido hacia las redes, de forma tal que cualquier archivo (que sea accesible) tenga una forma única de identificarse.

Cabe destacar como detalle adicional que el símbolo utilizado en una URI para separar los elementos de la ruta sigue el estándar de los sistemas UNIX, es decir, con barra simple (/). Esto quiere decir que, cuando se accede a un archivo en un sistema Windows desde Internet, las barras invertidas de la ruta al archivo en Windows deberán ser reemplazadas con barras simples.

4.2.4 Programas, Editores y Visualizadores

Los programas son uno de los elementos de software que más utilizamos. Son lo que provee a nuestra computadora de diferentes funcionalidades especificar, y nos permite realizar tareas

sumamente diversas con un mismo equipo.

A diferencia del resto de los archivos del equipo, un programa es un archivo ejecutable. Es decir, al pedirle al sistema operativo que abra el archivo de un programa, lo que este hará será ejecutarlo.

Ejecutar un programa consiste simplemente en llevar adelante una serie de pasos que el programa define. Estos pasos pueden ser muchos, y muy diversos, dando lugar a programas completamente distintos. Además, pueden incluir elementos visuales, como botones, imágenes y otros, para simplificar el uso para el usuario final.

Los programas pueden agruparse en categorías según diversos criterios. Por ejemplo, podríamos decir que hay programas que son “accesorios”, elementos prácticos que el usuario puede requerir y conviene tener a manos, como una calculadora o un anotador. También hay programas que podrían entrar en una categoría “reproductores de video”, que permiten al usuario ver videos en uno o más formatos, así como controlar factores como el sonido y otros.

Una categorización interesante que nos interesa mencionar aquí tiene que ver con como tratan los programas a los archivos que manipulan. En particular, podemos determinar dos grandes categorías, **visualizadores y editores**.

Un visualizador permite simplemente ver los datos que se encuentran almacenados en un archivo, pero no modificarlos. Por ejemplo, un reproductor de video permite ver una película digital, pero no modificarla. Un navegador de internet permite ver páginas web, pero nuevamente, no pueden modificarse usando ese programa.

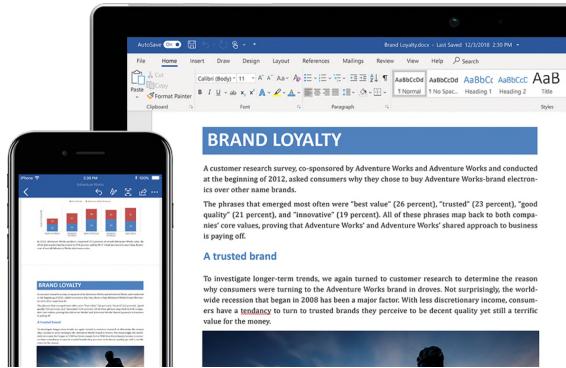
Definición 4.7 Un **visualizador** es un programa que permite ver el contenido de archivos que tienen algún formato específico.

Los editores en cambio permiten manipular la información del archivo, alterando sus contenidos. Por ejemplo, un editor de sonido, o un manipulador de imágenes permiten editar audio o fotografías digitales, y no son los programas que el usuario utilizaría si lo que le interesa es escuchar música o ver fotografías.

Definición 4.8 Un **editor** es un programa que permite crear y modificar el contenido de archivos que tienen algún formato específico.

Existen programas que cumplen ambas funciones al mismo tiempo. Por ejemplo Microsoft Word es el programa que se usa tanto para editar archivos de documentos, como para visualizar su contenido. Los documentos de Word no pueden (en principio) verse con otros programas que no sean Word, y tampoco editarse con otros programas.

Una mención especial merecen los **editores de texto**. Un editor de texto permite editar el contenido de un archivo de texto sin formato. Como veremos más adelante, algunos formatos de archivo no codifican la información en binario, sino como texto. Para estos archivos, un editor de



Banner publicitario de Microsoft Word con su aplicación para PC y teléfonos móviles.

Imagen de Microsoft.

texto permite modificar su contenido, independientemente si codifican imágenes, documentos, u otros.

4.3 Programas y Lenguajes

Un programa es el resultado del trabajo de un desarrollador de software (dentro de otros elementos), por lo que para realizar un programa se requiere de una serie de conocimientos técnicos específicos que permiten desarrollarlo.

Resulta interesante tener una noción básica de la forma en la que se realiza desarrolla un programa (sin internalizar en los conceptos específicos). Para encarar esta tarea es prioritario comprender el concepto de lenguaje, desde un punto de vista técnico.

4.3.1 Lenguajes

El lenguaje nos posibilita comunicarnos con quienes nos rodean, pero también con otras personas a lo largo del tiempo y el espacio. Gracias al lenguaje es que hoy tenemos registro de lo que pasó en tiempos distantes, en lugares remotos.

Definición 4.9 Un **lenguaje** es un sistema de comunicación, el cual se encuentra definido y estructurado, y que posee un contexto de uso (es decir, se puede establecer una semántica a determinados elementos).



Banderas de diversos países ondeando en la plaza de los Juegos Panamericanos en Rio de Janeiro.

Imagen de Wilson Dias / Agência Brasil

Puede sonar complejo en la definición, pero utilizamos el lenguaje todo el tiempo. La forma más común de lenguajes son los idiomas, como el español o el inglés. Cada uno de los idiomas define un conjunto de símbolos que son válidos, por ejemplo, el español cuenta con la letra “Ñ”, que no está presente en el inglés. Cualquier palabra que use dicho símbolo es inválida en inglés, y por tanto no forma parte de ese lenguaje, mientras que en cambio, si puede formar parte del nuestro. También las reglas de un idioma (que son extremadamente complejas) determinan que textos son válidos y cuales no. En primer lugar, la regla más sencilla determina palabras que existen en el lenguaje (dadas muchas veces por los diccionarios oficiales). La gramática determina luego como unir esas palabras para que cobren sentido.

No solo hay idiomas que han surgido y evolucionado naturalmente, sino que pueden existir lenguajes creados de forma artificial con fines específicos. Por ejemplo, algunas series de televisión han creado lenguajes con fines artísticos, como la serie de ciencia ficción Star Trek y el lenguaje Klingon, o Game of Thrones (basada en los libros de George R. R. Martin) y el Idioma Dothraki.¹¹

Pero también han habido intentos más nobles, como el Esperanto. Creado por L. L. Zamenhof, un oftalmólogo polaco, el Esperanto surge con la idea de que se transforme en un idioma capaz de ser fácilmente aprendido y hablado por cualquier persona en el mundo. La intención última era unir a la humanidad bajo un mismo lenguaje universal (como segunda lengua, sin reemplazar los idiomas oficiales que ya tenían las distintas naciones). El Esperanto es la lengua planificada más expandida y utilizada en el mundo hoy en día, y es reconocida por instituciones como la UNESCO.

Todo lenguaje está compuesto de dos partes, una **sintaxis** y una **semántica**. La sintaxis determina los símbolos y las reglas que es posible utilizar en el lenguaje, mientras que la semántica determina que significado le asociamos a una determinada sintaxis. Por ejemplo, las expresiones “Me baño en el río” y “Me río en el baño” son muy similares en cuanto a sintaxis se refiere, pero la semántica asociada, es decir, el significado de esas dos frases, es completamente distinto.

Pensemos también en una frase como “lo voy a tomar”. ¿Cuál es la semántica de dicha frase?. A priori es difícil saberlo pues la palabra “tomar” tiene muchos significados distintos dependiendo del contexto. Puede que se esté hablando de un cuarto de un hotel, o de un transporte público. También podría ser que la frase se refiera a una bebida, o a un helado. Incluso la palabra tomar puede referirse a agarrar un objeto. Decimos entonces que la frase es **ambigua**, pues su significado no queda claro si no se cuenta con el contexto subyacente.

La ambigüedad es una problemática que a los humanos no nos molesta demasiado, pues somos capaces de analizar contextos y desambiguar rápidamente una frase basándonos en la información que tenemos. Las máquinas por otro lado, no tienen esa capacidad. Por tanto, es difícil (imposible prácticamente) comunicarse con una computadora utilizando los idiomas que hablamos a diario. Por eso surgen lenguajes especialmente diseñados para tal fin, y que tienen como principal característica el no presentar ambigüedades. Estos lenguajes se conocen como **lenguajes formales**.

Definición 4.10 Un **lenguaje formal** es un lenguaje cuyos símbolos y reglas para unir esos símbolos están formalmente especificados. Al conjunto de los símbolos primitivos se lo llama **alfabeto**, y al conjunto de las reglas se lo llama **gramática formal**.¹²

Dentro de los lenguajes formales podemos identificar dos grandes categorías: los **lenguajes de programación**, cuya finalidad es poder escribir programas de computadora, y los **lenguajes de marcado**, cuyo fin es describir datos que un programa puede luego interpretar.

4.3.2 Lenguajes de programación

Los **lenguajes de programación** consisten en un conjunto de comandos (llamadas a veces instrucciones) que la computadora es capaz de entender. Mediante una secuencia de estas instrucciones se pueden lograr complejos programas que realizan diversas tareas. Así, puede procesar datos que provienen de sensores, o que son ingresados por el usuario, pueden imprimir información en pantalla, o transmitir información por la red, dependiendo de qué es lo que el programador haya determinado.

Cabe destacar que, en general, los comandos incluidos en un lenguaje suelen realizar tareas sencillas y bien puntuales. Por ejemplo, es casi seguro que un lenguaje de programación no va a incluir un comando para calcular la hipotenusa de un triángulo sabiendo los catetos del mismo. Sin embargo, una persona con conocimientos del teorema de Pitágoras puede llegar al mismo resultado utilizando una serie de comandos más simples (multiplicación, radicación y sumas). A continuación hay un ejemplo de programa que calcula la hipotenusa de un triángulo dados sus catetos.

■ Ejemplo 4.7

código de ejemplo para el cálculo de hipotenusa de un triángulo

1. Leer del teclado el número que el usuario ingresa como C1
2. Leer del teclado el número que el usuario ingresa como C2
3. CSQ1 \leftarrow C1 * C1
4. CSQ2 \leftarrow C2 * C2
5. HSQ \leftarrow CSQ1 + CSQ2

6. $H \leftarrow \text{raíz de HSQ}$
7. Informar al usuario en pantalla el valor H

Definición 4.11 Un **lenguaje de programación** es un lenguaje formal que especifica una serie de instrucciones para que una computadora procese y produzca diversas clases de datos. Los lenguajes de programación se usan para crear programas de computadora.¹³

En este código “*” se utiliza como símbolo para multiplicar, y “←” indica que el resultado de la cuenta debe ser recordado con un nombre (que figura a la izquierda de la flecha) para poder ser utilizado posteriormente.

No existe un único lenguaje de programación, sino que hay cientos de ellos, cada uno con sus propias características, y diseñados para satisfacer necesidades específicas. Así, **no existe un “mejor lenguaje”**, sino que existen lenguajes más o menos apropiados para solucionar un determinado problema. Por otro lado, muchos lenguajes distintos permiten solucionar los mismos problemas, pero con diferentes enfoques, por lo que el **programador puede elegir aquel que más se adapte a su necesidad, pero también a sus preferencias personales**.

Pero, si las computadoras trabajaban con binario, ¿Cómo hacen para entender ese programa?. La respuesta radica en utilizar un programa. El programador genera un archivo con texto, similar al expuesto arriba, y lo guarda en el equipo. Luego, le indica a un programa especial, llamado **compilador**, que procese ese archivo para transformar el texto que contiene en una secuencia de unos y ceros que correspondan a un programa que realiza las tareas descriptas en el archivo. Al proceso mencionado, que realiza el compilador, se lo conoce como **compilar**.

El compilador transforma entonces una secuencia de unos y ceros que representan texto, en una secuencia de unos y ceros que representan un programa

El archivo que contiene el texto que describe el programa se conoce como **código fuente** del programa, mientras que el programa resultante se conoce como **código objeto** o **código compilado**.

Definición 4.12 Un **compilador** es un programa que transforma un archivo con código fuente en un archivo con código objeto, capaz de ser ejecutado en una computadora.

También existen algunos lenguajes que no trabajan de esta forma, sino que un programa va leyendo cada una de las líneas del programa, y lleva adelante la acción descripta para esa línea previo a leer la siguiente linea. Esto se conoce como **interpretación**, y el programa que realiza esa tarea se conoce como **intérprete**.

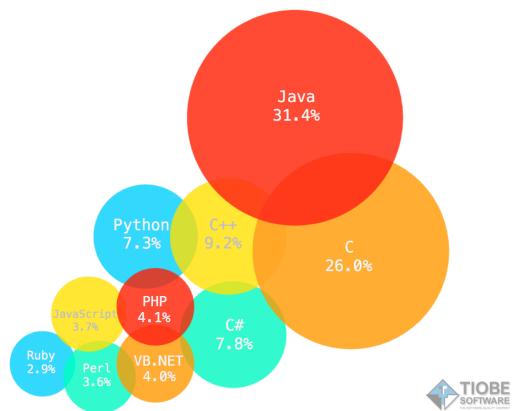


Gráfico de burbuja que muestra la popularidad de diferentes lenguajes de programación según el Índice TIOBE.

Imagen de Untapt con datos de TIOBE.

Definición 4.13 Un **interprete** es un programa que ejecuta paso a paso cada uno de los comandos descriptos en un archivo con código fuente.

Simplificación

La realidad es que el funcionamiento de los compiladores e interpretes es bastante más compleja. Hoy en día muchos interpretes realizan pasos de pre-compilación, y muchos compiladores funcionan con técnicas JIT. También se está obviando el tema de máquinas virtuales, lo cual representa un componente muy importante en los procesos de ejecución de programas actuales.

Estos temas merecen un trabajo mucho más en profundidad, así como un bagaje de conocimientos mucho mayor al que el presente intenta proporcionar.

Es importante a tener en cuenta que, ni los compiladores ni los interpretes trabajarán con cualquier texto, sino que esperan texto escrito en un lenguaje específico. Es decir, el compilador del lenguaje C, no funcionará si se intenta procesar con él un archivo con código Python. Así, cada lenguaje tiene su propio compilador, o su propio interprete (o ambos).

El lector puede o podría encontrar en diversa bibliografía términos como “**lenguajes compilados**” o “**lenguajes interpretados**”. Esta categorización responde simplemente al hecho de que, en general, el autor del lenguaje provee asociado a la definición su lenguaje (gramática, semántica, etc.), la herramienta que permite generar programas en dicho lenguaje (la cuál suele ser o bien un compilador, o bien un interprete). Sin embargo, nada evita que un lenguaje que era inicialmente distribuido con un interprete pueda ser luego distribuido con un compilador, sin alterar el lenguaje en sí. Esto quiere decir que la categorización anterior es incorrecta, pues ser compilado o interpretado no es una característica del lenguaje, sino de la herramienta que lo acompaña.

Otra categorización que suele realizarse es la que diferencia lenguajes entre aquellos que son de **bajo nivel** y los que son de **alto nivel**, distinguiendo incluso en ocasiones con un **medio nivel**. Los lenguajes de bajo nivel suelen proveer comandos que están asociados a las operaciones que puede realizar el hardware del equipo. Así, para cada comando, seguramente haya un circuito específico dentro del equipo. En cambio los lenguajes de alto nivel proveen construcciones más complejas, que no necesariamente se traducen a circuitos, y que están más ligadas a ideas o conceptos que el desarrollador puede querer expresar. Cuando el desarrollador realiza el programa, primero piensa la idea, la cual está expresada en lenguaje natural. Sin embargo, luego debe plasmar esa idea en el lenguaje de programación específico. Cuanto más parecido sea el lenguaje de programación a las ideas mentales del programador, más fácil será ese proceso.¹⁴

Una última categorización tiene que ver con la forma en la que el lenguaje permite estructurar ideas. Esto se conoce como el **paradigma** del lenguaje. Los primeros lenguajes de programación masivos utilizaban el **paradigma imperativo**, que consiste en secuencias de instrucciones dadas al equipo una tras otra. Existen otros paradigmas, como el **paradigma funcional**, más ligado a los desarrollos de teoría de la computación formal desarrollados por Alonzo Church, o la **programación orientada a objetos**, que utiliza procesos de abstracción conocidos como objetos para modelar los datos, y la comunicación entre ellos para modelar las transformaciones de esos datos. También existen otros paradigmas, más específicos y menos difundidos, y lenguajes que permiten escribir programas usando más de un paradigma al mismo tiempo.

4.3.3 Lenguajes de marcado

Otro tipo de lenguajes formales son los **lenguajes de marcado**. Estos lenguajes tienen como función describir datos, donde “datos” puede ser cualquier cosa. Por ejemplo, existen lenguajes

de marcado que describen publicaciones en foros, otros que describen páginas de wikis (como Wikipedia), otros que describen documentos, otros para imágenes, otros para páginas web, etc.

Recordemos que todas estas cosas podrían ser fácilmente representadas mediante archivos con diferente formato, esto es, con una codificación distinta del binario que representa el dato en el archivo. En los lenguajes de marcado, no se requieren diferentes formatos de archivo, sino que siempre se trabaja con texto plano, por lo que los datos **pueden ser editados con un editor de textos**.

La gracia es que el **texto contiene una serie de símbolos especiales, llamadas marcas** (de ahí el nombre de “lenguajes de marcado”). Estas marcas representan diferentes elementos semánticos dentro del documento, indicando por ejemplo, que una determinada parte se debe ver en negrita o en itálica, o que un determinado texto debería ser considerado un título, o que se debería insertar una tabla de datos en determinado lugar, o simplemente dando información semántica sobre la parte “marcada”. Estas marcas pueden luego ser interpretadas luego por un visualizador, que las mostrará como elementos visuales, o ser utilizadas por un programa para realizar tareas específicas de acuerdo a las mismas.

Existen cientos de lenguajes de marcado, cada uno intentando resolver una problemática distinta. Cada lenguaje define entonces qué símbolos representan marcas, y cómo deben ser utilizados en el texto, así como la asociación semántica de cada marca.¹⁵

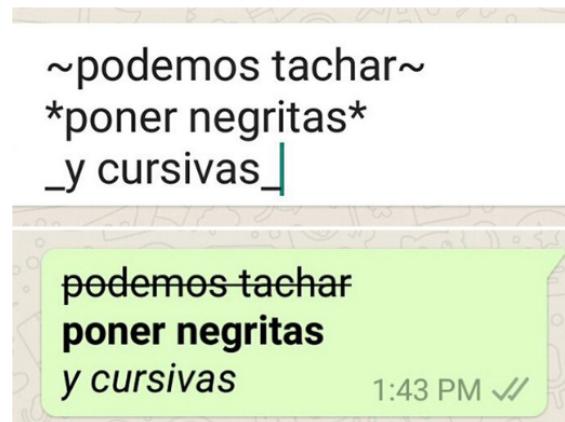
Definición 4.14 Un **lenguaje de marcado** consiste en una codificación que incorpora en un archivo de texto plano que junto con el texto incorpora una serie de marcas o etiquetas que contienen información sobre la semántica, la estructura o la forma de presentar el texto.

Sabías qué

Un archivo de texto solo contiene información del texto que contiene, es decir de las letras que están escritas. No maneja ninguna otra información, como si las letras estén en negrita, itálica o de algún color en particular. Sin embargo, muchos editores de texto diseñados para programar ofrecen una característica conocida como “syntax highlighting” o “code styling”, que colorea partes específicas del código, lo que le permite al desarrollador tener un panorama visual más ameno, facilitando su tarea.

Este libro utiliza coloreo en los lugares en donde se muestra código para hacer más fácil la comprensión del mismo.

Analicemos un ejemplo sencillo. El lenguaje de marcado de los **archivos de inicialización de Windows**, o archivos **.ini** consiste en texto que debe estructurarse en una serie de líneas. Cada línea consiste en un par de elementos, donde el primero, el de la izquierda, es conocido como “clave”,



Whatsapp trae incluido un lenguaje de marcas, que permite utilizar símbolos de texto simple al momento en el que el emisor redacta el mensaje para que el receptor visualice porciones del texto recibido con un formato particular.

Imagen propia.

y el segundo como “valor”. Ambos elementos se encuentran separados con un signo “=” entre ambos. Adicionalmente, un conjunto de líneas con pares clave valor pueden estar antecedidas con un nombre, colocado entre corchetes en la línea inmediatamente superior al conjunto, de forma tal de agruparlas bajo un mismo identificador. Finalmente, cualquier línea que comience con el símbolo “;” (punto y coma) corresponde a un comentario, y el texto de dicha línea no representa datos relevantes para el archivo, sino información para el desarrollador.¹⁶

Todas estas reglas pueden parecer complejas de entender, pero en realidad es uno de los lenguajes de marcado más sencillos que existe, aunque algunos autores consideran que es un lenguaje de configuración, más que de marcado.

Pero ¿Para qué sirve un archivo de inicialización de Windows? Bien, tanto el sistema operativo como diversas aplicaciones pueden utilizar el archivo para establecer preferencias del usuario. Por ejemplo, el código ficticio que se presenta a continuación podría utilizarse para configurar un navegador web:

■ Ejemplo 4.8

contenido hipotético de un archivo .INI

```

1 [Red] ; Poner UsarProxy=1 si no hay cortafuegos UsarProxy=1
      Proxy=192.168.0.1
2
3 [Preferencias] PaginaInicio=https://unq.edu.ar Maximizar=1

```

■

Las marcas en un archivo INI consisten en el símbolo “;”, que marca comentarios, o los caracteres “[” y “]” que determinan el nombre de un grupo, así como el signo “=” que separa la clave del valor en cada línea. El salto de línea, aunque no es un símbolo visible, también es considerado una marca, porque brinda información (termina un par clave valor y comienza otro).

Los archivos de configuración de Windows son un formato de archivo sencillo, pero hoy en día ya obsoleto.

Otros lenguajes de marcado incluyen **JSON** o **XML**, los cuales son utilizados para intercambios de datos en la web o configuración de aplicaciones más modernas. A continuación se deja un ejemplo sencillo de **XML**:¹⁷

■ Ejemplo 4.9

Una receta de cocina codificada en un archivo XML

```

1 <Receta> <Nombre> Banana con Dulce de Leche </Nombre>
      <Descripción> El mejor
      postre </Descripción> <Ingredientes> <Ingrediente> Banana
                  </Ingrediente>
      <Ingrediente> Dulce de Leche </Ingrediente> </Ingredientes>
      <Utensilios>
      <Utensilio> Cuchara </Utensilio> </Utensilios> <Pasos>
                  <Paso>Pelar la
      banana</Paso> <Paso>Destapar el frasco de dulce de leche</Paso>
      <Paso>Colocar la cuchara en el frasco</Paso> <Paso>Tomar una
                  gran cucharada

```

```
7     de dulce de leche</Paso> <Paso>Untar el dulce de leche en la  
8         banana</Paso>  
9     <Paso>Comer la banana</Paso> <Paso>Repetir el proceso si fuera  
9         necesario</Paso> <Paso>Cerrar frasco y lavar cuchara</Paso>  
10        </Pasos>  
10    </Receta>
```

El archivo anterior describe una receta de cocina de un típico postre argentino. El lenguaje XML contiene como marcas un elemento conocido como **etiqueta**. La etiqueta consiste en un nombre colocado entre los caracteres “<” y “>”, si es de apertura, o “</” y “>” si es de cierre. Cada etiqueta puede contener, o bien texto, o bien otras etiquetas.

XML permite definir los nombres de las etiquetas, y estructurarlas como se deseé. Por eso, este lenguaje es ampliamente utilizado para describir información. XML es sumamente flexible, y es considerado el padre de decenas de lenguajes de marcado. Por ejemplo, se utiliza para definir documentos y planillas de cálculo en los formatos de documento libre ODT y ODS (archivos similares a los de Word y Excel), como así también para describir imágenes en el formato SVG. La diferencia de estos lenguajes de marcado particulares con XML radica en que no cualquier etiqueta es válida, y no pueden tampoco anidarse de cualquier forma, sino que deben cumplir una serie de restricciones adicionales.

Dentro de los lenguajes que descienden de XML se encuentra **HTML**, o **HyperText Markup Language**, en español “lenguaje de marcado de hipertexto”. Este es tal vez el lenguaje de marcado más popular y más vastamente utilizado de la historia. Esto se debe a que HTML es el lenguaje con el que se **describen las páginas web**. Los elementos de una página web son estandarizados, y están dados por las etiquetas que el lenguaje define. Por ejemplo, la etiqueta “<a>” define un enlace a otro sitio web, o la etiqueta “” permite insertar una imagen.

Luego, un programa especializado, conocido como **browswer** o **navegador web**, actúa como visualizador de estos archivos, mostrando los elementos visuales que el autor haya escrito, sin mostrar las etiquetas.

La combinación de las distintas etiquetas, estructuradas de diversas formas, puede dar lugar a las más diversas y creativas páginas web que uno pueda desear. HTML se combina con otros lenguajes, como CSS, y JavaScript, para lograr sitios web interactivos y con grandes características visuales. Ejemplos y detalles de este lenguaje se pueden ver en el anexo ??.

Sabías qué

Casi todos los **browsers** permiten ver el **código fuente** de la página web que el usuario está navegando. No solo eso, sino que algunos incluyen herramientas que sirven a los desarrolladores para analizar, escribir, y reparar las páginas que escriben, permitiendo incluso modificar el código de páginas remotas de forma local y temporal.

Markdown es otro interesante lenguaje de marcado. Es ideal para escribir pequeños documentos informativos, como puede ser documentación, o correos electrónicos. La ventaja fundamental de Markdown radica en que no se requiere de un visualizador especializado para comprender el documento, sino que el mismo código fuente permite visualizar el contenido de forma clara. Un pequeño resumen del lenguaje se puede ver en el anexo ??.

4.4 Modelos de comercialización de software

El software puede ser comercializado de diversas formas en el mercado moderno. Tanto como producto o como servicio, además de generar servicios alrededor de la venta del software, como servicios de capacitación, mantenimiento, etc.

4.4.1 Copyright

Debe comprenderse que el software hoy en día goza de **derechos de autor** o **copyright**. Es decir que el autor de un programa goza de todos los derechos legales de explotación del mismo. Es decir, el autor puede determinar cosas como la forma en la que el software se vende, alquila o licencia, o cosas como para qué fines puede ser utilizado su programa, si puede ser utilizado como parte de otros programas, si puede ser utilizado comercialmente, etc.

Definición 4.15 El **derecho de autor** (o **copyright**, por su voz inglesa) consiste en un conjunto de normas jurídicas que defienden el derecho moral de los autores de obras literarias, artísticas, musicales, científicas, didácticas y de software, ya sea publicada o inédita, para explotar dicha obra comercialmente en cualquier forma que considere por un periodo de tiempo determinado.¹⁸

El modelo actual de comercialización de software nace en la época de 1970, con la aparición de las primeras microcomputadoras. Los fabricantes de estas primeras computadoras instalaban software de fábrica para venderlo como un valor agregado a sus equipos. Sin embargo, estos programas solo funcionaban en la máquina del fabricante, y por tanto uno quedaba atado a este para mantener compatibilidad de los productos. Además esos productos solían ser de baja calidad.

Con la masificación de las computadoras, se vuelve posible la venta de software como un producto o servicio de terceros, separado del fabricante. Microsoft es una de las empresas que promocionó este nuevo modelo de negocio, ofreciendo software de calidad a sus usuarios. Así, Microsoft Basic se volvió uno de los programas prácticamente imprescindibles a tener en un equipo a comienzos de 1980.



Oficinas de Microsoft en Barcelona.

Imagen de T Magazine (The New York Times Style Magazine: España).

Si bien un desarrollador puede optar por vender su software, el modelo de comercialización más común hoy en día es el modelo de **licenciamiento**. Es decir, quien adquiere un sistema, no lo compra de la misma forma que se adquiere un elemento físico, sino que lo que se adquiere es el **derecho a uso del sistema**. En ese sentido, el autor, llamado el **licenciatante**, le otorga al comprador, el **licenciatario**, el derecho de uso del programa. La **licencia** consiste entonces en un contrato legal entre estas partes, que especifica cosas como la forma en la que se puede utilizar el sistema, la forma en la que puede distribuirse, etc.

Definición 4.16 Una **licencia de software** es un contrato legal entre quien tiene los derechos de explotación del software (**licenciatante**) y quien suscribe para un uso determinado del mismo (**licenciatario**).

Romper el contrato de licencia, por ejemplo, usar un programa para un fin para el cual no se tiene permiso, o descargarlo de un sitio que no está oficialmente habilitado para su distribución, es **ilegal**. La descarga y distribución de programas de forma ilegal se conoce como **piratería**. Por ejemplo, descargar y utilizar Microsoft Windows sin haber pagado los cánones de licencia oficiales es considerado **piratería**.

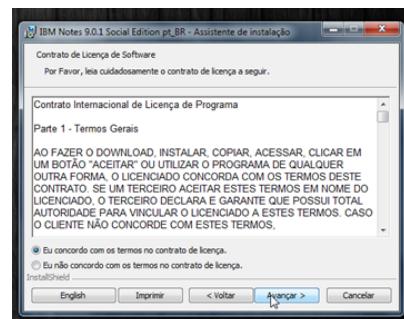
Definición 4.17 La **piratería de software** consiste en el incumplimiento del contrato legal que establece el propietario de los derechos de autor del software, tal como el uso indebido, descarga o adquisición en lugares no autorizados, redistribución del programa sin el consentimiento apropiado, prácticas de ingeniería inversa, modificaciones o alteraciones al programa original, entre otras.

En general, el licenciatario paga un canon o tarifa al momento de adquirir la licencia. Al pagarla, adquiere no solo el derecho a uso del sistema, sino en general el derecho a recibir las actualizaciones que aparezcan del mismo (muchas veces dentro de cierto rango de tiempo).

Algunos licenciantarios ofrecen el software de forma gratuita para uso dentro de cierto límite de tiempo (por ejemplo, 10 días de prueba del software). Esto permite a los potenciales compradores evaluar el producto previo a su compra, a la vez que posiciona potencialmente al desarrollador de una mejor forma frente al mercado. También puede suceder que la versión ofrecida de esta forma tenga limitaciones en su utilización, recortando características esenciales. Este tipo de software se conoce como **shareware**.

Definición 4.18 El **shareware** consiste en software que se ofrece al usuario final de forma gratuita por un periodo de tiempo, o con características limitadas, con la intención de que el usuario pueda robar el producto previo a la adquisición del mismo.¹⁹

Otros licenciatarios pueden ofrecer el software de forma completamente desinteresada, y por tanto gratuita. Este tipo de software se denomina **freeware**. No hay que confundir este término con software libre, que es un concepto completamente distinto. El freeware puede ofrecerse bajo ciertas restricciones,



Contrato de licencia de IBM Notes 9.0.1. Al presionar “Aceptar” el usuario está firmando un documento con validez legal, y cuyo incumplimiento puede llevarlo a que la contraparte lo demande.

Imagen de Lhreis.

por ejemplo, es gratuito, pero no si se usa comercialmente, o es gratuito, pero no puede distribuirse de otra forma que no sea mediante la tienda oficial del autor.

Definición 4.19 El **freeware** consiste en software que se ofrece al usuario final de forma gratuita, bajo un licencia que puede o no contener restricciones.²⁰

En general, el desarrollador solo ofrece el programa final en forma de **código objeto**, o dicho de otra forma, se ofrece el programa compilado, pero no su código fuente. Aquellos programas que adicionalmente ofrecen acceso al código fuente suelen denominarse **open source** o **software libre**, algo que trataremos en la siguiente sección.

Con la masificación de Internet, es cada vez más común encontrar sistemas que funcionan en línea, desde un navegador web. El licenciatario puede optar por ofrecer el servicio con un modelo de alquiler, cobrándole al licenciatario de forma mensual o anual por el servicio, o ofrecerlo de forma gratuita y solventando los gastos mediante la inserción de publicidad. Este modelo se conoce como **software como servicio (SaaS)**.

Otra modalidad muy común es el **software a medida**, donde el desarrollador no realiza un producto genérico para distribuir al mercado, sino que realiza un sistema especialmente diseñado para un cliente particular, a pedido de este. Dependiendo del contrato el mismo puede incluir solamente el código compilado, o también el código fuente, manejándose en general costos más elevados para este último. Junto con ello, suelen pautarse jornadas de capacitación para los usuarios del sistema, manuales y un periodo de soporte o garantía que puede pagarse junto con el desarrollo, o pagarse mensualmente luego de finalizado el mismo.²¹

4.4.2 Software Libre

Previo al actual modelo comercial del software, este era algo que se compartía entre los científicos e ingenieros en las universidades. La historia cuenta que Richard Stallman, un desarrollador de software se encontró ya a fines de los 70 trabajando en una empresa donde múltiples equipos estaban conectados en red con una única impresora. La impresora tenía el problema de trabarse de forma habitual, sin notificar al usuario del problema y este solo se enteraba de que había ocurrido un error cuando se acercaba a la impresora, la cual se encontraba en la otra punta de la oficina. Stallman quería solucionar el problema, y solicitó al fabricante de la impresora que le proporcionara el software de la misma. La empresa se negó a entregar el software, incluso cuando Stallman ofreció trabajar gratis para mejorar las características del mismo.

Stallman entendió que el **modelo de licenciamiento y comercialización de la época limitaba la libertad de los usuarios, que quedaban atados a la voluntad del fabricante**. Esto, según él, limitaba el progreso del software en general, y atentaba contra la libertad de las personas y la sociedad en su conjunto. Añorando el modelo de los inicios del software, anunció en 1983 que iba a comenzar a trabajar en desarrollar un sistema operativo completamente libre, en donde los usuarios no estuvieran limitados a ataduras por parte del licenciatario. En 1985 funda la **Free Software Foundation**, una fundación sin fines de lucro con la intención de llevar adelante su proyecto.

También enuncia el concepto de **software libre**. Un software, para Stallman, es libre si cumple con cuatro características que él denomina “libertades”.

- Libertad 0: Ejecutar el programa como se quiera y para el fin que se deseé.
- Libertad 1: Posibilidad de estudiar el código fuente del programa y modificarlo a medida

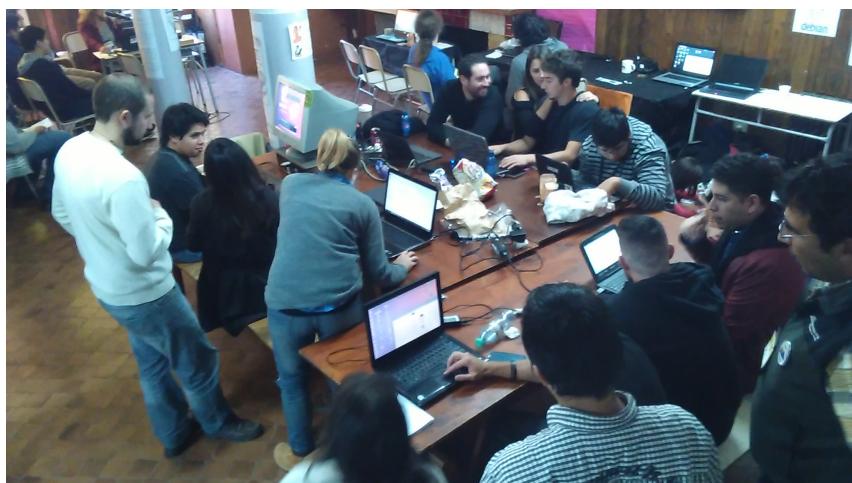
para solucionar las necesidades propias.

- Libertad 2: Distribuir copias exactas del programa, tal cual lo distribuye el fabricante a quien se deseé.
- Libertad 3: Distribuir copias modificadas del programa a quien se deseé.

Para cumplir estas libertades, todo software que se considere libre **debe garantizar el acceso al código fuente** a cualquier usuario. De lo contrario no sería posible llevar adelante las libertades 1 y 3. El software que no garantiza estas propiedades se conoce como **software propietario** (porque tiene un propietario, a diferencia del libre que le pertenece a la comunidad) o **software privativo** (porque priva al usuario de sus derechos).²²

Definición 4.20 El **software libre** es aquel que cumple con las cuatro libertades enunciadas por Richard Stallman, y que permite al usuario utilizar el programa, estudiarlo, compartirlo, modificarlo y compartir copias modificadas. Para ello el software libre debe permitir acceder a su código fuente.

Definición 4.21 El **software propietario, o software privativo** es todo aquel software que viola alguna de las libertades enunciadas por Richard Stallman.



El Festival Latinoamericano de Instalación de Software Libre (FLISoL) es un evento en el que usuarios de software libre ayudan a quienes se acercan a reemplazar el software privativo de sus equipos por software libre. El evento también cuenta con charlas de divulgación o técnicas y otras actividades. Se realiza en diversas sedes de múltiples países al mismo tiempo una vez al año. La imagen corresponde al evento realizado en el 2017 en la ciudad de Bariloche.
Imagen de Cristian Caravá, Jose Gimenez, Mirian Ripoll, Ariel Kennedy, Andrés Grad Gut, Stefan Ronacher, Lucas Passalacqua y Pedro Wood

Estas libertades permiten teóricamente que los usuarios sean quienes tienen el poder en el sistema, en lugar del autor del software. También crea la licencia **GPL (General Public Licence)**, un modelo de licenciamiento de software que reconoce al autor, a la vez que mantiene la garantía de estas libertades. La licencia **GPL** se considera una licencia **Copyleft**, concepto también creado por Stallman en contraposición al copyright, y que estipula que toda obra derivada de un producto con licencia copyleft debe necesariamente incluir la misma licencia. Así, uno no puede simplemente

tomar un programa con licencia GPL, modificarlo y comercializarlo como propio con una licencia distinta a la GPL. Esto conlleva que las modificaciones realizadas por cualquier usuario terminan siempre siendo beneficiosas a la sociedad toda.

Otras licencias aparecen a posteriori, como la licencia **MIT**, o la **BSD**, que también son consideradas licencias libres. Sin embargo estas no son licencias copyleft, permitiendo a otros tomar el código y empaquetarlo como un producto comercial. El código fuente de **Microsoft Windows**, por ejemplo, posee código fuente original del sistema operativo **BSD**, que posee la licencia del mismo nombre.

El software libre se corresponde a una ideología, ligada al humanismo y las corrientes de izquierda socialistas. El **fin último es el avance social, y no la comercialización del software**. Sin embargo, esto no evita que diversas empresas puedan dedicarse al software libre como emprendimiento rentable. En general **el modelo productivo del software libre implica la venta de soporte a terceros que utilizan el sistema, aunque también puede venderse el software** (por ejemplo, vender la versión compilada, aunque se deja libre el código fuente). Como no todos tienen los conocimientos técnicos para compilar el código, muchas personas pagan por la comodidad de no tener que aprender a hacerlo.

El proyecto de Stallman de crear un sistema operativo completamente libre nunca logró completarse. Si bien se crearon sendas herramientas que acompañan a un sistema operativo, la parte fundamental que maneja el hardware, conocido como kernel, jamás terminó de realizarse. En 1991, un joven finés llamado **Linus Torvalds** compartió en Internet su intención de llevar adelante el desarrollo de un kernel para un sistema operativo clon de UNIX que corriera en computadoras de escritorio, y compartió libremente su código fuente. Contactado por Stallman y convencido del movimiento de software libre, decidió licenciar su software con GPL, bautizándolo Linux. Este sistema operativo se ha vuelto hoy en día el sistema operativo libre más popular del mundo.

4.4.3 Open Source

Muchas empresas encontraron al modelo de software libre altamente beneficioso. Al publicar el código fuente cientos de desarrolladores pueden contribuir a mejorar el programa realizado, y lo hacen de forma desinteresada y gratuita. Esto implica menores costos para las empresas que llevan adelante este proceso, pues no deben mantener el sistema costeando desarrolladores, sino que es la comunidad la que lo mantiene.

Las empresas suelen tomar el software realizado como libre y empaquetarlo para la venta, en ocasiones agregando funcionalidades adicionales que no se encuentran disponibles en la versión libre, y que vuelven al sistema en otro más potente o más apetecible por los usuarios. En general se ofrecen los sistemas con un doble licenciamiento, comercial, por un lado, y libre, por otro.

Así, diversas empresas comenzaron a abrir su código, pero no por una visión filantrópica, sino por una cuestión comercial y económica. El software que se enmarca en este modelo se conoce como **Open Source**. En la práctica, el software open source y el software libre comparten las mismas características, pero la filosofía que subyace a ambas es completamente distinta. La **Open Source Initiative** fue creada como una ONG con la intención de regular las licencias que son consideradas open source, y que incluye a las licencias libres, pero agrega otro conjunto de licencias que violan en ocasiones las libertades básicas del software libre. Así, todo software libre es open source, pero no a la inversa.

4.4.4 Demistificación de frases sobre el FOSS

Al conjunto de sobre libre con software open source se lo suele denominar **FOSS** (Free Open Source Software).

A diferencia de lo que uno podría pensar, **software libre no significa software gratuito**. Si bien es cierto que gran parte del software libre se ofrece de forma gratuita, la realidad es que existe software gratuito que no es libre. Más aún, existe software libre que no es gratuito.

Durante los años han sido varias las empresas que han trabajado en el desarrollo de software, y algunas de ellas, como Microsoft, Apple o IBM han alcanzado gran éxito comercial gracias al software privativo y a prácticas de negocio que implicaban el aprisionamiento de los usuarios a un sistema, el monopolio y el lobby político. No es de extrañar que Richard Stallman y muchos otros seguidores del movimiento software libre vean a estas empresas como “malvadas”. Sin embargo estas empresas solo hacen lo que toda empresa debe hacer, intentar generar ganancia a sus inversionistas.

Sin embargo, también son varias las empresas que desarrollan y promueven el software libre o al menos el software open source. Google por ejemplo ha publicado numerosas herramientas para desarrolladores que se distribuyen como libres. En particular, el sistema operativo Android es un sistema operativo libre, basado en Linux. Facebook y Twitter también lo hacen. Los modelos de negocio de estas empresas pasan por la venta de servicios y publicidad, y no por la venta de software.

Otras empresas que han desarrollado múltiples FOSS incluyen a la ya desaparecida Sun Microsystems, Novell, Red Hat y Canonical. Estas empresas proveen servicios empresariales alrededor del software libre que construyen, dandole la posibilidad al usuario de elegir si desea pagar por profesionales capacitados por la empresa para mantener el software de acuerdo a sus necesidades, o desea capacitar su propio personal. También brindan capacitaciones y certificaciones a profesionales.

Es decir, es falso que quien hace FOSS no va a ganar dinero, o va a entregar un producto gratis. Existen formas múltiples de ganar dinero con el FOSS.

Muchas veces se realizan comparaciones entre el software libre y el software privativo. En general las comparaciones suelen estar guiadas con intereses particulares de uno u otro grupo. A continuación hay algunas de las proclamas que muchas veces se escuchan a favor y en contra del software libre en estas comparaciones, y algunas demistificaciones de las mismas:

El software libre es más barato. Si bien puede ser cierto en la mayoría de las ocasiones, algunos productos de software libre, se deben comprar. Por ejemplo, Red Hat Enterprise Linux, puede llegar incluso a ser más costoso que su contrapartida privativa en determinadas configuraciones, incluyendo soporte y otros.

El software libre corre en todas las plataformas. Completamente falso. Un programa FOSS solo corre en la plataforma para la cual fue diseñado, al igual que un sistema privativo. Grupos de desarrolladores independientes que gustan de la aplicación suelen portar la misma a diferentes plataformas, pero no es seguro que esto ocurra con cualquier programa.

El software libre es más seguro. La seguridad nada tiene que ver con lo libre o lo privativo. Se dice que Linux suele ser más seguro que Windows, y esto se usa de palanca para el



Linus Torvalds, creador de Linux, en la conferencia LinuxCon Europa 2014 en la ciudad de Düsseldorf.

Imagen de Krd.

alegato mencionado, pero en otros programas esto no es necesariamente así. La única ventaja realmente visible tiene que ver con el hecho de poder verificar el código fuente del software libre, que nos permite asegurarnos qué es lo que el programa realiza realmente con nuestros datos, mientras que el software privativo no lo permite.

El software libre es de menor calidad. Esto también es falso. Es cierto que en el software privativo grandes empresas colocan mucho dinero, con la intención de obtener un buen producto que puedan comercializar, mientras que el software libre se basa en el apoyo de algunos pocos desarrolladores que lo hacen en su tiempo libre. Sin embargo, como ya mencionamos, muchas empresas lucran también con el software libre, por lo que están dispuestas a invertir recursos en este. En general proyectos bien establecidos y con grandes bases de usuarios suelen tener una muy buena calidad, incluso a veces mayor que la de sus contrapartes privativas.

El software libre no requiere actualizaciones Todo el software tiene, y muchas veces requiere actualizaciones. A medida que el hardware se actualiza, el software debe actualizarse para soportar dicho hardware, lo mismo ocurre con programas que corren en un determinado sistema operativo, si este último cambia, puede que el programa deba cambiar. Además los desarrolladores actualizan sus programas dotándolos de nuevas funcionalidades, corrección de fallos y otros. Libre o privativo nada tiene que ver en el asunto.

El software libre no brinda soporte técnico En parte esto es cierto. En general el software libre se distribuye bajo un concepto “as is” (como está), que significa que el autor no se hace responsable por fallos o errores que tenga el programa, ni por los problemas que este pueda ocasionar. Sin embargo, el soporte existe, ya sea a través de foros o sitios de la comunidad de desarrolladores y usuarios o a través de servicios pagos que uno puede contratar. Los sistemas privativos en general tampoco ofrecen soporte, o se cobra un plus para poder acceder al mismo.

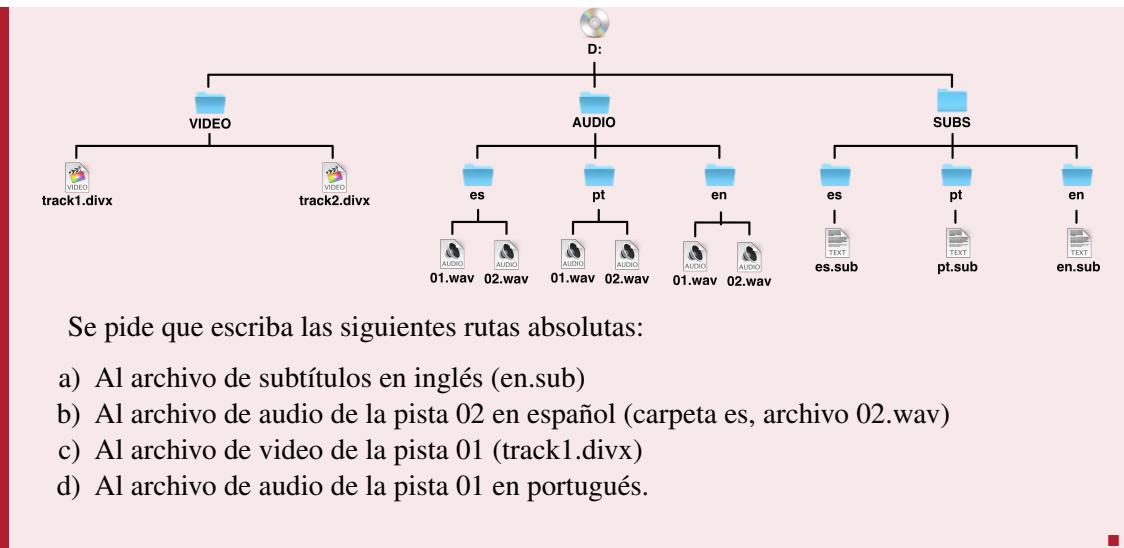
Probablemente estas no sean las únicas cosas que se dicen con poco criterio en el ámbito del software libre, sino que hay muchas otras. Entender como funcionan las computadoras y el software en particular permite discriminar mejor los postulados claramente erróneos de aquellos que no lo son, así como comprender los beneficios y las problemáticas de utilizar software libre.

4.5 Actividades

Ejercicio 4.1 Utilizando Internet, determine para cada uno de los nombres completos de archivo a continuación, de qué formato de archivo se trata (imagen, audio, texto, programa ejecutable, etc.).

- | | |
|------------------------------|----------------------------|
| a) ACDC.mp3 | j) showcase.odp |
| b) LEEME.txt | k) cuadernillo.pdf |
| c) 2018_06_15_221653.jpg | l) day_of_the_tentacle.exe |
| d) run.py | m) bumblebee.wav |
| e) index.php | n) library.c |
| f) cv.docx | ñ) configuration.xml |
| g) materias.xls | o) logo.png |
| h) subtitles_tbbt_s01e04.zip | p) LEEME.md |
| i) script.sh | q) casa.dwg |

Ejercicio 4.2 Teniendo en cuenta el siguiente sistema de archivos de Windows que corresponde a un DVD de una película.



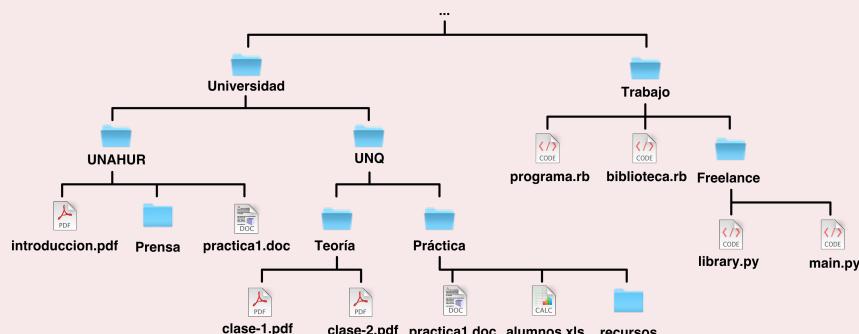
Ejercicio 4.3 Teniendo en cuenta el sistema de directorios del ejercicio anterior y sabiendo qué:

- Todo archivo de subtítulos pesa 500 KB.
- Los archivos de audio de la pista 01 pesan 5 MB.
- Los archivos de audio de la pista 02 pesan 25 MB, menos el de idioma portugués que pesa 27 MB.
- El archivo de video de la pista 01 pesa 340 MB
- El archivo de video de la pista 02 pesa 660 MB.

Se pide determine las siguientes:

- Cuanto pesa la carpeta VIDEO
- Cuanto pesa la carpeta SUBS
- Cuanto pesa en total el disco D:

Ejercicio 4.4 Teniendo en cuenta la siguiente porción de un sistema de archivos de un sistema Linux:



Describa la ruta absoluta a las carpetas que se encuentren vacías, asumiendo que la carpeta superior es la raíz del sistema.

Ejercicio 4.5 Utilizando el mismo sistema de archivos del ejercicio anterior, escriba las siguientes rutas relativas:

- a) Desde la carpeta más externa de la jerarquía hasta el archivo main.py
- b) Desde la carpeta más externa de la jerarquía hasta el archivo alumnos.xls
- c) Desde la carpeta UNQ hasta el archivo clase-2.pdf
- d) Desde la carpeta UNAHUR al archivo introducción.pdf
- e) Desde la carpeta UNQ al archivo introducción.pdf
- f) Desde la carpeta Teoría en UNQ al archivo programa.rb

Ejercicio 4.6 Un programa “hola mundo” consiste en código que simplemente imprime en la pantalla las palabras “hola mundo”. Este tipo de programa se ha hecho popular como una forma de poder dar a los programadores una vista mínima y rápida de la sintaxis de un lenguaje, y en cursos que se enfocan en la enseñanza de lenguajes de programación en lugar de sus conceptos, es muy común encontrarlo como una de las primeras muestras de código o actividades a realizar.

Se pide entonces que busque en internet el código de un programa “hola mundo” para los siguientes lenguajes de programación. Puede encontrar una gran colección en el sitio <http://helloworldcollection.de>

- a) Python
- b) Lisp
- c) C
- d) Java
- e) Assembly ARM
- f) Assembly z80, Console

¿Qué reflexiones puede realizar después de ver los diferentes códigos?

Ejercicio 4.7 El sitio web <https://alternativeto.net> permite a sus usuarios buscar un programa por nombre, y brinda alternativas a ese programa con funcionalidades equivalentes o similares. Dentro de los resultados, se puede filtrar por sistema operativo y por el tipo de licencia (comercial, gratuita o libre).

Se pide que busque al menos una alternativa que sea software libre y/o open source para cada uno de los siguientes programas:

- a) Windows 10
- b) Microsoft Office Suite
- c) Adobe Acrobat Reader
- d) Internet Explorer
- e) Adobe Photoshop
- f) Autodesk AutoCAD
- g) Age of Empires II

Referencias

- [1] Charles Petzold. *Code: The Hidden Language of Computer Hardware and Software*. Microsoft Press, 2000 (véase página 59).
- [2] Noam Nisan y Shimon Schocken. *The Elements of Computing Systems: Building a Modern Computer from First Principles*. MIT Press, 2005 (véase página 60).
- [3] Ron White y Tim Downs. *How Computers Work: The Evolution of Technology*. 10.^a edición. How it works. Que Publishing, 2015 (véanse páginas 60, 65, 66).
- [4] Philip A. Laplante. *Dictionary of Computer Science, Engineering and Technology*. CRC Press, 2000 (véanse páginas 62, 67).
- [5] ASCII Corporation y Nippon Gakki Co. LTD. *V9938 MSX Video: Technical Data Book Programmer's Guide*. Yamaha Nippon Gakki Co. LTD., 1985 (véase página 63).
- [6] Dan Gookin. *PCs For Dummies*. 10^a ed. For dummies. Hoboken, Nueva Jersey: Wiley, 2005 (véanse páginas 72, 75).
- [7] Linfo Team. «Filename Extension Definition». En: *The Linux Information Project* (2018). URL: http://www.linfo.org/filename_extension.html (véase página 73).
- [8] FOLDOC. «Absolute Path». En: *Free Online Dictionary of Computing* (2018). URL: <http://foldoc.org/absolute+path> (véase página 76).
- [9] FOLDOC. «Relative Pathname». En: *Free Online Dictionary of Computing* (2018). URL: <http://foldoc.org/relative+pathname> (véase página 79).
- [10] M. Nottingham. «RFC 7320». En: *Internet Engineering Task Force (IETF)* (2014). URL: <https://tools.ietf.org/html/rfc7320> (véase página 81).
- [11] Richard Littaur. *The Dothraki Language Dictionary*. Web published, 2017 (véase página 83).
- [12] K. V. N. Sunitha y N. Klyani. *Formal Languages and Automata Theory*. Pearson Education India, 2015 (véase página 84).
- [13] Robert W. Sebesta. *Concepts of programming languages*. 7.^a edición. Addison Wesley, 2005 (véase página 85).
- [14] Richard L. Wexelblat. *History of programming languages*. ACM monograph series. Nueva York: Academic Press, 1981 (véase página 86).
- [15] James H. Coombs, Allen H. Ranear y Steven J. DeRose. «Markup Systems and the Future of Scholarly Text Processing». En: *Communications of the ACM*. Universidad de Michigan, Detroit, Estados Unidos: ACM, 1987 (véase página 87).
- [16] Microsoft. «GetPrivateProfileString function». En: *Microsoft Developer Network* (2018). URL: [https://msdn.microsoft.com/en-us/library/windows/desktop/ms724353\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724353(v=vs.85).aspx) (véase página 88).
- [17] Benoit Marshal. *XML by example*. QUE, 2000 (véase página 88).
- [18] Copyright. *Oxford's dictionary*. <https://en.oxforddictionaries.com/definition/copyright>, 2018 (véase página 90).
- [19] Shareware. *Merriam-Webster's dictionary*. <https://www.merriam-webster.com/dictionary/shareware>, 2018 (véase página 91).
- [20] Freeware. *Merriam-Webster's dictionary*. <https://www.merriam-webster.com/dictionary/freeware>, 2018 (véase página 92).

-
- [21] Kul Bhushan C. Saxena, Swanand J. Deodhar y Mikko Ruohonen Ruohonen. *Business Model Innovation in Software Product Industry: Bringing Business to the Bazaar*. Management for Professionals. Springer India, 2017 (véase página 92).
 - [22] Richard M. Stallman y Joshua Gay. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Paramount, California: CreateSpace Independent Publishing Platform, 2009 (véase página 93).

Lógica

5 Introducción a la lógica 103

- 5.1 Aproximación a la lógica
- 5.2 Elementos constituyentes de la lógica
- 5.3 Conectivas
- 5.4 Utilidad de la lógica en la informática
- 5.5 La lógica como formuladora de preguntas
- 5.6 Precedencia de conectivas
- 5.7 Análisis de expresiones
- 5.8 Resumen
- 5.9 Actividades

6 Lógica proposicional 127

- 6.1 Proposiciones, atómicas y compuestas
- 6.2 Pasaje de texto a fórmulas
- 6.3 Razonamientos
- 6.4 Otras conectivas más complejas
- 6.5 Fórmulas bien formadas
- 6.6 Análisis de validez de razonamientos por tabla de verdad
- 6.7 Relación entre las premisas y la conclusión
- 6.8 Álgebra de Boole
- 6.9 Reglas de inferencia
- 6.10 Falacias bien conocidas
- 6.11 Límites de la lógica proposicional
- 6.12 Actividades

7 Lógica de predicados 157

- 7.1 Individuos, Propiedades y Relaciones
- 7.2 Predicados
- 7.3 Universo de discurso
- 7.4 Cuantificadores
- 7.5 Actividades



5. Introducción a la lógica

Red Lógica de un Enrutador con LEDs.

Fotografía de tschoenemeyer.

Construimos las computadoras para facilitarnos el realizar cálculos, algo que nos ha caracterizado como especie por milenios. Pero hacer cuentas no es la única característica propia de los seres humanos. Somos, al menos por lo que sabemos, la única especie capaz de producir información novedosa mediante el pensamiento y la razón.

La **razón** es la base de la ciencia. Es lo que nos ha permitido formular teorías, leyes, postulados, etc. y es lo que permite que todas las disciplinas y ciencias, tanto naturales, empíricas, formales, sociales o económicas, avancen constantemente. Aunque no solo aplica a la ciencia, pues razonamos de forma constante. Con nuestra mente detectamos patrones, realizamos conexiones entre información y llegamos a conclusiones sobre lo que sucedería en casos hipotéticos. Esto ha permitido a nuestra especie tomar decisiones que nos ayudan a preservar nuestra vida, ser más competitivos con respecto a otros, consiguiendo más y mejores recursos, y por tanto, evolucionar con ventaja por sobre otras especies.

La **lógica** es el estudio de esos procesos mentales que llevan a la razón. Es una ciencia compleja, con diversos enfoques, muchísima historia y ramificaciones.

Siendo la lógica una ciencia en si misma, su estudio completo abarcaría mucho más que las escasas páginas que le dedicaremos en esta unidad, mucho más que un único libro de hecho. Aquí solamente nos adentraremos en algunos conceptos fundamentales, que dan sustento a gran parte de los elementos de las ciencias de la computación, y que luego aplicaremos en la sección de programación, pero que también aparecerán en el futuro en múltiples lugares para quienes opten por dedicarse al estudio de alguna disciplina relacionada a las ciencias de la computación y la informática.

En esta unidad comenzaremos por entender la lógica desde los conocimientos previos que se espera el lector posea. Si bien indagaremos en definiciones, no entraremos en profundos formalismos, ni presentaremos un sistema lógico formal de análisis, dejando estos elementos para los siguientes capítulos.

5.1 Aproximación a la lógica

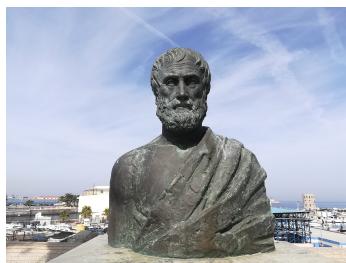
Antes de empezar a trabajar con la lógica como ciencia formal, empecemos por tener una aproximación sobre esta ciencia que, incluso sin saberlo, utilizamos de forma constante en nuestro día a día. Vamos a abordar primero de dónde surge, y luego, cuáles son algunos de los elementos constituyentes claves. A posteriori veremos por qué es útil conocer y entender de lógica para un programador.

Comencemos pues por el principio, de dónde surge la lógica.

5.1.1 Historia de la lógica

La idea de razonar es algo que define a la especie humana, y por tanto no es de extrañar que ya en la Antigua Grecia alrededor del Siglo IV a.C. los filósofos intentaran comprender cómo funciona el proceso del pensamiento y la razón. La palabra actual “lógica” deriva del griego antiguo *logiké* (*λογική*), que significa “dotada de razón, intelectual, argumentativa” y que a su vez viene de *lógos* (*λόγος*), “palabra, pensamiento, idea, argumento, razón o principio”. Así, la palabra “lógica” se refiere a la ciencia que tiene por estudio el pensamiento y la razón. Por supuesto, los griegos no fueron los únicos en realizar estudios sobre el tema, ya que tanto en China como en la India aparecieron de forma independiente tratados sobre el tema alrededor de la misma época que en Grecia.

Aristóteles fue el primero en formalizar los razonamientos, utilizando letras para representar términos y estableciendo las fórmulas válidas de los mismos. Además estudió los razonamientos inductivos, base de lo que constituye la ciencia experimental, cuya lógica está estrechamente ligada al método científico. Los islámicos, y luego los lógicos de la Edad Media continuarán el análisis aristotélico de la lógica, vinculado más a la filosofía que a otras ciencias.



Busto de bronce de Aristóteles en la ciudad de Ceuta, España, del artista Serrán Pagán.

Fotografía de Wikimedia Commons de CarlosVdeHasburgo

Ya en la modernidad, se comenzó a estudiar la lógica desde puntos de vista más formales, y vincular la ciencia a las matemáticas. Matemáticos como Gottlob Frege, Giuseppe Peano, Ernst Schröder y Charles Peirce extendieron los conceptos de la lógica aristotélica, agregando elementos como las variables o los cuantificadores “todo” y “algunos”, permitiendo revelar la estructura lógica de las oraciones en lenguajes naturales. Así, se iría extendiendo el estudio de esta ciencia, definiendo “lógicas” de distintos “ordenes” (primer y segundo orden).

Por su parte, George Boole, a quien ya hemos mencionado en este libro, publicó un breve tratado titulado “*El análisis matemático de la lógica*”, y en 1854 otro más importante titulado “*Las leyes del pensamiento*”. La idea de Boole fue construir a la lógica como un cálculo con propiedades similares a la matemática. Al mismo tiempo, Augustus De Morgan publica en 1847 su obra “*Lógica formal*”, donde introduce las “*leyes de De Morgan*” e intenta generalizar la noción de silogismo. Otro importante contribuyente fue John Venn, quien en 1881 publicó su libro “*Lógica Simbólica*”, donde introdujo los famosos diagramas de Venn.

Ya en 1910, a raíz de varias crisis en las bases fundamentales de la matemática, Bertrand Russell y Alfred Whitehead publican “*Principia mathematica*”, un trabajo en el que logran construir gran parte de las matemáticas a partir de la lógica, demostrando que la lógica es una ciencia subyacente a las matemáticas.

Más de 2000 años de historia nos llevan a la actualidad de la lógica como ciencia, que hoy sustenta cualquier desarrollo científico, tanto de las ciencias empíricas (biología, química, física, etc.) como formales (matemáticas y sus diversas ramas) y también las ciencias sociales (legales, comunicación, filosofía, etc.) y económicas. Hoy en día se entiende a la lógica como una ciencia que requiere un análisis formal (parecido a las matemáticas, como veremos luego) y que sigue métodos estrictos para su análisis.

Existen diversos “*típos de lógicas*”, como la lógica proposicional, la lógica de predicados, la lógica hermitica, lógicas modales, entre otras varias. También el enfoque de estudio varía, según la necesidad de la disciplina subyacente. Para las ciencias de la computación nos interesarán centrarnos en la lógica proposicional y la lógica de predicados. Pero necesitamos entender mejor qué es la lógica, como ciencia en general, antes de poder estudiar cualquiera de ellas.

5.1.2 El objeto de estudio de la lógica

Para poder comprender la **lógica** debemos comenzar por saber qué es y qué estudia. Para ello, veremos primero una definición simple.

Definición 5.1 La **lógica** es una ciencia formal que estudia los principios de la deducción de los razonamientos, así como su validez.

Pero para entender esta definición, primero debemos comprender qué es un **razonamiento**. Veamos una segunda definición para ello.

Definición 5.2 Un **razonamiento** es el producto de un **proceso mental** mediante el cual, a partir de una o más piezas de información de la cual se dispone, llamadas **premisas**, se arriba a una nueva pieza de información, llamada **conclusión**.

Es decir, el **razonamiento** es el producto de **razonar**, un proceso que produce el cerebro humano. En este proceso tomamos información previamente disponible, es decir, información ya conocida, para descomponerla, analizarla y procesarla, y arribar así a una nueva información.

Lo más fácil para comprender el concepto es ver un ejemplo concreto, por lo que recurriremos a un ejemplo clásico. Considere la siguiente información como **premisas**, es decir, como información ya conocida a priori.

■ **Ejemplo 5.1**

Todos los humanos son mortales.

Socrates es un humano.



Una **conclusión** posible sería determinar que “*Socrates es mortal*”. Nuestro cerebro puede realizar el análisis de las dos primeras oraciones y encontrar que hay elementos relacionados entre ellas, elucidando una información que no era conocida previamente (el hecho de que Socrates sea mortal).

Notar que si alguien pensara como conclusión que “*Socrates es inmortal*”, e intentara atravesarlo con una espada para probarlo, probablemente intentaríamos impedirlo inmediatamente. Es que de la información contenida en las premisas no hay forma de concluir la inmortalidad de Socrates, salvo mediante un proceso mental incorrecto. Así, la persona que arribó a dicha conclusión, no ha razonado de forma adecuada, llegando a una conclusión que no tiene sentido.

Los razonamientos pueden partir de una o más premisas, pero siempre tienen una única

conclusión. A veces nuestro cerebro realiza múltiples conclusiones a partir de un conjunto de premisas, pero con respecto a lo que concierne a la lógica estos se consideran múltiples razonamientos, y se debe estudiar cada uno de forma independiente.

La **deducción** es entonces el proceso que realizamos para partir de información conocida para arribar a nueva información. La deducción sigue formas bien definidas y precisas, y es por eso que podemos concluir en la mortalidad de Socrates, pero no en su inmortalidad. Si concluimos lo segundo, nuestro razonamiento sería **inválido**, mientras que por el contrario, si concluimos lo primero, podemos decir que es un razonamiento **válido**.

La lógica se va a encargar de estudiar entonces cuándo un razonamiento es válido, y cuándo no lo es.

Sabías qué

En realidad, nos estamos centrando en un tipo particular de razonamientos, llamados **razonamientos deductivos**, y por tanto estamos abusando de la simplificación en nuestras definiciones.

Existen otros tipos de razonamientos, como los inductivos (que surgen de razonar por inducción, en lugar de por deducción), y que pueden ser a su vez inductivos por enumeración, o inductivos por analogía. En el primero de este tipo de razonamientos se obtiene una conclusión global a partir de premisas que hablan de casos particulares (Por ej. de las premisas “*la droga X resultó exitosa para curar la enfermedad en el paciente 1*”, “*La droga X resultó exitosa para curar la enfermedad en el paciente 2*”, etc. se puede concluir que “*La droga X resultará exitosa para curar a todos los pacientes*”), en el segundo se obtiene un caso particular a partir de otros similares (Por ej. con las premisas anteriores arribar a “*La droga X resultará exitosa para curar al paciente N*”).

Este tipo de lógica es útil en las ciencias empíricas, donde a veces, el hecho de que se cumpla la conclusión el 99 % del tiempo es lo suficientemente bueno. Sin embargo, siempre pueden encontrarse excepciones a la regla. En el ejemplo anterior, pueden haber personas que sean alérgicas a la droga en cuestión, o que la droga no haga efecto en sus organismos.

Los razonamientos deductivos, en cambio, tienen una característica particular. **Si las premisas de las que partimos son verdaderas, un razonamiento bien realizado siempre arribará a una conclusión verdadera.** Esto es así ya que la información de la conclusión, en realidad está “oculta” en las premisas, es decir, está expresada en las premisas, aunque de otra forma, no evidente. Por eso este tipo de razonamientos es el empleado en las ciencias formales (como en la matemática), para realizar **demostraciones**.

Con esto tenemos más claro de qué se trata la lógica. Pero ¿A qué nos referimos con **ciencia formal**? Pues bien, se trata de que el objeto de estudio no son los elementos en sí, sino las formas de los elementos. Además de la lógica hay solo otra ciencia formal, la matemática. Veamos un ejemplo de esta última.

■ Ejemplo 5.2

Supongamos que tenemos 4 naranjas y decidimos comer una. ¿Cuántas naranjas nos quedan para comer? Esto los saben hasta los niños pequeños, nos quedan 3 naranjas.

Si de pronto tenemos 4 manzanas, y nos comemos una, sería ridículo pensar que debemos aprender algo nuevo para poder saber cuántas nos quedan.

Así, *la forma* de la operación a realizar no tiene que ver con naranjas o manzanas, sino con cantidades, con números. Las matemáticas nos enseñan que la forma es “ $4 - 1 = 3$ ”, y una vez aprendemos a solucionar ese problema, no nos importan qué sean los elementos.

Dicho de otra forma:

$$\text{Operación real: } 4 \text{ 🍊} - 1 \text{ 🍊} = 3 \text{ 🍊} \quad \text{Forma: } 4 - 1 = 3$$

y también:

$$\text{Operación real: } 4 \text{ 🍎} - 1 \text{ 🍎} = 3 \text{ 🍎} \quad \text{Forma: } 4 - 1 = 3$$

■

La lógica opera de forma similar. Pensemos en las premisas “*Todos los perros van al cielo*” y “*Firulais es un perro*”. La conclusión sería que “*Firulais va al cielo*”. La forma para este razonamiento, es la misma que para el razonamiento anterior sobre Socrates, que sería algo como “*Todos los x hacen/son algo*”, “*a es un x*” por lo que se puede concluir que “*a hace/es algo*”.

Por ahora no nos interesa analizar la estructura de un razonamiento en detalle, sino solamente ver que es posible encontrar una estructura idéntica en distintos razonamientos, para comprender mejor el concepto de “*forma*” al que nos referimos cuando hablamos de lógica.

5.2 Elementos constituyentes de la lógica

Una cosa importante a comprender es que no existe “una lógica” sino que hay distintos formalismos lógicos, cada una con sus particularidades, propiedades y sus utilidades de aplicación.

5.2.1 Valores de verdad

Finalmente, es muy importante mencionar que la lógica se basa en asignar lo que llamamos un **valor de verdad** a diversos elementos, de forma similar a como en las matemáticas se le asignan valores numéricos a los elementos. Esto permite realizar análisis sobre diversas situaciones para determinar si un razonamiento es válido o inválido.

Por lo tanto, debemos también entender de qué se trata un **valor de verdad**. Empecemos por una definición.

Definición 5.3 Los **valores de verdad** son valores binarios y dicotómicos, es decir, dos valores que son opuestos entre sí, y donde no hay terceras opciones ni respuestas que impliquen ambos valores al mismo tiempo.

En el lenguaje natural, es decir, cuando hablamos en español (u otro idioma), usamos en general las respuestas “*sí*” y “*no*” como valores de verdad. Pensemos en una pregunta cualquiera que pueda responderse con si o no, por ejemplo, “*¿Está lloviendo el día de hoy en la Ciudad de Buenos Aires?*”. Según el día y la hora que se esté leyendo esto, la respuesta concreta será o bien “*sí*” o bien “*no*”, pero seguro está lloviendo o bien no lo está, es decir, la respuesta es binaria. Más aún, los valores son opuestos, es decir, responder que “*si llueve*” es opuesto a responder que “*no llueve*”, y no puede darse que llueva y no llueva al mismo tiempo.

Tenemos otras formas de referirnos a ese “*sí*” en el lenguaje natural, por ejemplo “*afirmativo*”, “*cierto*”, o “*verdadero*”. De igual forma, tenemos otras palabras para “*no*”, como “*negativo*”, “*mentira*” o “*falso*”.

Al hablar de lógica en términos formales se utilizan los términos “**verdadero**” y “**falso**” para referirnos a los posibles valores de verdad que puede tomar un elemento.

Sabías qué

Otras disciplinas usar otros nombres para el mismo concepto de valor de verdad. Por ejemplo, la electrónica utiliza los nombres “1” y “0” para este concepto, y lo utilizan para representar la idea de “*es verdadero que hay electricidad*” o “*es falso que hay electricidad*”, cómo ya vimos en la unidad anterior.

Hablar de 1 y 0 es confuso para los no iniciados en la electrónica y en la lógica, ya que esos mismos símbolos se utilizan en la matemática, con un significado diferente, numérico.

La lógica entonces nos permite hablar de elementos verdaderos y falsos, de formas y estructuras para esos elementos, y de validez o invalidez de las combinaciones que hacemos sobre ellos, lo cual sirve para probar nuestros razonamientos.

En las próximas secciones vamos a trabajar sobre algunos elementos constituyentes de la lógica, que nos permiten combinar elementos, y vamos a entender mejor algunas utilidades prácticas de la lógica.

5.3 Conectivas

Desviémonos por un momento sobre la lógica como ciencia e imaginemos que en nuestro barrio han montado una moderna verdulería, atendida pura y exclusivamente por un robot. El verdulero robot es capaz de atender a los clientes, responder sobre el stock de las frutas y verduras, y venderlas, claro. Cómo es un robot, su repertorio de trabajo es limitado, y solo responde a cosas muy puntuales, y realizar preguntas que están fuera del repertorio del robot solo logra que este no nos responda.

Lo que nos va a interesar en esta analogía son las preguntas sobre el stock. La versión inicial del robot solo comprende preguntas simples sobre una fruta o verdura, para indicarnos si hay en stock, o no hay en stock. Así, solo podemos realizar preguntas de la forma “*¿hay X?*”, donde X es una fruta o verdura, por ejemplo, “*¿hay sandías?*” o “*¿hay cebolla?*”.

Ahora bien, un cierto día queremos preparar una ensalada de frutas, un popular postre que, bien preparado, debe llevar al menos tres ingredientes, manzana, banana y naranja. No disponemos de los elementos para la preparación en casa, por lo que nos aventuramos a la verdulería, a preguntar al robot si tiene stock de los ingredientes que necesitamos.

Por supuesto, no podemos preguntar al verdulero robot si tiene o no los ingredientes de una ensalada de frutas, pues nada sabe sobre cuestiones culinarias. Es decir, si bien nosotros queremos preguntarle al robot “*¿hay ingredientes para una ensalada de frutas?*”, no podemos hacerlo, y debemos recurrir a preguntas más simples, basadas en lo que el robot entiende. Por tanto debemos realizar tres preguntas independientes: “*¿hay manzanas?*”, “*¿hay bananas?*” y “*¿hay naranjas?*”.

El robot, por supuesto, podría responder con “*sí*” o “*no*” a cualquiera de esas preguntas, y es nuestro trabajo deducir cuándo se puede preparar la ensalada de frutas y cuándo no.

Pensemos en un posible escenario. El robot respondió con “*sí*” a “*¿hay manzanas?*”, con “*sí*” a “*¿hay bananas?*” y con “*no*” a “*¿hay naranjas?*”. En este caso, **no** podremos preparar la ensalada de frutas, pues nos faltaría un ingrediente, las naranjas. Podemos apreciar que la única respuesta viable a cada pregunta, para que podamos elaborar nuestro postre, debe ser “*sí*”.

Si pudieramos interactuar con un robot más avanzado, que entienda preguntas un poco más complejas, tal vez preguntaríamos algo del estilo “*¿hay manzanas, bananas y naranjas?*”, es decir, uniríamos las tres preguntas en una sola, usando una palabra especial para “**conectar**” esos

elementos, el “y” (gramaticalmente, la coma entre manzanas y bananas representa lo mismo que un “y”).

Pensemos otro ejemplo antes de ir a una definición. Si queremos preparar un bizcochuelo, debemos agregar la ralladura de un cítrico noble, como las naranjas o los limones. Nuevamente, no podemos ir al verdulero robot y preguntarle si tiene los ingredientes para un bizcochuelo, solo podemos preguntar por frutas específicas, “*¿hay limones?*” y “*¿hay naranjas?*”.

Analicemos una vez más las respuestas posibles. Si el verdulero nos responde con “*sí*” a ambas preguntas, podremos elegir que cítrico preferimos. Si responde a una con “*sí*” y la otra con “*no*” tendremos que usar el cítrico disponible (ya sea el limón o las naranjas), pero aún podremos preparar el bizcochuelo. Solo si ambas respuestas son “*no*” es que se nos hará imposible preparar el mismo.

En una versión más avanzada del robot preguntaríamos algo como “*¿hay limones o naranjas?*”. Notar que acá la palabra que usamos para conectar los elementos es “*o*”, y no “*y*”. Estas palabras para conectar son lo que llamamos **conectivas**, y tienen una connotación semántica asociada. No es lo mismo preguntar “*¿hay manzanas y bananas?*” que preguntar “*¿hay manzanas o bananas?*”. Podemos pensar estas preguntas como compuestas de partes más pequeñas, y la respuesta a ella depende de las respuestas individuales a cada una de las preguntas que la componen, así como de si usamos “*y*” o si usamos “*o*” para unirlas.

La siguiente tabla muestra en cada fila un escenario de posibles respuestas, donde el robot contesta a cada una de las preguntas individuales únicamente, y cómo interpretaríamos la pregunta compuesta según la conectiva elegida.

| <i>¿hay manzanas?</i> | <i>¿hay bananas?</i> | <i>¿hay manzanas y bananas?</i> | <i>¿hay manzanas o bananas?</i> |
|-----------------------|----------------------|---------------------------------|---------------------------------|
| Si | Si | Si | Si |
| Si | No | No | Si |
| No | Si | No | Si |
| No | No | No | No |

Así, la **conectiva** utilizada es importante, porque cambia la semántica de la pregunta. Esto es algo que probablemente nos resulte obvio de nuestra vida cotidiana, donde usamos las palabras “*y*” y “*o*” de forma habitual, y esperamos una connotación semántica identica (o al menos similar) a la arriba mencionada.

Con esta idea en mente estamos listos para dar una definición sobre las conectivas en términos de la lógica.

Definición 5.4 Una **conectiva** es un elemento que permite unir una o dos entidades portadoras de valor de verdad, para conformar una entidad más grande, cuyo valor de verdad depende de las entidades constituyentes y de la semántica asociada a la conectiva.

En estos sencillos ejemplos, podríamos pensar a nuestras preguntas como las unidades porta-



Sweeper, un robot que recolecta pimientos morrones.

Fotografía de
<http://www.sweeper-robot.eu>

doras de verdad, aunque veremos en los próximos capítulos a qué nos referimos con esto cuando veamos lógicas con sistemas formales específicos. Entonces, dicho de otra forma, y en nuestro ejemplo del verdulero robot, una conectiva permite unir una o dos preguntas para conformar una pregunta más grande.

Pensemos por un momento en las siguientes preguntas:

- “*¿hay manzanas y bananas?*”
- “*¿hay manzanas y también bananas?*”
- “*¿hay manzanas y además bananas?*”

Se puede apreciar como todas las formas de preguntar refieren a lo mismo. Para que la pregunta completa se responda como “*sí*”, esperaríamos que las respuestas individuales de “*¿hay manzanas?*” y “*¿hay bananas?*” sean “*sí*”. Los **lenguajes naturales**, como el español o el inglés, suelen ser muy ricos y diversos, e incluyen sinónimos y diversas construcciones gramaticales que son capaces de expresar la misma idea de distintas formas.

A partir de ahora entonces, cuando tengamos una conectiva, no vamos a optar por realizar una única pregunta con la conectiva en el medio (ej. “*¿hay manzanas y bananas?*”) sino que en su lugar vamos a preferir mencionar las preguntas que componen a ese elemento como preguntas individuales, de forma de dejar más claro la conectiva entre ellas (ej. “*¿hay manzanas? Y ¿hay bananas?*”). En la siguiente sección veremos que la lógica opta por utilizar símbolos específicos para cada conectiva, lo que evita las ambigüedades causadas por elementos gramaticales.

Pero ¿Por qué decimos que pueden unir una entidad sola? ¿Con qué se une? Bueno se une con la connotación semántica de la conectiva en sí misma. Volvamos a la verdulería robot y veamos un nuevo ejemplo.

Sofía es una nena muy glotona que disfruta mucho de comer la ensalada de frutas que prepara su mamá. Pero su mamá no puede preparar ensalada de frutas todos los días, ya que no siempre están disponibles los ingredientes en la verdulería. Sofía acaba de cumplir años, y comienza a sospechar de su madre, pensando que tal vez le dice que no hay ingredientes para preparar ensalada de frutas, cuando en realidad es solo que no tiene ganas de cocinar. Sin embargo, como ahora ya tiene edad suficiente para salir a la calle sola, decide ir a la verdulería por su cuenta para corroborar que su madre, quien acaba de decirle que no puede preparar ensalada de frutas porque no hay manzanas, está diciendo la verdad. Sofía se acerca a la verdulería y su primer intento de interacción con el robot verdulero consiste en preguntarle “*¿Es cierto que no hay manzanas?*”, respuesta que el robot responde sorpresivamente para los presentes con “*sí*”.

Se pregunta entonces al lector, ¿Estaba mintiendo la mamá de Sofía? Pensemos primero en la pregunta de Sofía y la respuesta del verdulero: “*¿Es cierto que no hay manzanas?*”, con la consiguiente respuesta “*sí*”. Ese “*sí*” no refiere a “*¿hay manzanas?*”, sino a si es cierto lo que Sofía está diciendo a continuación de “*es cierto qué*”, lo cual es “*no hay manzanas*”. Si la respuesta del verdulero fue “*sí*” quiere decir que es cierto, por lo que efectivamente “*no hay manzanas*”, mientras que si hubiera sido “*no*”, sería porque no es cierto que “*no hay manzanas*”, es decir, que habría manzanas.

Si bien parece un trabalenguas, lo que está expresando la pregunta de Sofía es una pregunta que está relacionada a “*¿hay manzanas?*” pero que es inversa a esta. Es decir, si la respuesta a “*¿hay manzanas?*” era “*sí*”, la respuesta a la pregunta de Sofía debería ser “*no*”, mientras que si la respuesta a “*¿hay manzanas?*” es “*no*”, la respuesta a la pregunta de Sofía debería ser “*sí*”.

En este ejemplo también hay una conectiva, la idea de cambiar la respuesta, o lo que en general llamamos “*no*”. Cuando trabajamos con preguntas en español, a veces uno tiende a preguntar cosas como “*¿no hay mayonesa?*”, esperando que nos respondan “*sí*” y nos pasen la mayonesa en el caso

de que hubiera (por “*si hay*” y no por “*si, es cierto que no hay*”). En español, así como en muchos otros idiomas, es raro usar la idea semántica de ese “*no*” en preguntas.

Sin embargo, si pensamos en afirmaciones, decir “*hay mayonesa*” y decir “*no hay mayonesa*” son claramente cosas distintas, siendo una afirmación opuesta a la otra, algo sobre lo que ahondaremos en los siguientes capítulos. Este es un ejemplo de una conectiva que trabaja sobre una única entidad portadora de valor de verdad, cambiando su valor. Veremos más sobre las distintas conectivas en la siguiente sección.

El concepto de **conectiva** es transversal a todas las formas de la lógica, ya que modela un elemento fundamental del lenguaje natural, que tiene una connotación semántica importante en las oraciones que componemos.

Cómo mencionamos al inicio de esta unidad, la gracia de la lógica consiste en poder analizar la forma de los razonamientos y sus elementos constituyentes, por lo que no nos interesa cual es la (o las) palabra exacta que se está utilizando como conectiva, sino cuál es la semántica que se le está dando a esa palabra. Esto permite identificar la conectiva, y por tanto entender mejor el rol de los elementos involucrados. Por eso nos interesa “**la forma**”, y no el contenido.

En esta sección vamos a profundizar y formalizar las conectivas que hemos trabajado en los ejemplos en la Sección 5.3. Si bien existen otras conectivas, abordaremos las mismas recién en el Capítulo 6.

Como parte de esta formalización, dejaremos atrás la idea de “*si*” y de “*no*” como respuestas a nuestras preguntas, para pasar a hablar en lenguaje lógico, teniendo entonces “*verdadero*” (**VERDADERO**, o simplemente **V**) y “*falso*” (**FALSO**, o simplemente **F**) en su lugar.

5.3.1 Conjunción

La **conjunción** es la conectiva que permite unir dos elementos formando uno más grande, cuyo valor de verdad es verdadero solo cuando **ambos elementos son también verdaderos**, y falso cuando cualquiera de los elementos que lo constituyen es falso. Hemos visto esta conectiva en el ejemplo de la ensalada de frutas, en expresiones como “*¿hay manzanas? Y ¿hay bananas?*”.

La conjunción se asocia en el español a la palabra “y”, pero también a otra serie de palabras o frases como: “*también*”, “*además*”, “*adicionalmente*”, “*en adición*”, “*más*”, “*incluso*”, “*inclusive*”, entre otras varias¹.

Ya vimos formas de expresar la misma pregunta a nivel semántico con elementos gramaticales o sintácticos diferentes en la sección anterior, y mencionamos también que a cada conectiva se le asocia un símbolo distinto. El símbolo para la conjunción es “ \wedge ”. Así, podríamos expresar la pregunta del ejemplo como “*¿hay manzanas? \wedge ¿hay bananas?*”.

Para entender mejor la conectiva podemos pensar en cualquier par de elementos portadores de valor de verdad genéricos, “ Φ ” (phi) y “ Ψ ” (psi), que son unidos por la conectiva de conjunción, quedando “ $\Phi \wedge \Psi$ ”. Así, podemos analizar el valor semántico de dicha conjunción de forma genérica, analizando el valor de verdad de cada una de las partes que lo constituyen.

- Si Φ es **VERDADERO** y Ψ es **VERDADERO**, entonces $\Phi \wedge \Psi$ es **VERDADERO**
- Si Φ es **VERDADERO** y Ψ es **FALSO**, entonces $\Phi \wedge \Psi$ es **FALSO**
- Si Φ es **FALSO** y Ψ es **VERDADERO**, entonces $\Phi \wedge \Psi$ es **FALSO**
- Si Φ es **FALSO** y Ψ es **FALSO**, entonces $\Phi \wedge \Psi$ es **FALSO**

Esto quiere decir que, si reemplazamos Φ y Ψ por preguntas cualquiera, y las unimos con una

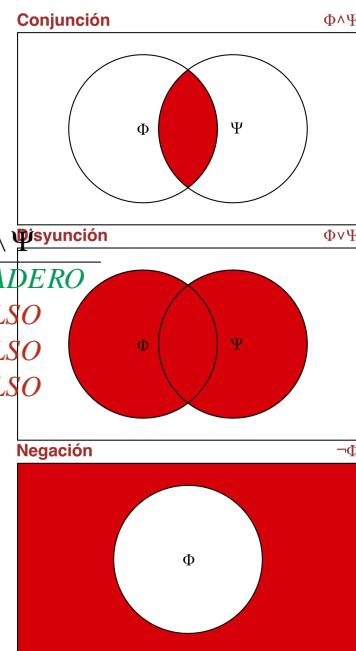
¹Puede verse una lista más amplia en el Anexo C

conjunción, la semántica asociada será la arriba expuesta, independientemente de la pregunta que usemos.

Podemos expresar más cómodamente cada uno de esos casos posibles y los resultados de la conjunción en una tabla, en donde hay una columna para cada uno de los elementos constituyentes de la conjunción (Φ y Ψ) y una columna para el resultado de la conjunción ($\Phi \wedge \Psi$). En cada fila se analiza una situación con valores de verdad distintos para los elementos constituyentes. Obtenemos entonces la siguiente tabla:

| Φ | Ψ | $\Phi \wedge \Psi$ |
|-----------|-----------|--------------------|
| VERDADERO | VERDADERO | VERDADERO |
| VERDADERO | FALSO | FALSO |
| FALSO | VERDADERO | FALSO |
| FALSO | VERDADERO | FALSO |

Llamamos a esta tabla la **tabla semántica de la conectiva**, ya que muestra la semántica asociada a la misma. Cada conectiva tendrá su propia tabla semántica.



Diagramas de Venn de las operaciones lógicas.
Elaboración propia.

5.3.2 Disyunción

Mientras que la conjunción requiere que ambos elementos sean verdaderos, la **disyunción** solo pide que **al menos alguno lo sea**. La disyunción es la idea del “*o uno, u otro, o ambos*”.

En general asociamos la disyunción con la palabra “*o*”, como en ejemplo del bizcochuelo que trabajamos con anterioridad, “*¿hay limones? O ¿hay naranjas?*”. Sin embargo, acá también pueden haber otras palabras o frases del español que pueden ser identificadas como conectivas de disyunción, como “*o también*”, “*o bien*”, “*aunque de pronto*” o construcciones como “*tal vez..., tal vez...*” o “*puede... o puede...*”²

En este caso el símbolo que se utiliza es \vee ³

La tabla semántica para la conectiva, con entidades genéricas, quedaría de la siguiente forma:

| Φ | Ψ | $\Phi \vee \Psi$ |
|-----------|-----------|------------------|
| VERDADERO | VERDADERO | VERDADERO |
| VERDADERO | FALSO | VERDADERO |
| FALSO | VERDADERO | VERDADERO |
| FALSO | VERDADERO | FALSO |

5.3.3 Negación

La conjunción y la disyunción son conectivas **binarias**, es decir, unen dos elementos. Por su lado, la **negación** es una conectiva **unaria**, es decir que trabaja sobre un único elemento.

La **negación** es la idea de “*no*”, en el lenguaje natural, aunque se pueden usar otras frases como

²Nuevamente se puede consultar el Anexo C para una lista más amplia.

³Notar que es el mismo símbolo de la conjunción, pero invertido, lo cual a veces trae confusión. Se recomienda mantener referencia de la simbología a mano hasta memorizarla. Puede ver la simbología en el Anexo C.

“no es cierto que”, “es falso que”, entre otras.

El símbolo es \neg y su semántica es la de **cambiar el valor de verdad de la entidad sobre la que opera**. Ya vimos el ejemplo de la pregunta de Sofía al verdulero robot, la que podemos expresar entonces como “ $\neg \text{ ``¿hay manzanas?''}$ ”.

La tabla de su semántica general es la siguiente:

| Φ | $\neg\Phi$ |
|-----------|------------|
| VERDADERO | FALSO |
| FALSO | VERDADERO |

Estas tres conectivas son las más simples que podemos encontrar, y las utilizamos de forma cotidiana en el lenguaje natural, pero como ya mencionamos, no son las únicas. Sin embargo, debemos primero acostumbrarnos a trabajar con conectivas y valores de verdad antes de poder ver conectivas más complejas.

5.4 Utilidad de la lógica en la informática

Cabe preguntarse después de entender el objeto de estudio y sus elementos básicos, cuál es el punto de estudiar esta ciencia en las disciplinas informáticas en general.

5.5 La lógica como formuladora de preguntas

Una característica interesante que podemos apreciar entonces es la capacidad que nos brindan las conectivas para pensar preguntas complejas en términos de otras preguntas más simples. Podemos decir por ejemplo que la pregunta “¿hay para preparar ensalada de fruta?” es lo mismo que preguntar “¿hay manzanas? \wedge ¿hay bananas? \wedge ¿hay naranjas?”. Lo mismo vale para la pregunta “¿hay para preparar bizcochuelo?” que queda expresada como “¿hay limones? \vee ¿hay naranjas?”.

Las computadoras, al igual que nuestro verdulero robot, tienen un repertorio limitado en cuánto a lo que son capaces de hacer y responder. De hecho, como ya vimos, solo hacen cuentas, y una “respuesta” de la computadora, no es más que electricidad (o ausencia de ella) en un cable específico. Incluso aplicando el concepto de caja negra, y subiendo varios niveles en la abstracción de las ciencias de la computación, las respuestas posibles de una computadora son limitadas.

Así, esta característica de la lógica es de especial relevancia en las ciencias de la computación, en diversas áreas, aunque vamos a centrarnos en este caso en el área de la programación.

Pensemos en una característica que tienen hoy muchos teléfonos celulares notebooks o computadoras de escritorio, cambiar el esquema de colores del equipo cuando se hace de noche, el llamado “modo oscuro”, que se activa de forma automática.

Para poder realizar un programa que active automáticamente el modo oscuro, un programador debería contar con que la computadora es capaz de responder una simple pregunta, “¿es de noche?”. Pero las computadoras en general, no incluyen circuitería específica para poder responder una pregunta tan abstracta. Pensemos además, que esa pregunta, se responde no solo diferente en diferentes horas del día, sino en diferentes días y regiones, por lo que no es posible realizar una solución en hardware para algo tan complejo.

De esta forma, un programador debe encontrar algún conjunto de preguntas que la computadora sepa responder, y que le sirvan para responder lo que verdaderamente le interesa, saber si “¿es de noche?”. Bien, las computadoras suelen incorporar un reloj, por lo que pueden decir la hora del día (asumiendo que la hora esté configurada adecuadamente). Además hoy en día es normal contar

con conexión a internet, y los servicios de sistemas meteorológicos nacionales suelen publicar en diversos lugares la hora del ocaso. Si pudieramos obtener ambos datos, saber si es o no de noche implica simplemente comparar la hora del ocaso con la hora del equipo (si el equipo tiene una hora posteriores, es de noche, sino, aún es de día).

Este ejemplo es tal vez un poco difícil de entender, pues además de preguntas que se responden con verdadero o falso hay también otros elementos involucrados, como fechas y comparaciones, pero es un ejemplo real que busca simplemente que el lector se haga una idea de cómo los programadores deben buscar “reformular” sus preguntas en términos de otros elementos disponibles en el equipo.

En la Unidad IV, dedicada a programación, veremos como esta capacidad de reformular preguntas se vuelve sumamente útil para resolver algunos problemas computacionales.

5.6 Precedencia de conectivas

Vimos cómo se pueden unir dos preguntas con conectivas de conjunción y disyunción, y como se le puede cambiar el valor de verdad mediante la negación. También vimos tablas para entender la semántica de dichas conectivas en casos genéricos, y cómo podemos tener preguntas más complejas como “*¿hay manzanas? \wedge ¿hay bananas? \wedge ¿hay naranjas?*”, que involucran más de dos preguntas pequeñas.

Vamos a aventurarnos a continuación en comprender como **operan** las conectivas lógicas en preguntas complejas, donde hay múltiples conectivas de distinto tipo, para, ya en la ?? poder entender la forma de operar con estos elementos. Para esto, vamos a llamar **elementos base** a cada pregunta simple, es decir, las preguntas que no tienen conectivas (Del estilo de las que el verdadero robot sabría responder) y **expresión** a una pregunta compleja, formada por uno o varios elementos base unidos con conectivas.

Y es que en primer lugar hay que entender que, para el análisis de la lógica en términos formales, las conectivas lógicas operan de forma similar a las operaciones matemáticas. Por ejemplo, la suma es también una operación binaria entre elementos, al igual que la conjunción, con la diferencia de que esos elementos son números, en lugar de elementos portadores de valor de verdad.

Veamos un caso concreto para entender como operar en situaciones con múltiples conectivas. En matemáticas estamos acostumbrados a trabajar con múltiples operaciones en una misma expresión, y tenemos convenciones para trabajar con las mismas para determinar de que forma llegar al resultado. Por ejemplo, podemos tener una cuenta expresada en la forma “ $((2 + 4) \times ((7 + 3)^2))$ ”. Si bien la expresión tiene muchos elementos y operadores entre ellos, los **paréntesis** ayudan a desambiguar el orden en el que debemos resolver cada operación. Sin embargo, no solemos escribir absolutamente todos los paréntesis. Por ejemplo, los más externos, nunca los escribimos, y tampoco el de la potenciación. Más aún, pensemos en la expresión “ $7 \times 5 + 4 \times 6$ ” y pensemos en la forma de resolverla. La multiplicación asocia más fuerte que la suma, debiendo resolverse primero (lo que llamamos coloquialmente “separar en términos”), por lo que incluso sin paréntesis involucrados, existe un orden en el que las operaciones deben realizarse. Por último, en el caso de una expresión como “ $2 + 3 + 4$ ”, existe un orden implícito de operación, de izquierda a derecha, por lo que en realidad es lo mismo que expresar “ $(2 + 3) + 4$ ”, y la forma de operar involucra solucionar primero “ $(2 + 3)$ ”, para luego, al resultado, sumarle “4”.

En lógica ocurren cosas muy similares. Si queremos entender el valor de verdad de una pregunta con varias partes, como “*¿hay manzanas? \wedge ¿hay bananas? \wedge ¿hay naranjas?*”, tenemos que comprender cómo se asocian esos elementos base, para interpretar las conectivas en el orden

correcto, y obtener un valor de verdad al final de las mismas que sea adecuado. El orden en el que realizamos las operaciones se conoce como **precedencia**.

Definición 5.5 La **precedencia** es el orden en el que se deben resolver los diversos operadores de una fórmula determinada, debiendo respetarla para una resolución adecuada.

Por un lado, y al igual que en matemáticas, al momento de formalizar, se pueden utilizar **paréntesis** para indicar de forma explícita la precedencia. Así, decir “ $(\text{¿hay manzanas?} \wedge \text{¿hay naranjas?}) \vee \text{¿hay limones?}$ ” no es lo mismo que decir “ $\text{¿hay manzanas?} \wedge (\text{¿hay naranjas?} \vee \text{¿hay limones?})$ ”. En cualquier caso, lo que está dentro de los paréntesis tiene precedencia por lo que está fuera, por lo que primero debería resolverse lo que está dentro. La semántica de cada una de esas expresiones es por tanto distinta⁴.

Podemos usar paréntesis para desambiguar la semántica de una expresión que involucra múltiples conectivas, pero la gran cantidad de paréntesis puede volverse confusa rápidamente. Por ello existen operaciones que tienen precedencia sobre otra (también se dice que tienen prioridad o más jerarquía). Esto es algo a lo que también estamos acostumbrados de las matemáticas, donde la expresión $2 + 3 \times 5$ debe ser resuelta primero por la multiplicación, y luego la suma, ya que la multiplicación tiene mayor precedencia.

En la lógica la precedencia es la siguiente:

1. Negación
2. Conjuncción y Disyunción

Es decir, si tenemos una expresión como “ $\text{¿hay manzanas?} \vee (\neg \text{¿hay naranjas?} \wedge \text{¿hay limones?})$ ”, ya que no hay paréntesis, la forma en que debe interpretarse es: primero la negación, que como es un operador unario, asociará un único elemento; luego como los elementos tienen igual precedencia, leeremos de izquierda a derecha, salvo que haya paréntesis, como es este caso. Entonces, la interpretación correcta es “ $\text{¿hay manzanas?} \vee ((\neg \text{¿hay naranjas?}) \wedge \text{¿hay limones?})$ ” y no “ $\text{¿hay manzanas?} \vee (\neg (\text{¿hay naranjas?} \wedge \text{¿hay limones?}))$ ”.

El orden de precedencia cobrará especial importancia cuando contemos con un sistema formal subyacente más elaborado y empecemos a formalizar todos los elementos que componen a nuestras expresiones, así como cuando deseemos realizar pruebas, comprender la validez de los razonamientos, etc. Por ahora, esta explicación debería ser suficiente para lo que nos convoca en este capítulo. Una versión completa con el resto de las conectivas existentes se presentará en el Anexo C.

5.7 Análisis de expresiones

En la sección anterior vimos que podemos tener múltiples preguntas unidas con varias conectivas, como en el caso de “ $\text{¿hay manzanas?} \vee \neg \text{¿hay naranjas?} \wedge \text{¿hay limones?}$ ”. También, que las operaciones de las conectivas son unarias o binarias, por lo que hay un orden en el que deben realizarse las mismas al momento de interpretar las expresiones completas de forma semántica.

Pero como hacemos entonces poder saber el valor de verdad que tiene una expresión compuesta de varias preguntas y conectivas. Es decir, ante una serie de respuestas concretas a las preguntas que componen una expresión, por ejemplo, si sabemos que la respuesta a “ ¿hay manzanas? ” es “**VERDADERO**”, a “ ¿hay naranjas? ” es “**FALSO**” y a “ ¿hay limones? ” es “**FALSO**”, ¿Cuál es el

⁴Notar que no existe una propiedad asociativa entre conectivas distintas, aunque si existen para algunas conectivas. Veremos propiedades de algunas conectivas en el Capítulo 6.

valor de verdad de la expresión? ¿Cómo llego a dicho valor? Lo que necesitamos es una forma de “calcular” dicho valor.

En primer lugar debemos comprender otro concepto clave de la lógica, la idea de **valuación**, de la que veremos primero su definición.

Definición 5.6 Una **valuación** es la asignación de un valor de verdad concreto a cada elemento base portador de verdad que participa en una expresión concreta.

Un ejemplo de valuación sería decir que “¿*hay manzanas?*” es “**VERDADERO**”, “¿*hay naranjas?*” es “**FALSO**” y “¿*hay limones?*” es “**FALSO**”. Por otro lado, si “¿*hay manzanas?*” es “**FALSO**”, “¿*hay naranjas?*” es “**VERDADERO**” y “¿*hay limones?*” es “**FALSO**” entonces tenemos otra valuación, diferente a la primera.

Es decir, una valuación consiste en asignar valores de verdad concretos a cada una de los elementos que componen la expresión a evaluar. La cantidad de valuaciones posibles de una expresión está dada por la cantidad de elementos base en la misma, es decir, la cantidad de preguntas elementales a responder.

Si por ejemplo, hay una única pregunta, entonces tendremos solamente dos valuaciones posibles (o bien le asignamos verdadero o bien falso a dicha pregunta). Si tenemos dos preguntas, pasaremos a tener cuatro valuaciones posibles, y si tenemos tres preguntas, serán ocho. Como regla general podemos decir entonces que:

Corolario 5.7.1 cantidad de valuaciones diferentes = $2^{\text{cantidad de elementos base}}$

Así, si deseamos conocer el valor de la expresión para una valuación concreta, se puede realizar un análisis semántico mediante resolución operacional, paso a paso para cada conectiva. Sin embargo, no es la única forma de analizar una expresión compleja, ya que en ocasiones queremos poder resolver la expresión para todas las valuaciones posibles al mismo tiempo, para lo cual necesitaremos una herramienta llamada tabla de verdad.

5.7.1 Análisis mediante semántica

Si sabemos los valores concretos de los elementos base, es decir, **si contamos con una valuación**, podemos analizar la expresión completa “**resolviendo**” cada una de las conectivas. Para ello, nos valdremos de las tablas semánticas de cada una de las conectivas y de la descomposición de la expresión en partes.

Veamos el ejemplo de la expresión antes mencionada, “¿*hay manzanas?* $\vee \neg$ ¿*hay naranjas?* \wedge ¿*hay limones?*”. El primer paso es determinar la precedencia, para saber el orden en el que debemos evaluar los distintos elementos. Ya hemos realizado este paso en la sección anterior, y determinamos la precedencia como “(¿*hay manzanas?* \vee ((\neg ¿*hay naranjas?*) \wedge ¿*hay limones?*))”. Con esto en mano, podemos resolver desde los paréntesis más internos a los más externos, reemplazando para cada pregunta por el valor de verdad correspondiente para la valuación que se está analizando. En este caso iremos por la valuación donde “¿*hay manzanas?*” es “**VERDADERO**”, “¿*hay naranjas?*” es “**FALSO**” y “¿*hay limones?*” es “**FALSO**”.

Ejemplo 5.3

Valuación a analizar:

“¿*hay manzanas?*” es “**VERDADERO**”

“¿*hay naranjas?*” es “**FALSO**”

“¿hay limones?” es “**FALSO**”

Expresión a analizar:

“¿hay manzanas? $\vee \neg$ ¿hay naranjas? \wedge ¿hay limones?”

Expresión con precedencia analizada:

“(¿hay manzanas? $\vee ((\neg$ ¿hay naranjas?) \wedge ¿hay limones?))”

Resolución paso a paso:

“(¿hay manzanas? $\vee ((\neg$ ¿hay naranjas?) \wedge ¿hay limones?))”

→ Se reemplaza ¿hay naranjas? por el valor que tiene en la valuación, **FALSO**.

“(¿hay manzanas? $\vee ((\neg$ **FALSO**) \wedge ¿hay limones?))”

→ Se resuelve el \neg según su tabla semántica, para **FALSO** la tabla indica que la negación debe ser **VERDADERO**.

“(¿hay manzanas? \vee (**VERDADERO** \wedge ¿hay limones?))”

→ Se reemplaza ¿hay limones? por el valor que tiene en la valuación, **FALSO**.

“(¿hay manzanas? \vee (**VERDADERO** \wedge **FALSO**))”

→ Se resuelve el \wedge según su tabla semántica, para el caso donde el primer elemento es **VERDADERO** y el segundo es **FALSO** la tabla indica que la conjunción debe ser **FALSO**.

“(¿hay manzanas? \vee **FALSO**)”

→ Se reemplaza ¿hay manzana? por el valor que tiene en la valuación, **VERDADERO**.

“(**VERDADERO** \vee **FALSO**)”

→ Se resuelve el \vee según su tabla semántica, para el caso donde el primer elemento es **VERDADERO** y el segundo es **FALSO** la tabla indica que la disyunción debe ser **VERDADERO**.

“VERDADERO”

■

Como se puede apreciar, la forma de resolver es muy similar a lo que uno realizaría en matemáticas, con la salvedad de que aquí no hay números, sino valores de verdad. Conocer las operaciones, y el resultado que se obtiene de cada una ante ciertos valores de verdad, es importante para poder resolver cada paso, y se puede recurrir a las tablas semánticas de cada conectiva para ello.

Sin embargo, esta metodología tiene la limitación de solo permitir determinar el valor de verdad final para una única valuación. Si se desea saber el resultado para otra valuación, se deberá volver a realizar todos los pasos, reemplazando ahora cada pregunta por los valores de la nueva valuación.

Recordemos que la cantidad de valuaciones es de dos, elevado a la cantidad de elementos base de la expresión. Para expresiones con dos elementos base, ya serán cuatro veces las que deberemos realizar el análisis, algo poco viable.

Sin embargo, es ideal para cuando queremos determinar el valor de verdad de la expresión sabiendo exactamente los valores de los elementos que la componen. Por ejemplo, si alguien me respondió las preguntas.

Si lo que se desea es poder analizar todas las posibles valuaciones al mismo tiempo, se debe recurrir a otra herramienta, las tablas de verdad.

5.7.2 Tablas de verdad

Una **tabla de verdad** permite visualizar y analizar todas las posibles valuaciones, y como estas influyen en la expresión final y en cada una de sus partes. Consiste en una tabla de doble entrada, en donde cada columna representa un elemento portador de valor de verdad (un elemento base o una expresión compuesta por conectivas) y cada fila una posible valuación.

La tabla de verdad es ideal para poder analizar expresiones donde hay varios elementos base, pues permite una más rápida resolución para cada valuación que la resolución paso a paso, pero también es útil para casos en donde no sabemos cuál es la valuación que nos interesa. Por ejemplo, puede que nos interese saber cuales son las valuaciones que hacen la expresión verdadera, y cuales no.

Aún así, necesitaremos tantas filas como valuaciones posibles, lo cual hace que si tenemos muchas preguntas, las tablas de verdad dejen de ser una gran herramienta para el análisis, pues ante 5 preguntas ya habrá que analizar 32 valuaciones diferentes, es decir, terminaremos con una tabla de 32 filas. Existen otras formas de análisis que no veremos en esta sección.

Para ver cómo armar la tabla de verdad, resolveremos nuevamente el ejemplo de la expresión “*¿hay manzanas? $\vee \neg$ ¿hay naranjas? \wedge ¿hay limones?*”. Ya sabemos que esta expresión contiene tres elementos base, y serán por tanto ocho las valuaciones posibles (2^3). Para comenzar a armar nuestra tabla deberemos arrancar entonces por tres columnas, una para cada pregunta elemental, y 8 filas, que de momento estarán vacías.

| <i>¿hay manzanas?</i> | <i>¿hay naranjas?</i> | <i>¿hay limones?</i> |
|-----------------------|-----------------------|----------------------|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Ahora hay que completar esta parte de la tabla y asegurarnos de que contamos con todas las valuaciones posibles, así como que no estamos duplicando ninguna valuación. Para ello podemos completarlo de forma mecánica siguiendo los pasos enunciados a continuación: Comenzando por la primer columna, completamos la mitad de las filas con *VERDADERO* y la otra mitad con *FALSO*, quedando la tabla como la siguiente.

| <i>¿hay manzanas?</i> | <i>¿hay naranjas?</i> | <i>¿hay limones?</i> |
|-----------------------|-----------------------|----------------------|
| <i>V</i> | | |
| <i>F</i> | | |

Pasamos a la segunda columna y hacemos lo mismo, pero esta vez completamos un cuarto de los casos con *VERDADERO*, otro cuarto con *FALSO*, y luego repetimos un cuarto más con

VERDADERO y otro cuarto con **FALSO**, quedando así.

| <i>¿hay manzanas?</i> | <i>¿hay naranjas?</i> | <i>¿hay limones?</i> |
|-----------------------|-----------------------|----------------------|
| V | V | |
| V | V | |
| V | F | |
| V | F | |
| F | V | |
| F | V | |
| F | F | |
| F | F | |

Volvemos a repetir el proceso, pero ahora tomando de a un octavo, con el primer octavo como **VERDADERO**, el segundo como **FALSO** y repitiendo. La tabla quedará así.

| <i>¿hay manzanas?</i> | <i>¿hay naranjas?</i> | <i>¿hay limones?</i> |
|-----------------------|-----------------------|----------------------|
| V | V | V |
| V | V | F |
| V | F | V |
| V | F | F |
| F | V | V |
| F | V | F |
| F | F | V |
| F | F | F |

De esta forma obtenemos todas las posibles valuaciones (notar como no hay dos filas que se repitan en la tabla). Esta dinámica puede aplicarse independientemente de la tabla y la cantidad de filas que tenga. Por ejemplo, si hubiera solo dos preguntas, habrá solo 4 filas y 2 columnas, la primera se completará tomando mitad y mitad, y la segunda tomando de a cuartos, para completar toda la tabla. Si fueran cuatro preguntas, tendríamos 16 filas, y habría un cuarto paso, tomando de a dieciseisavos.

Ahora estamos listos para resolver las conectivas de la tabla. El paso siguiente es ir agregando columnas, una para cada operación, en el orden en que deben resolverse según la precedencia. Ya hemos analizado la precedencia de la expresión, y sabemos que la primera operación a resolver en este ejemplo es “ $\neg \text{ } \langle \text{¿hay naranjas?} \rangle$ ”. Por esto agregamos dicha columna a la tabla.

| <i>¿hay manzanas?</i> | <i>¿hay naranjas?</i> | <i>¿hay limones?</i> | $\neg \text{ } \langle \text{¿hay naranjas?} \rangle$ |
|-----------------------|-----------------------|----------------------|---|
| V | V | V | |
| V | V | F | |
| V | F | V | |
| V | F | F | |
| F | V | V | |
| F | V | F | |
| F | F | V | |
| F | F | F | |

Para completar la columna “ $\neg \text{ } \langle \text{¿hay naranjas?} \rangle$ ” debemos mirar la o las preguntas involucradas en dicha operación. En este caso, la única parte que aparece es “*¿hay naranjas?*”, es decir, la que figura en la segunda columna de nuestra tabla. Así, deberemos mirar dicha columna y aplicar para cada fila la semántica de la conectiva de negación.

La negación nos indica en su tabla semántica genérica que, para cualquier cosa que negemos, si el valor era verdadero, debe pasar a ser falso, y si era falso, debe pasar a ser verdadero. Luego, para completar la columna fila a fila, debemos ir mirando el valor de la segunda columna para dicha fila, y aplicando el cambio indicado por la conectiva en su tabla semántica. Por ejemplo, en la primer fila el valor en la segunda columna es **VERDADERO** por lo que deberemos completar en la cuarta

columna con **FALSO**. Si seguimos el proceso para cada fila de la tabla tendremos⁵.

| ¿hay manzanas? | ¿hay naranjas? | ¿hay limones? | $\neg \text{¿hay naranjas?}$ |
|-------------------------|-------------------------|------------------------|------------------------------|
| V | V | V | F |
| V | V | F | F |
| V | F | V | V |
| V | F | F | V |
| F | V | V | F |
| F | V | F | F |
| F | F | V | V |
| F | F | F | V |

Continuamos con la operación “ $\neg \text{¿hay naranjas?} \wedge \text{¿hay limones?}$ ”. En este caso, las columnas involucradas son la cuarta y la tercera. Es decir, el resultado de “ $\neg \text{¿hay naranjas?}$ ” se usa para calcular “ $\neg \text{¿hay naranjas?} \wedge \text{¿hay limones?}$ ”. Nuevamente entonces agregamos una columna, y seguimos la semántica, en este caso, de la conjunción, que nos dice que para cada fila, debemos completar con verdadero solo cuando ambas columnas involucradas son verdaderas, y con falso en cualquier otro caso. Si seguimos esto obtendremos:

| ¿hay manzanas? | ¿hay naranjas? | ¿hay limones? | $\neg \text{¿hay naranjas?}$ | $\neg \text{¿hay naranjas?} \wedge \text{¿hay limones?}$ |
|-------------------------|-------------------------|------------------------|------------------------------|--|
| V | V | V | F | F |
| V | V | F | F | F |
| V | F | V | V | V |
| V | F | F | V | F |
| F | V | V | F | F |
| F | V | F | F | F |
| F | F | V | V | V |
| F | F | F | V | F |

La última operación de nuestra expresión es “ $\text{¿hay manzanas?} \vee \neg \text{¿hay naranjas?} \wedge \text{¿hay limones?}$ ”, es decir, la expresión completa, e implica resolver la disyunción entre la primera columna, y la quinta columna, que acabamos de completar. Semánticamente la disyunción nos dice en su tabla que debemos completar con verdadero cualquier fila que, en alguna de las columnas involucradas, tenga por valor verdadero, y falso solamente cuando en ambos casos es falso.

| ¿hay manzanas? | ¿hay naranjas? | ¿hay limones? | $\neg \text{¿hay naranjas?}$ | $\text{¿hay manzanas?} \wedge \neg \text{¿hay naranjas?}$ | $\text{¿hay manzanas?} \vee \neg \text{¿hay naranjas?} \wedge \text{¿hay limones?}$ |
|-------------------------|-------------------------|------------------------|------------------------------|---|---|
| V | V | V | F | F | V |
| V | V | F | F | F | V |
| V | F | V | V | V | V |
| V | F | F | V | F | V |
| F | V | V | F | F | F |
| F | V | F | F | F | F |
| F | F | V | V | V | V |
| F | F | F | V | F | F |

La última columna nos permite visualizar en que casos (en que valuaciones) la expresión completa es verdadera, y en que casos no lo es. De hecho, salvo las columnas para los elementos base, y la última columna que muestra la expresión completa, el resto de las columnas no son en general parte de interés, pues solo se realizan como pasos intermedios para llegar al valor de la expresión. Es decir, no podemos prescindir de elaborar esas columnas, porque sin ellas no podríamos arribar al resultado, pero la realidad es que, una vez completa la tabla, podemos eliminar esas columnas intermedias. Así, la tabla anterior quedaría como:

⁵Marcamos en la tabla con color celeste la columna que acabamos de completar, y en amarilla la columna que miramos para poder hacerlo. El color solo aporta mayor claridad al paso que acabamos de mencionar, pero no es significativo en el proceso de resolución.

| $\varphi \text{ hay manzanas?}$ | $\psi \text{ hay naranjas?}$ | $\chi \text{ hay limones?}$ | $\varphi \text{ hay manzanas?} \vee \neg \psi \text{ hay naranjas?} \wedge \chi \text{ hay limones?}$ |
|---------------------------------|------------------------------|-----------------------------|---|
| V | V | V | V |
| V | V | F | V |
| V | F | V | V |
| V | F | F | V |
| F | V | V | F |
| F | V | F | F |
| F | F | V | V |
| F | F | F | F |

En esta sección nos concentraremos únicamente en la estructura y en la resolución mecánica de una tabla de verdad, más no procederemos a su análisis exhaustivo, concepto que abordaremos en el Capítulo 6.

Sin embargo, mencionaremos tres escenarios posibles que pueden darse tras completar una tabla de verdad y algunas de sus implicancias.

Si en algunas valuaciones, la expresión final resulta **VERDADERO** y en algunas resulta **FALSO**, decimos que estamos frente a una **contingencia**.

También puede darse el caso en el que no haya ninguna valuación que haga verdadera la expresión, es decir, que para toda valuación, el resultado de la expresión es **FALSO**. Ante esta situación decimos que se trata de una **contradicción**.

El caso contrario sería que no haya casos falsos, es decir, que para toda valuación, la expresión sea **VERDADERO**. Esto se conoce con el nombre de **tautología**.

Las tautologías y las contradicciones son interesantes de analizar, ya que son lo que permite formular leyes lógicas. Veremos en el próximo capítulo como usar este conocimiento para tal fin.

5.8 Resumen

En este capítulo intentamos abordar la lógica desde una perspectiva intuitiva al lector, comprendiendo no solo la definición e historia de esta ciencia, así como su objeto de estudio, sino algunos de los diversos elementos transversales a la disciplina. Sin embargo, no hemos establecido ningún sistema formal particular.

Incluso así, la cantidad de conceptos involucrados en esta introducción han sido varios, por lo que intentaremos proporcionar un breve resumen al lector de lo expuesto.

La **lógica** es una ciencia formal que estudia la **deducción** en los **razonamientos**, es decir, los procesos de pensamiento del cerebro humano que llevan a elaborar **conclusiones** a partir de una o más **premisas**. Por ser una ciencia formal, a la lógica no le interesarán los elementos que constituyen el razonamiento en sí, sino la **forma** que tienen dichos razonamientos.

La lógica basa su análisis de **validez** de un razonamiento mediante la operatoria con **valores de verdad** (binarios y dicotómicos, **verdadero** y **falso**). En todo sistema formal de la lógica habrá elementos que son portadores de valor de verdad (encaramos la idea pensando en preguntas que se responden por “*sí*” o “*no*”).

Los distintos elementos portadores de valor de verdad pueden ser elementos base, es decir estar solos, o formar parte de expresiones más complejas, mediante el uso de **conectivas**. Una **conectiva** es un elemento fundamental de la lógica, que permite unir elementos portadores de verdad, para elaborar un nuevo elemento, más complejo, y cuyo valor semántico depende tanto de la conectiva como de los elementos constituyentes. Pueden ser **unarias** (como la **negación**) o **binarias** (como la

conjunción y la disyunción).

Cuando en una expresión hay más de una conectiva, se pueden utilizar paréntesis para desambiguar la **precedencia**. Además hay un orden de precedencia implícito para cuando no se colocan paréntesis (negación, conjunción y luego disyunción). La precedencia permite entender sin ambigüedad el significado semántico de una expresión con múltiples conectivas, indicando qué elemento debe ser “resuelto” primero, y cuál después.

Para analizar una expresión compleja podemos elaborar una **tabla de verdad**, que nos permite visualizar fácilmente todos los elementos involucrados, así como cada posible **valuación**. La tabla de verdad se puede completar de forma mecánica, aunque su interpretación y análisis son más complejos, por lo que se dejarán para el próximo capítulo.

Según el resultado de la expresión en todas las valuaciones podemos clasificarlo como: una **tautología** (todas verdaderas), una **contradicción** (todas falsas) o una **contingencia** (algunas verdaderas, algunas falsas).

Si se tiene una valuación concreta, o se desea analizar solo un caso particular, puede realizarse en su lugar una resolución semántica paso a paso.

Además vimos una de las utilidades de la lógica en las ciencias de la computación, la capacidad de permitir **elaborar preguntas a partir de otras más simples**.

En el Capítulo 6 veremos una primera definición formal de la lógica, basada en los principios de Aristóteles, y profundizaremos sobre varios de los elementos aquí trabajados, definiendo claramente quienes son los elementos portadores de valor de verdad, ahondando en nuevas conectivas, realizando el pasaje del lenguaje natural al lenguaje formal y viceversa, y estableciendo formas de determinar si un razonamiento es o no válido mediante el análisis de las tablas de verdad.

Luego veremos los límites de la lógica proposicional con los que se encontraron los matemáticos de la Edad Moderna. Esto dará lugar a, ya en el Capítulo 7, a trabajar la lógica de predicados, que resuelve algunos de los problemas de la lógica proposicional, y permite formalizar razonamientos que no eran posibles de formalizar con el sistema anterior. Veremos allí sobre la forma de mencionar elementos, sus propiedades y relaciones, y maneras de hablar sobre todos los elementos del universo, o sobre algún elemento en particular, así como hacer aseveraciones sobre ellos.

5.9 Actividades

Ejercicio 5.1 Dados los siguientes razonamientos, indique si se tratan de razonamientos deductivos o inductivos.

- Dado que las versiones más nuevas de los navegadores soportan HTML5, y dado que el Microsoft Edge es la última versión del navegador de dicha empresa, se sigue que el Microsoft Edge soporta HTML5.
- Mi primo compró una computadora aquí y tenía instalado Windows 10. Mi hermano compró una computadora aquí y tenía instalado Windows 10. Mi tío compró una computadora aquí y tenía instalado Windows 10. Si compro una computadora aquí tendrá instalado Windows 10.
- Maria se inscribió el jueves y solo tenía habilitadas las materias del CI. Pablo se inscribió el jueves y solo tenía habilitadas las materias del CI. Juan se inscribió el jueves y solo tenía habilitadas las materias del CI. Por lo tanto, todo alumno que se inscriba el jueves tiene solo habilitadas las materias del CI.



Ejercicio 5.2 Considerando las siguientes preguntas como básicas:

- ¿hay harina?
- ¿hay manteca?
- ¿hay aceite?
- ¿hay agua?
- ¿hay huevos?
- ¿hay yerba?
- ¿hay chocolate?
- ¿hay azúcar?

Se le pide que, utilizando las mencionadas como preguntas básicas^a y las conectivas lógicas vistas, exprese las preguntas generales a continuación^b:

- a) **¿hay para hacer una torta?** (Una torta requiere harina, huevos y manteca)
- b) **¿hay para hacer huevos fritos?** (Requiere huevos y aceite)
- c) **¿hay para hacer huevos duros?** (Requiere huevos y agua)
- d) **¿Puedo almorzar huevos?** (Ya sean duros o fritos)
- e) **¿hay para hacer una torta de chocolate?** (Idéntico a una torta, más chocolate)
- f) **¿Solo se puede tomar mate amargo?** (Cuando se puede tomar mate, es decir, hay agua y yerba, pero no hay azúcar)
- g) **¿No hay nada para el mate?** (Cuando se puede tomar mate, pero no hay torta de ningún tipo)

^aRecuerde que puede elaborar preguntas auxiliares que le ayuden a solucionar las pedidas

^bSe espera escriba algo de la forma “¿hay manzanas? \wedge ¿hay bananas?”

Ejercicio 5.3 Considerando las siguientes preguntas básicas y sus respuestas asociadas:

- ¿Plutón es un planeta? **FALSO**
- ¿Marte es un planeta? **VERDADERO**
- ¿Marte es un satélite? **FALSO**
- ¿Deimos es un satélite? **VERDADERO**
- ¿Ganímedes es un satélite? **VERDADERO**
- ¿Eros es un planeta? **FALSO**
- ¿Eros es un satélite? **FALSO**

Se le pide que, utilizando las mencionadas como preguntas básicas y las conectivas lógicas vistas, exprese las preguntas generales a continuación y determine el valor de verdad de cada una.

- a) **¿Son Marte y Plutón planetas?**
- b) **¿Es Marte un planeta o es Plutón un planeta?**
- c) **¿Es Marte un planeta o un satélite?**
- d) **¿Es cierto que Marte es un planeta y Plutón no lo es?**
- e) **¿Es cierto que Ganímedes y Eros son satélites?**
- f) **¿Es cierto que Eros no es un satélite, pero Deimos si lo es?**
- g) **¿Es Deimos un satélito o es cierto que Marte es un planeta?**
- h) **¿Es Eros un satélite o es cierto que es un planeta?**

Ejercicio 5.4 Considerando las siguientes preguntas como básicas:

- ¿hay ejercito enemigo al Norte?
- ¿hay ejercito enemigo al Este?
- ¿hay ejercito enemigo al Sur?
- ¿hay ejercito enemigo al Oeste?
- ¿hay ejercito aliado al Norte?
- ¿hay ejercito aliado al Este?
- ¿hay ejercito aliado al Sur?
- ¿hay ejercito aliado al Oeste?

Se le pide que, utilizando las mencionadas como preguntas básicas^a y las conectivas lógicas vistas, exprese las preguntas generales a continuación^b:

- ¿Se está amenazado?** (Cuando hay ejercito enemigo en alguna dirección)
- ¿Se está libre de peligro?** (Cuando no hay ejercitos enemigos en ninguna dirección)
- ¿Se tiene apoyo?** (Cuando hay un ejercito aliado en alguna dirección)
- ¿Se está hasta las manos?** (Cuando no hay apoyo y se está amenazado)
- ¿Se puede neutralizar alguna amenaza?** (Cuando hay un ejercito enemigo en alguna dirección, pero también hay un ejercito aliado allí)
- ¿Se puede neutralizar todas las amenazas?** (Cuando se puede neutralizar en todas las direcciones)

^aRecuerde que puede elaborar preguntas auxiliares que le ayuden a solucionar las pedidas

^bSe espera escriba algo de la forma “*¿hay manzanas? \wedge ¿hay bananas?*”

Ejercicio 5.5

Considerando las siguientes preguntas simples:

- ¿Se cumple P?
- ¿Se cumple Q?
- ¿Se cumple R?

Se le pide que analice que valuaciones darán verdadero y cuales falso para las siguientes preguntas compuestas^a. Además se le pide determine cuales de ellas se tratan de tautologías, cuales de contradicciones y cuáles de contingencias.

- ¿Se cumple P? \wedge \neg ¿Se cumple P?**
- ¿Se cumple P? \vee \neg ¿Se cumple P?**
- ¿Se cumple P? \wedge ¿Se cumple Q?**
- ¿Se cumple P? \wedge (¿Se cumple Q? \vee ¿Se cumple R?)**
- ¿Se cumple P? \vee ¿Se cumple Q?**
- \neg ¿Se cumple P? \vee (¿Se cumple Q? \wedge ¿Se cumple R?)**
- \neg ¿Se cumple P? \vee \neg (¿Se cumple Q? \vee ¿Se cumple R?)**

^aSi pensar en preguntas como “*¿Se cumple P?*” le resulta confuso, piense en reemplazarlas por preguntas que le sean más familiares, como “*¿Se cumple que el lobo vive en el bosque?*”. Lo interesante del ejercicio es que la pregunta en cuestión no es realmente relevante, sino la forma de la pregunta compuesta.

Ejercicio 5.6

Sabiendo que las siguientes preguntas compuestas evalúan todas a **VERDADERO**

- ¿La palmera está creciendo torcida? y ¿Es cierto que el árbol no dio paltas este año?
- ¿Es cierto que no hay flores en el cantero? o ¿El árbol dio paltas este año?

Se pide que responda las siguientes preguntas simples

- ¿El árbol dio paltas este año?
- ¿Hay flores en el cantero?
- ¿La palmera está creciendo torcida?

Ejercicio 5.7 Dadas las siguientes preguntas base y sus respuestas:

- ¿Es Microsoft el creador de Windows? **VERDADERO**
- ¿Es Microsoft el creador de GNU? **FALSO**
- ¿Es Richard Stallman el creador de Windows? **FALSO**
- ¿Es Richard Stallman el creador de GNU? **VERDADERO**

Analice las preguntas a continuación, detecte las conectivas involucradas y reemplantee las preguntas para que estén elaboradas en términos de las preguntas dadas. Luego, analice el valor de verdad de las mismas.

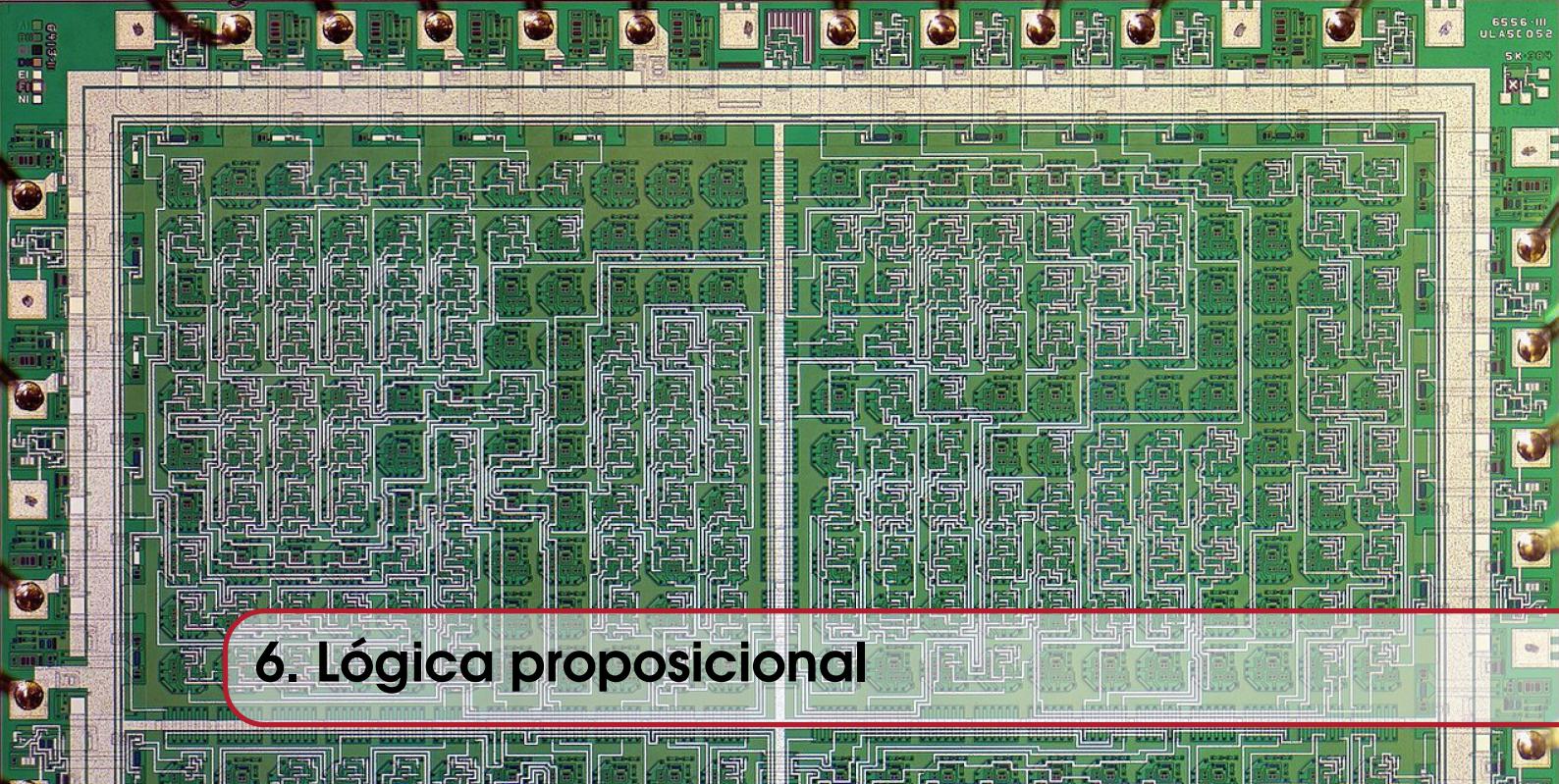
- a) ¿Es cierto que no es cierto que Microsoft es el creador de GNU?
- b) ¿Es cierto que Richard Stallman es el creador de Windows o de GNU?
- c) ¿Es cierto que Microsoft no es el creador de GNU pero que Richard Stallman si lo es?
- d) ¿Es cierto que GNU no es del mismo creador que Windows?

Ejercicio 5.8 Se ha encontrado vida en otro planeta, y se ha decidido nombrar a los animales encontrados como “Woofle”, “Brlfks” y “Morlock”.

Cada animal tiene sus características distintivas (pueden ser grandes o chicos, con o sin pelo, acuáticos o terrestres).

Si las respuestas a todas las preguntas siguientes son **VERDADERO**, enumere qué características tiene cada animal. Para ello, realice la tabla de verdad de cada pregunta y analice las respuestas a las preguntas base en las valuaciones verdaderas.

- ¿El Woofle tiene pelo y no es acuático?
- ¿El Morlock es terrestre y es grande, y el Brlfks es acuático?
- ¿El Brlfks es pequeño o el Morlock es pequeño?
- ¿El Woofle no es grande o el Brlfks es grande? ¿Y es cierto que el Morlock no es pequeño?
- ¿El Woofle tiene pelo y el Brlfks no? ¿O es cierto que el Brlfks no tiene pelo y el Morlock tiene pelo?



6. Lógica proposicional

Circuito de un Uncommitted Logic Array de Ferranti.
Fotografía de Richard Evans.

La **lógica proposicional**, también llamada **lógica de orden cero**, es el sistema de lógica inicialmente trabajado por Aristóteles, y es el más simple de entender y trabajar, aunque hay límites de esta que no permiten representar todos los razonamientos posibles. En este sistema, el elemento portador de verdad es la **proposición**, una oración informativa que versa sobre algo, y de la que se puede decir que lo que afirma es **verdadero** o **falso**.

Por ejemplo, “*En este momento está lloviendo en Buenos Aires*” es una proposición. Su valor de verdad concreto dependerá del día y la hora en que el lector lea estas líneas, pero seguro, o bien se cumplirá, o bien no se cumplirá.

Notar que el lenguaje natural está lleno de otros elementos que no son proposiciones, como las oraciones exclamativas (“*¡Estoy enojado!*”), las imperativas (“*Preparame un jugo de frutas*”) o las interrogativas (“*¿Hay leche en la heladera?*”), entre otras formas de uso del lenguaje. No es el objetivo del presente enfocarnos en las formas de uso del lenguaje, sino simplemente entender el concepto de proposición. Así una oración debe **afirmar “algo”** para ser una proposición. A este tipo de oraciones se las llama informativas, o a veces enunciativas, afirmativas o descriptivas.

6.1 Proposiciones, atómicas y compuestas

El elemento más pequeño que constituye la lógica proposicional es precisamente la proposición. Veamos una definición formal:

Definición 6.1 Una **proposición** es una entidad portadora de valor de verdad. En la lógica proposicional es el elemento base.

Pensemos en la oración “*La Tierra es plana*”. Esta oración es una proposición que, según el contexto científico actual, es claramente **falsa**. Sin embargo hay movimientos de personas que reivindican el terraplanismo, y afirman que la proposición es **verdadera**. Más aún, si vivieramos

hace 500, 1000 o 2000 años, sería una afirmación considerada generalmente como verdadera, y se miraría raro a quien dijera que es falsa (incluso en algunas épocas, varias personas serían asesinadas por afirmar semejantes cosas).

Así, cualquier **oración informativa**, por inverosímil que parezca, por ridícula que suene la afirmación, es una proposición, y su **valor de verdad**, puede (y debe) ser analizado.

Pensemos ahora en la siguiente oración “*La Tierra es redonda y el Sol gira alrededor de La Tierra*”. Por supuesto, esta también es una proposición, y su valor de verdad dependerá del contexto en el que se analice, pero esta proposición es diferente a la anterior. Gracias a los avances científicos probamos que la Tierra es redonda, pero que no es cierto que el Sol gire entorno a esta. La proposición sería entonces falsa, ya que si bien la primer parte de lo que asevera es cierto, lo segundo no lo es. Sin embargo, si vivieramos previo a la aceptación de la Revolución de Copérnico, ya sabríamos que la Tierra es redonda (pues se navegaba el globo), pero creeríamos que el Sol gira en torno a la Tierra, y no al revés, por tanto asumiendo la proposición completa como verdadera.

Lo interesante de dicha proposición es precisamente que hay **dos partes** en la misma. Por un lado “*La Tierra es redonda*” y por otro “*El Sol gira alrededor de la Tierra*”. El **valor de verdad** de la proposición toda dependerá del valor de verdad de cada una de dichas partes, y claro, de la **conectiva** que hay entre ellas, en este caso “y”. Así, se espera que ambas partes sean verdaderas para ser verdadera la proposición toda.

Decimos entonces que hay proposiciones que son **compuestas**, es decir, **están formadas por otras proposiciones y conectivas**. El valor de verdad de una proposición compuesta dependerá de las partes que la componen y de la conectiva que las asocia. Por otro lado, hay proposiciones que son **simples**, es decir, **no pueden dividirse más** (no hay conectivas), y su valor de verdad depende de sí mismas, una vez que son contrastadas con la realidad.

Definición 6.2 Una **proposición simple** (o atómica) es una proposición cuyo valor de verdad depende solo de si misma y no puede dividirse.

Definición 6.3 Una **proposición compuesta** (o molecular) es una proposición que está formada por una o dos proposiciones unidas por una conectiva, y cuyo valor de verdad depende de sus elementos constituyentes así como de la semántica de la conectiva que los une.

Teniendo la idea de proposición, podemos comenzar a analizar proposiciones complejas e incluso razonamientos.

6.2 Pasaje de texto a formulas

Hasta ahora hemos obviado uno de los elementos más importantes de la lógica, cómo pasar de un razonamiento expresado en lenguaje natural a uno expresado en lenguaje formal, es decir, a una **fórmula lógica**. Esto es clave para poder realizar el análisis posterior de la forma de dichos razonamientos y evaluar su validez.

6.2.1 Pasaje a fórmulas de proposiciones atómicas

Previo a trabajar con razonamientos completos, trabajaremos con proposiciones, y veremos como plantear una fórmula para estas. En el proceso veremos algunos conceptos importantes de la

lógica proposicional.

Comencemos por analizar un ejemplo sencillo, la proposición “*Hoy está lloviendo en la Ciudad de Buenos Aires*”. Si miramos la proposición detenidamente no encontraremos ninguna palabra que refiera a una conectiva de ningún tipo (“y”, “además”, etc. para conjunción, “o”, “o bien”, etc. para disyunción, o “no”, “no es cierto que” para negación, entre otras). Esto quiere decir que la proposición es **atómica**, o sea, es una única proposición en sí misma, y no hay partes más pequeñas que la compongan.

Una proposición atómica se formaliza asignandole una letra minúscula a la misma, para definir lo que se conoce como el **diccionario**. El diccionario asocia los elementos que aparecen en la fórmula con una proposición del lenguaje natural. La fórmula en este caso implica solo dicha letra.

■ Ejemplo 6.1

Expresión en lenguaje natural:

Hoy está lloviendo en la Ciudad de Buenos Aires

Diccionario:

p = Hoy está lloviendo en la Ciudad de Buenos Aires

Fórmula:

p

■

Es costumbre utilizar la letra “ p ” (por proposición) para la primera de las proposiciones que forman el diccionario, siguiendo luego con “ q ” para la segunda, “ r ” para la tercera, y las restantes letras del abecedario para las sucesivas. Cada letra se conoce como **variable proposicional**.

Definición 6.4 Una **variable proposicional** es una variable discreta que puede ser **verdadera** o **falsa**. Es el elemento básico de construcción de fórmulas proposicionales.

La elección de la letra a utilizar es completamente arbitraria, por lo que podríamos haber elegido otra cualquiera, por ejemplo, la “ l ” (por la idea de que está “lloviendo”). Esto daría lugar a un diccionario y formula distintos:

■ Ejemplo 6.2

Expresión en lenguaje natural:

Hoy está lloviendo en la Ciudad de Buenos Aires

Diccionario:

l = Hoy está lloviendo en la Ciudad de Buenos Aires

Fórmula:

l

■

Sin embargo, para el estudio puntual de la lógica, la letra elegida es irrelevante. Así, ambas fórmulas, consideradas de forma independiente, se consideran equivalentes.

6.2.2 Pasaje a fórmulas de proposiciones moleculares

Vayamos ahora con algo ligeramente más complejo, la proposición “*La tierra es redonda y no es el centro del universo*”. En este caso, podemos encontrar en el texto la palabra “y”, que indica una idea semántica clara, la conjunción. Así, vemos que esta proposición es **compuesta**. Si

dividimos entonces esta proposición en dos partes en el lugar donde encontramos la conjunción, podemos volver a realizar el análisis de cada parte por separado. Así tenemos por un lado “*La tierra es redonda*”, una proposición atómica, por lo que corresponderá asignarle una única letra; y por otro lado tenemos “*no es el centro del universo*”, que debe analizarse más cuidadosamente. En primer lugar la oración es negativa, es decir, hay un “*no*” en la misma. Ese “*no*” es una conectiva de negación¹. Por otro lado, la oración por si misma carece de sujeto, y para comprender bien de qué habla debemos encontrarlo. La frase “*no es el centro del universo*” se refiere a la Tierra, este es el sujeto tácito en nuestra oración.

Si volvemos a expresar nuestra oración usando conectivas, y siendo explícitos con el sujeto de cada oración, tendríamos algo como “*La tierra es redonda $\wedge \neg$ La Tierra es el centro del universo*”. Por precedencia, la negación asocia antes que la conjunción, por lo que no necesitamos paréntesis para desambiguar, ya que actúa la precedencia, de la cual charlamos en la Sección 5.6.

Tenemos entonces dos proposiciones atómicas distintas que componen la expresión a pasar a fórmula. Debemos asignar una letra a cada una de esas proposiciones para elaborar el diccionario. Vamos a elegir las letras “*p*” y “*q*” en este caso, pero como dijimos, podrían haber sido cualquier otro par de letras). La fórmula consiste en tomar la oración ya analizada y con las conectivas en su lugar, y reemplazar las proposiciones en lenguaje natural por las letras asignadas.

■ Ejemplo 6.3

Expresión en lenguaje natural:

La tierra es redonda y no es el centro del universo

Proposición luego de determinar conectivas:

La tierra es redonda $\wedge \neg$ La Tierra es el centro del universo

Diccionario:

p = La tierra es redonda

q = La Tierra es el centro del universo

Fórmula:

$p \wedge \neg q$

■

Por supuesto, distintas oraciones podrán tener distintas conectivas, algunas bastante explícitas, otras tal vez no tanto. Veamos un último ejemplo con la frase “*El gato en la caja está vivo o muerto*”. La primer reacción del lector puede implicar pensar que el “*o*” indica una disyunción simple, y que hay dos proposiciones atómicas, de la siguiente manera:

■ Ejemplo 6.4

Expresión en lenguaje natural:

El gato en la caja está vivo o muerto

Proposición luego de determinar conectivas:

El gato en la caja está vivo \vee El gato en la caja está muerto

Diccionario:

p = El gato en la caja está vivo

¹Las proposiciones atómicas siempre van a estar expresadas de forma positiva. Por ejemplo, si nos encontramos con algo como: “*el perro no estará feliz*”, debemos pensar el caso positivo, “*el perro estará feliz*” y aplicar la conectiva de negación. No solo la palabra no puede indicar negación, ya que una frase como “*el perro estará infeliz*” podría ser interpretada también como negación.

$q = \text{El gato en la caja está muerto}$

Fórmula:

$$p \vee q$$

Pero esto no es del todo correcto. Cuando hablamos de “*el gato en la caja*” en ambas proposiciones atómicas, nos referimos en ambas oportunidades al mismo gato. Un gato no puede estar vivo y muerto al mismo tiempo², por lo que no pueden ocurrir ambas al mismo tiempo, y la idea de estar muerto es lo mismo que la de no estar vivo, por lo que en realidad la oración tiene una negación implícita. Una forma más adecuada de representar esto sería:

■ **Ejemplo 6.5**

Expresión en lenguaje natural:

El gato en la caja está vivo o muerto

Proposición luego de determinar conectivas:

El gato en la caja está vivo $\vee \neg$ El gato en la caja está vivo

Diccionario:

$p = \text{El gato en la caja está vivo}$

Fórmula:

$$p \vee \neg p$$

■

6.3 Razonamientos

Los **razonamientos**, en la lógica proposicional, **están formados de varias proposiciones**. Algunas de estas actúan como **premisas**, y una en particular es la **conclusión**. En ese sentido, vamos a encontrar razonamientos compuestos de múltiples oraciones, y el trabajo complejo consiste en identificar cuáles de ellas son las premisas y cuáles la conclusión. Luego, simplemente se debe pasar a fórmula cada una de esas oraciones, con un detalle adicional a considerar con respecto a lo expuesto previamente, el diccionario unificado. Veamos un ejemplo concreto, tenemos el siguiente razonamiento:

“*El gato en la caja está o bien vivo o bien muerto. Pero el gato no estaba muerto. Por lo tanto, el gato estaba vivo.*”.

En este razonamiento, hay dos premisas y una conclusión. Podemos encontrar fácilmente las tres oraciones que componen el razonamiento mirando los signos de puntuación, ya que el punto seguido está delimitando las oraciones de forma clara. Sin embargo, nos resta saber dónde está la conclusión, es decir, cual de las tres oraciones es la conclusión. Para ello, en el lenguaje natural existen palabras o frases que dan lugar al interlocutor a comprender cuándo se está por empezar a hablar de la conclusión. Llamamos a estos elementos **indicadores de conclusión**.

Definición 6.5 Un **indicador de conclusión** es una palabra o frase que permite separar las premisas de la conclusión en un razonamiento expresado en lenguaje natural. Ante la presencia de un indicador de conclusión, las premisas se encontrarán en las oraciones previas al indicador,

²Para quienes sean conocedores de la paradoja planteada por Schrödinger para explicar los estados cuánticos, acá estamos trabajando con una idea puramente clásica, o dicho de otra forma, sobre el estado del gato luego de mirar dentro de la caja.

y la conclusión en la oración inmediatamente posterior al mismo.

El indicador de conclusión en este ejemplo es “*Por lo tanto*”, en la última oración. Eso nos indica entonces que lo previamente expuesto son premisas, y lo que sigue es la conclusión. Podemos entonces decir que:

■ Ejemplo 6.6

Expresión en lenguaje natural:

El gato en la caja está o bien vivo o bien muerto. Pero el gato no estaba muerto. Por lo tanto, el gato estaba vivo.

Indicador de conclusión:

Por lo tanto

Premisas:

El gato en la caja está o bien vivo o bien muerto.

El gato no estaba muerto.

Conclusión:

El gato estaba vivo. ■

Notar que al expresar la conclusión, tomamos solo la parte que sigue al indicador de conclusión, sin incluirlo. Adicionalmente se puede apreciar que al expresar las premisas obviamos la palabra “*Pero*” para la segunda de ellas. Esa palabra actúa como conector entre ambas premisas, y, de momento, no tendrá relevancia en el análisis.

Para expresar un razonamiento en texto dejando claras premisas y conclusión, se suele usar una sintaxis que implica exponer las proposiciones detectadas, a razón de una por renglón, donde la última debe ser la conclusión, y debe estar separada del resto mediante una línea horizontal. Para el ejemplo anterior, podríamos expresar el razonamiento como:

■ Ejemplo 6.7

Expresión en lenguaje natural:

El gato en la caja está o bien vivo o bien muerto. Pero el gato no estaba muerto.

Por lo tanto, el gato estaba vivo.

Indicador de conclusión:

Por lo tanto

Razonamiento:

El gato en la caja está o bien vivo o bien muerto. ■

El gato no estaba muerto.

El gato estaba vivo.

Pero si bien hemos identificado las premisas y la conclusión, aún no hemos encontrado una fórmula para el razonamiento. La idea es encontrar ahora, en cada una de las proposiciones encontradas, cuáles son las conectivas involucradas. Ya hemos visto como encontrar las conectivas y expresar la idea del gato en la caja, por lo que contaremos con lo siguiente.

■ Ejemplo 6.8

Razonamiento con conectivas:

$\begin{array}{l} \text{El gato en la caja está vivo} \vee \neg \text{El gato en la caja} \\ \text{está vivo} \\ \neg \neg \text{El gato en la caja está vivo.} \end{array}$

El gato en la caja está vivo.

Notar como hemos cambiado la idea del “*gato muerto*” por la negación del “*gato vivo*”, como ya hemos mencionado previamente, lo cual implica que en la segunda premisa, hay una doble negación.

Para pasar a fórmula debemos elaborar un diccionario, pero no uno para cada proposición encontrada, sino un diccionario único para todas las premisas detectadas. En el ejemplo anterior tenemos una única premisa atómica, que aparece tanto en las premisas como en la conclusión, por lo que habrá una única variable proposicional. Luego, se debe reemplazar la aparición de las proposiciones atómicas en el texto con la variable proposicional.

■ **Ejemplo 6.9**

Diccionario:

$p = \text{El gato en la caja está vivo}$

Fórmula del razonamiento:

$$\begin{array}{c} p \vee \neg p \\ \neg \neg p \\ \hline p \end{array}$$

Al expresar en fórmula un razonamiento, es innecesario (y hasta incomodo) expresar el razonamiento en múltiples renglones. Para expresar el razonamiento en una única línea debemos escribir las premisas, separadas por coma, y luego un símbolo especial, entre estas y la conclusión, llamado **secuente** (\vdash). El razonamiento anterior se podría expresar entonces como:

■ **Ejemplo 6.10**

$$p \vee \neg p, \neg \neg p \vdash p$$

Así, podríamos decir que la fórmula completa para el razonamiento, incluyendo el diccionario quedaría de la siguiente forma:

■ **Ejemplo 6.11**

Diccionario:

$p = \text{El gato en la caja está vivo}$

Fórmula:

$$p \vee \neg p, \neg \neg p \vdash p$$

Veremos ahora un segundo ejemplo con el razonamiento “*La Tierra es redonda, ya que o bien es plana o bien es redonda, y sabemos que la Tierra no es plana.*”. Acá no hay puntos que separan las oraciones, sino que están separadas por comas. No solo eso, sino que no vamos a encontrar un indicador de conclusión para determinar dónde está la conclusión. Lo que encontraremos en su lugar será un **indicador de premisa**.

Definición 6.6 Un **indicador de premisa** es una palabra o frase que permite separar las premisas de la conclusión en un razonamiento expresado en lenguaje natural. Ante la presencia de un indicador de premisa, las premisas se encontrarán en las oraciones posteriores al indicador, y la conclusión en la oración previa al mismo.

El indicador de premisa marca que la conclusión está al comienzo, y las premisas se encuentran en cambio al final del razonamiento, dado por la frase “ya que”.

■ **Ejemplo 6.12**

Expresión en lenguaje natural:

La Tierra es redonda, ya que o bien es plana o bien es redonda, y sabemos que la Tierra no es plana.

Indicador de premisa:

Ya que

Razonamiento:

*O bien la Tierra es plana o bien es redonda.
La Tierra no es plana.*

La Tierra es redonda.

Nuevamente analizaremos cada una de las proposiciones detectadas para elaborar un único diccionario y así determinar la fórmula. Mostraremos esto en un único paso en el siguiente ejemplo:

■ **Ejemplo 6.13**

Razonamiento:

*O bien la Tierra es plana o bien es redonda.
La Tierra no es plana.*

La Tierra es redonda.

Razonamiento con conectivas:

*La Tierra es plana \vee La Tierra es redonda.
 \neg La Tierra es plana.*

La Tierra es redonda.

Diccionario:

$p = \text{La Tierra es plana}$ $r = \text{La Tierra es redonda}$

Fórmula:

$$\begin{array}{c} p \vee r \\ \neg p \\ \hline r \end{array}$$

Fórmula en una línea:

$$p \vee r, \neg p \vdash r$$

Como detalle adicional, vale la pena mencionar que en muchas ocasiones es necesario reformular

lar las oraciones al momento de elaborar el diccionario de un razonamiento, pues la misma idea puede presentarse en distintos tiempos verbales, o con diferentes estructuras gramaticales. Hay que ignorar esos detalles y buscar una oración que represente la idea semántica para elaborar el diccionario.

Por supuesto pueden haber razonamiento en donde solamente un indicador de premisa, otros donde haya solo un indicador de conclusión, o incluso casos en donde haya ambos. Las palabras mencionadas en estos ejemplos no son las únicas posibles de utilizar como indicador de premisa o de conclusión, ya que tenemos muchas otras.

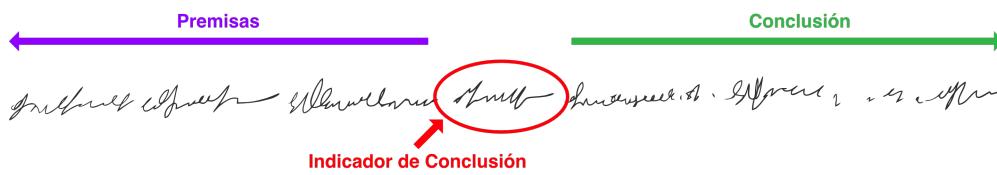
6.3.1 Indicadores de conclusión:

Existen varias palabras que pueden actuar como indicadores de conclusión. A continuación se presenta una breve lista de estos.

- Por lo tanto
- En consecuencia
- Se concluye que
- Se deduce
- Es por ello que
- Por ende
- Luego
- Entonces
- Por lo cual
- De esto se desprende que

La mayoría de estas palabras o frases dan la idea de secuencia, es decir que algo sucede porque otras cosas sucedieron previamente, o de cuasalidad, que algo se cumple solo si otras cosas se cumplen previamente.

Recordemos que un indicador de premisa separa las premisas de la conclusión de forma tal que las premisas estarán **antes**, y la conclusión estará **después**, del indicador, como se presenta en el siguiente gráfico:



6.3.2 Indicadores de premisa:

También existen varias palabras que pueden actuar como indicadores de premisa. A continuación se presenta una breve lista de estos.

- Dado que
- Ya que
- Esto es así porque
- Porque
- Esto se sigue de
- En vista de que
- Pues

Acá la idea es la inversa, es decir, que hay cosas que se cumplieron ya que otras se habían cumplido antes, o de que algo ocurrió debido a que otras cosas ocurrieron.

Con esto ya tenemos una idea más clara de cómo los razonamientos son interpretados por la lógica proposicional, aunque los razonamientos que podemos procesar son acotados, porque de momento conocemos solo tres conectivas.

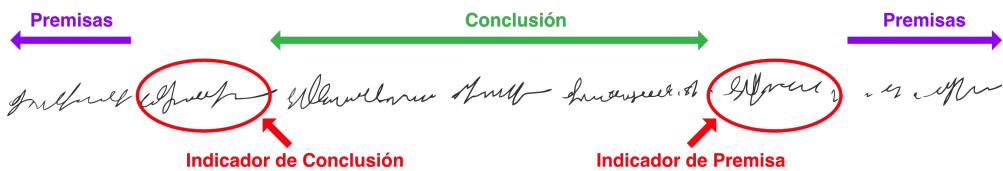
En la sección siguiente veremos otra serie de conectivas que nos ayudarán a plantear más y mejores razonamientos, y a posteriori veremos como los razonamientos pueden ser analizados para determinar su validez.

Recordemos que un indicador de premisa separa las premisas de la conclusión de forma tal que

las premisas estarán **antes**, y la conclusión estará **después**, del indicador, como se presenta en el siguiente gráfico:



Adicionalmente, y para finalizar esta sección, podemos mencionar que pueden existir razonamientos en donde hay tanto indicadores de premisas como de conclusión. En este caso, puede darse que la conclusión se encuentre en el medio del razonamiento. El siguiente gráfico muestra esta idea.



6.4 Otras conectivas más complejas

Hasta ahora hemos trabajado únicamente con tres conectivas, la conjunción, la disyunción y la negación, que son las más comunes de encontrar en el lenguaje natural en el habla cotidiana. Sin embargo, hay varias expresiones semánticas de lenguajes naturales que no pueden expresarse con dichas conectivas, y que son claves para poder escribir razonamientos interesantes.

En esta sección veremos entonces otra serie de conectivas que podemos hallar en el lenguaje natural, su simbología, ejemplos y su tabla semántica.

6.4.1 Disyunción excluyente

Si pensamos en frases como “*El ascensor está detenido en un piso o bien está moviéndose*” tal vez se nos ocurra representar esto mediante una disyunción, algo como “*El ascensor está detenido en un piso* \vee *El ascensor está moviéndose*”. Sin embargo, si consideramos que el ascensor no puede estar en dos lugares al mismo tiempo, entonces ese “*o bien*” no nos habla de una disyunción normal, sino de una en donde puede ocurrir lo primero o lo segundo, pero seguro no pueden suceder ambas cosas al mismo tiempo.

La idea de la **disyunción excluyente** es precisamente esa, la idea de “**uno u otro, pero no ambos**”. En las proposiciones es posible encontrar expresiones como “*Esta noche iré al cine o al teatro, pero no a ambos*”, aunque no es tan común. Muchas veces, como en el ejemplo del ascensor, simplemente encontraremos una palabra que refiere a la idea de disyunción, y es el lector quien debe interpretar ambas partes y comprender que no pueden suceder ambas al mismo tiempo.

Esto vuelve a veces difícil comprender en qué puntos hay disyunciones y en cuales disyunciones excluyentes, en especial a quienes recién comienzan a trabajar la idea de formalizar proposiciones enunciadas en el lenguaje natural. Es importante entonces tomarnos un pequeño tiempo adicional para pensar si las proposiciones unidas por la disyunción son excluyentes o no.

El símbolo que utilizamos para esta conectiva es $\vee\!\!\!/\!$, y su tabla semántica es la siguiente:

| Φ | Ψ | $\Phi \vee \Psi$ |
|--------|--------|------------------|
| V | V | F |
| V | F | V |
| F | V | V |
| F | F | F |

Es decir, la disyunción excluyente indica que una expresión es verdadera solo cuando se cumple una de las dos partes que la componen, pero no cuando se cumplen ambas.

6.4.2 Implicación

La **implicación** (también llamada **condicional**) es la conectiva que se utiliza para la idea que una condición sucede solo si otra a sucedido. Es decir, la idea de secuencia lógica, de que una cosa sucede de otra, en consecuencia de la primera. Un ejemplo simple sería “*Si hay fuego, entonces habrá humo*”. Es decir, el hecho de que haya humo es una consecuencia de que haya fuego.

En términos semánticos, esta frase se debe considerar verdadera cuando, si contrastamos con la realidad, efectivamente hay fuego y hay humo, lo cual sería esperable, y falso si hubiera fuego pero no ocurre que haya humo. Si no hay fuego, y no hay humo tampoco, entonces la frase también debe considerarse verdadera. El caso que suele confundir a la mayoría es que la frase debe ser considerada verdadera aún cuando no hay fuego, pero efectivamente hay humo. Si lo pensamos, la frase habla de lo que ocurrirá cuando haya fuego, pero no dice nada de lo que sucede si no hay fuego. Así, puede ser que no haya fuego y aún así exista humo, producto de otra cosa, por ejemplo, una reacción química puede producir humo sin que haya fuego involucrado.

El símbolo usado para la implicación es \rightarrow y su tabla semántica sigue lo recién mencionado:

| Φ | Ψ | $\Phi \rightarrow \Psi$ |
|--------|--------|-------------------------|
| V | V | V |
| V | F | F |
| F | V | V |
| F | F | V |

Notar que la idea de implicación está muy relacionada a la idea de razonamiento, ya que el indicador de conclusión también tiene una idea de secuencia. De hecho, veremos que esta conectiva y esos indicadores de conclusión están intimamente relacionados. Por supuesto, el problema surge al intentar identificar conectivas en un razonamiento, donde las palabras que se utilizan para la conectiva de implicación son similares a las que se utilizan como indicador de conclusión. Algo a tener en cuenta es que el indicador de conclusión va a estar siempre al comienzo de una oración (la cual es la conclusión) mientras que las conectivas suelen estar a la mitad de una oración.

6.4.3 Equivalencia

La última conectiva que analizaremos será la conectiva de **equivalencia** (también llamada **doble implicación** o **bicondicional**). Esta conectiva deriva de la idea de “*si y solo si*”, es decir, que algo ocurre solo cuando otra cosa ocurre al mismo tiempo, y no ocurre solo cuando la otra tampoco ocurre. Un ejemplo podría ser “*Ganaremos el torneo si y solo si ganamos el próximo partido*”. Es decir, no puede darse que se gane el torneo si no se gana el siguiente partido, y su inversa también se cumple, no se ganará el torneo si no se gana el partido, algo que se da, por ejemplo, en una final.

La semántica tiene además la idea de igualdad en términos lógicos, por ejemplo, la frase “*Sentir la brisa y correr por el campo es lo mismo que correr por el campo y sentir la brisa*”, lo cual habla de la idea de comutatividad en la lógica.

El símbolo de esta conectiva es \leftrightarrow y su tabla semántica es la siguiente:

| Φ | Ψ | $\Phi \leftrightarrow \Psi$ |
|----------|----------|-----------------------------|
| <i>V</i> | <i>V</i> | <i>V</i> |
| <i>V</i> | <i>F</i> | <i>F</i> |
| <i>F</i> | <i>V</i> | <i>F</i> |
| <i>F</i> | <i>F</i> | <i>V</i> |

Este tipo de conectivas se utiliza mucho al momento de realizar demostraciones matemáticas, y se vuelve de vital importancia en la misma. Por ejemplo, en $3 + x = 7 \leftrightarrow x = 4$.

6.5 Formulas bien formadas

Ahora que ya conocemos todas las conectivas, podemos hablar de cuándo una fórmula de la lógica proposicional está **bien formada**. Una formula puede no tener sentido, incluso si se usan los símbolos de las conectivas que hemos visto. Por ejemplo, en matemática, si escribimos algo como $+5 \cdot 9$ nos encontramos ante símbolos matemáticos, pero sin sentido alguno. En la lógica pasa lo mismo, las conectivas deben usarse de forma adecuada para que una fórmula tenga sentido.

Toda variable proposicional es una **fórmula bien formada** (FBF). Es decir, p, q, r , etc. son todas fórmulas bien formadas. Teniendo eso en consideración, y suponiendo que Φ y Ψ son fórmulas bien formadas, todas las siguientes son también fórmulas bien formadas.

- $(\neg\Phi)$
- $(\Phi \wedge \Psi)$
- $(\Phi \vee \Psi)$
- $(\Phi \vee \Psi)$
- $(\Phi \rightarrow \Psi)$
- $(\Phi \leftrightarrow \Psi)$

Notar como todas las fórmulas mencionadas se encuentran entre paréntesis, ya que esto será importante en un momento.

Pensemos ahora un ejemplo puntual. Queremos saber si la siguiente es una fórmula bien formada o no. Para ello, debemos analizar desde adentro hacia afuera para ver si se cumplen las reglas de formación de fórmulas bien formadas.

| | |
|---|---|
| $((p \wedge (q \vee (\neg r))) \wedge (r \vee p))$ | Se quiere probar si esta fórmula es una FBF. |
| 1. p | Es una FBF por ser solo una variable proposicional |
| 2. q | Es una FBF por ser solo una variable proposicional |
| 3. r | Es una FBF por ser solo una variable proposicional |
| 4. $(r \vee q)$ | Los paréntesis externos corresponden a una disyunción por lo que tanto r como q deben ser FBFs. 4. es entonces FBF ya que 2. y 3. prueban ser FBFs |
| 5. $(\neg r)$ | Los paréntesis externos se corresponden a una negación, por lo que r debe ser una FBF, y esto se prueba por 3. |
| 6. $(q \vee (\neg r))$ | Los paréntesis externos corresponden a una disyunción, por lo que tanto q como $(\neg r)$ deben ser FBFs. Esto se prueba por 2. y por 5. |
| 7. $(p \wedge (q \vee (\neg r)))$ | Los paréntesis externos vuelven a corresponder a una conjunción, por lo que tanto p como $(q \vee (\neg r))$ deben ser FBFs. Esto se prueba por 1. y por 6. |
| 8. $((p \wedge (q \vee (\neg r))) \wedge (r \vee p))$ | Los paréntesis más externos corresponden a una conjunción, por lo que para este sea una FBF, tanto $(p \wedge (q \vee (\neg r)))$ como $(r \vee p)$ deben ser FBFs. Esto se prueba por 7. y por 4. |

Los errores más comunes al momento de escribir fórmulas que hacen que la fórmula no esté bien formada suelen estar relacionadas a los siguientes ejemplos:

| | |
|---------------------|--|
| $(p \neg q)$ | Usar la negación como operador binario, en lugar de operador unario. |
| $(\wedge p)$ | Usar un operador binario como un operador unario. |
| $(p \wedge \vee q)$ | Usar dos operadores binarios de forma consecutiva |
| (pq) | Ausencia de operador entre dos variables proposicionales o fórmulas |
| $(p(\wedge q))$ | Aperturas o cierres de paréntesis en lugares incorrectos |

(6.1)

Así, es importante prestar especial atención para no cometer estos errores, y asegurar que la fórmula que estamos escribiendo es una fórmula bien formada.

Sin embargo, cuando empezamos a analizar, los paréntesis de una fórmula bien formada evitarán cualquier ambigüedad en la forma en la que debe interpretarse la misma, pero la gran cantidad de los mismos resulta molesta. Para evitar escribir algunos de ellos, vamos a seguir las siguientes reglas:

- Los paréntesis más externos se omiten, siempre. Ej. la FBF $(p \wedge (q \vee r))$ podemos escribirla como $p \wedge (q \vee r)$.
- La negación no llevará paréntesis externos. Ej. la FBF $(p \wedge (\neg(q \wedge p)))$ podemos escribirla como $p \wedge \neg(q \wedge p)$.
- Varias ocurrencias del mismo operador no llevan paréntesis. Ej. Ej. la FBF $(p \wedge (q \wedge r))$ podemos escribirla como $p \wedge q \wedge r$ y la FBF $(\neg(\neg p))$ se puede escribir como $\neg\neg p$.
- Podemos omitir paréntesis en todo lugar donde la precedencia deje en claro que operación debe analizarse primero que otra. Ej. la FBF $((p \wedge q) \rightarrow r)$ se puede escribir como $p \wedge q \rightarrow r$

El hecho de omitir paréntesis no hace que la fórmula deje de ser bien formada, sino que simplifica la complejidad de lectura de la misma. Ante la duda, es preferible contar con los paréntesis, pero en este libro omitiremos aquellos que no sean necesarios, salvo que los paréntesis

ayuden a explicar un concepto o tema.

6.5.1 Orden de precedencia

El **orden de precedencia** tiene que ver con la prioridad con la que leemos determinados símbolos en una fórmula. Ya habíamos visto la misma en la Sección 5.6, pero ahora debemos definir una nueva precedencia, ya que hay nuevos símbolos. La precedencia que usaremos será:

1. Negación
2. Conjunción
3. Disyunción
4. Implicación
5. Equivalencia

Es decir, que la negación asocia más fuerte, luego la conjunción, la disyunción, la implicación, y por último la equivalencia. Esto permite eliminar paréntesis, ya que los elementos que asocian más fuerte no necesitan paréntesis para indicar que su interpretación debe ser previa a otros elementos. Así $((((\neg p) \wedge q)) \rightarrow r)$ se puede escribir como $\neg p \wedge q \rightarrow r$ ya que la precedencia nos permite simplificar los paréntesis.

6.6 Análisis de validez de razonamientos por tabla de verdad

Ahora que sabemos como detectar un razonamiento e identificar sus partes, vamos a querer poder determinar si el mismo es **válido** o no. Pero, ¿Qué significa que un razonamiento sea válido?

Definición 6.7 Un razonamiento se dice **válido** cuando **las premisas implican a la conclusión**. Es decir, que si sus premisas son verdaderas, entonces la conclusión es necesariamente verdadera.

Es decir, lo que tiene que ocurrir es que si para cada valuación donde todas las premisas son verdaderas, entonces la conclusión en cada una de esas valuaciones también debe ser verdadera.

Analicemos el razonamiento a continuación para poder comprender la idea. Vamos a realizar el pasaje a fórmula del razonamiento junto con la presentación del mismo, pero el lector puede intentar realizar paso a paso la identificación de las partes como se presentó en la sección anterior³.

Ejemplo 6.14

Razonamiento en lenguaje natural:

Si hay fuego entonces habrá humo. Pero no hay humo. Por lo tanto, no había fuego.

Diccionario:

$f =$ Hay fuego

$h =$ Hay humo

Fórmula:

$$f \rightarrow h, \neg h \vdash f$$

Vamos a elaborar ahora la tabla de verdad para este razonamiento. Para un razonamiento, a diferencia de para una simple proposición, donde nos interesa llegar a una única columna final, aquí

³Recuerde que al momento de armar el diccionario debe reformular las proposiciones para simplificar elementos gramaticales y los tiempos verbales.

debemos asegurarnos de tener una columna para cada premisa, y una columna para la conclusión. Por supuesto, puede que sea necesario elaborar columnas intermedias para resolver los diversos pasos, antes de arribar a alguna de las columnas que nos interesan, algo que ocurrirá cuando la fórmula sea compleja .

Comenzaremos entonces por elaborar una tabla donde “ f ” y “ h ” (las variables proposicionales) son los elementos base. En nuestro ejemplo, no necesitaremos columnas intermedias, ya que las fórmulas tienen todas una única conectiva, por lo que iremos directamente a colocar una columna para cada premisa, y una para la conclusión.

| f | h | Premisa 1 $f \rightarrow h$ | Premisa 2 $\neg h$ | Conclusión $\neg f$ |
|-----|-----|--------------------------------|-----------------------|------------------------|
| V | V | | | |
| V | F | | | |
| F | V | | | |
| F | F | | | |

Como ya dijimos, al no haber pasos intermedios, mirando f y h podemos completar todas las columnas restantes.

| f | h | Premisa 1 $f \rightarrow h$ | Premisa 2 $\neg h$ | Conclusión $\neg f$ |
|-----|-----|--------------------------------|-----------------------|------------------------|
| V | V | V | F | F |
| V | F | F | V | F |
| F | V | V | F | V |
| F | F | V | V | V |

Ahora que tenemos la tabla lista, cabe preguntarse si este razonamiento es válido o no. Lo que nos va a interesar no son las proposiciones atómicas, sino solamente las premisas y la conclusión, por lo que eliminaremos las dos primeras columnas.

| Premisa 1 $f \rightarrow h$ | Premisa 2 $\neg h$ | Conclusión $\neg f$ |
|--------------------------------|-----------------------|------------------------|
| V | F | F |
| F | V | F |
| V | F | V |
| V | V | V |

Ahora bien, debemos tener presente lo que significa que el razonamiento sea válido. Cuando un razonamiento es válido, para todos los casos en donde la totalidad de las premisas sea verdadera, la conclusión debe también ser verdadera. Busquemos primero las filas donde las premisas son ambas verdaderas.

| Premisa 1 $f \rightarrow h$ | Premisa 2 $\neg h$ | Conclusión $\neg f$ |
|--------------------------------|-----------------------|------------------------|
| V | F | F |
| F | V | F |
| V | F | V |
| V | V | V |

Como vemos, esto se cumple únicamente en la última fila (hemos marcado con amarillo las premisas en ese caso para que sea fácil de identificar). Una vez identificada las filas que cumplen el criterio, debemos verificar que para cada una de ellas, la conclusión sea también verdadera. En este caso, podemos ver que también lo es.

| Premisa 1 $f \rightarrow h$ | Premisa 2 $\neg h$ | Conclusión $\neg f$ |
|--------------------------------|-----------------------|------------------------|
| V | F | F |
| F | V | F |
| V | F | V |
| V | V | V |

Así, decimos que el razonamiento es **válido**. En el caso de que para cualquiera de las filas en donde las premisas son verdaderas la conclusión sea falsa, el razonamiento será **inválido**.

Veamos un segundo ejemplo, un poco más complejo. Esta vez iremos directamente por la fórmula de un razonamiento, sin importar cuál es el razonamiento concreto del cual se obtuvo.

■ Ejemplo 6.15

$$(p \rightarrow q) \rightarrow r, \neg p \vdash r$$
■

En este caso hay solo dos premisas, ya no tan simple la primera de ellas, por lo que necesitaremos pasos intermedios para calcular su valor de verdad en cada valuación de la tabla. La conclusión es trivial nuevamente. Realizaremos la tabla de verdad de manera idéntica a como lo hicimos en las ocasiones previas.

| p | q | Conclusión r | $p \rightarrow q$ | Premisa 1 $(p \rightarrow q) \rightarrow r$ | Premisa 2 $\neg p$ |
|-----|-----|-------------------|-------------------|--|-----------------------|
| V | V | V | V | V | F |
| V | V | F | V | F | F |
| V | F | V | F | V | F |
| V | F | F | F | V | F |
| F | V | V | V | V | V |
| F | V | F | V | F | V |
| F | F | V | V | V | V |
| F | F | F | V | F | V |

Podemos apreciar que la conclusión termina coincidiendo con una de las variables proposicionales, motivo por el cual aparece antes que las premisas. Primero, eliminemos todos los pasos intermedios, que fueron solo necesarios para calcular el valor de las premisas y conclusión, y quitemos también las variables proposicionales (no r , pues es la conclusión). Como el orden de las columnas no es importante, aprovecharemos además para cambiar el orden de los elementos, dejando la conclusión al final, para un análisis posterior más sencillo.

| Premisa 1 $(p \rightarrow q) \rightarrow r$ | Premisa 2 $\neg p$ | Conclusión r |
|--|-----------------------|-------------------|
| V | F | V |
| F | F | F |
| V | F | V |
| V | F | F |
| V | V | V |
| F | V | F |
| V | V | V |
| F | V | F |

Buscaremos los lugares donde ámbas premisas sean verdaderas.

| Premisa 1 $(p \rightarrow q) \rightarrow r$ | Premisa 2 $\neg p \wedge q$ | Conclusión r |
|--|--------------------------------|-------------------|
| V | F | V |
| F | F | F |
| V | F | V |
| V | F | F |
| V | V | V |
| F | V | F |
| V | V | V |
| F | V | F |

Como vemos hay dos valuaciones en donde las premisas son verdaderas al mismo tiempo. Es decir, lo que nos interesa es que en cada una de esas filas la conclusión sea verdadera.

| Premisa 1 $(p \rightarrow q) \rightarrow r$ | Premisa 2 $\neg p \wedge q$ | Conclusión r |
|--|--------------------------------|-------------------|
| V | F | V |
| F | F | F |
| V | F | V |
| V | F | F |
| V | V | V |
| F | V | F |
| V | V | V |
| F | F | F |

En este caso, vemos que efectivamente ocurre que la conclusión es verdadera cuando las premisas lo son, por lo que el razonamiento termina por ser válido.

Trabajaremos un último ejemplo. Este es similar al anterior, pero una de las premisas ha sido cambiada.

■ Ejemplo 6.16

$$(p \rightarrow q) \rightarrow r, p \vdash r$$

■

Siendo que ya hemos visto en reiteradas ocasiones como realizar la tabla de verdad paso a paso, iremos directo a la tabla final de este razonamiento.

| Premisa 1 $(p \rightarrow q) \rightarrow r$ | Premisa 2 p | Conclusión r |
|--|------------------|-------------------|
| V | V | V |
| F | V | F |
| V | V | V |
| V | V | F |
| V | F | V |
| F | F | F |
| V | F | V |
| F | F | F |

Nuevamente, para que el razonamiento sea válido, tenemos que buscar los lugares donde las premisas sean todas verdaderas, y verificar que allí la conclusión también lo sea.

| Premisa 1 $(p \rightarrow q) \rightarrow r$ | Premisa 2 p | Conclusión r |
|--|------------------|-------------------|
| V | V | V |
| F | V | F |
| V | V | V |
| V | V | F |
| V | F | V |
| F | F | F |
| V | F | V |
| F | F | F |

Podemos apreciar como en la tercera fila, valuación que tiene a p por verdadero, q por falso, y r por verdadero, da como resultado premisas verdaderas, pero una conclusión falsa. Es esta fila la que nos indica que el razonamiento es **inválido**, ya que encontramos un caso en donde, ante premisas verdaderas, la conclusión termina por ser falsa, violando lo que plantean los razonamientos deductivos.

6.7 Relación entre las premisas y la conclusión

Hemos mencionado como, en un razonamiento proposicional, cuando las premisas son verdaderas, la conclusión debe ser necesariamente verdadera. Otra forma de plantear esto es decir que “**Las premisas implican lógicamente a la conclusión**”.

Esto quiere decir que hay una relación lógica entre las premisas y la conclusión que sigue la siguiente forma:

Corolario 6.7.1 premisa 1 \wedge premisa 2 $\wedge \dots \wedge$ premisa N \rightarrow conclusión

Detengamonos un segundo uno de los razonamientos ya analizados, “*El gato en la caja está o bien vivo o bien muerto. Pero el gato no estaba muerto. Por lo tanto, el gato estaba vivo.*”. El indicador de conclusión, en el lenguaje natural, coincide en gran medida con las palabras que indican una implicación en una proposición compuesta. Esto se debe a que efectivamente expresa una idea de implicación, entre las premisas y la conclusión. Más aún, ese “*Pero*” que en su momento habíamos dejado de lado, coincide con una palabra que se usa para conjunción. Esto es así porque efectivamente cumple el rol de conjunción entre las premisas.

Si nos detenemos sobre la fórmula de dicho razonamiento, “ $p \vee \neg p, \neg \neg p \vdash p$ ”, podemos ver que es posible probar el razonamiento mediante su tabla de verdad y análisis posterior, de la siguiente forma:

| Premisa 1 $p \vee \neg p$ | Premisa 2 $\neg \neg p$ | Conclusión p |
|------------------------------|----------------------------|-------------------|
| V | V | V |
| V | F | F |

Otra forma para probar que el razonamiento sea válido sería calcular que las premisas impliquen a la conclusión, es decir, que “ $(p \vee \neg p) \wedge (\neg \neg p) \rightarrow p$ ” sea una **tautología**. Recordemos que, que sea tautología significa que siempre es verdadera, para cualquier valuación.

| Premisa 1 $p \vee \neg p$ | Premisa 2 $\neg \neg p$ | Conclusión p | Prueba de razonamiento completo $(p \vee \neg p) \wedge (\neg \neg p) \rightarrow p$ |
|------------------------------|----------------------------|-------------------|---|
| V | V | V | V |
| V | F | F | V |

Esto prueba el razonamiento, porque quiere decir que solo en aquellos casos donde las premisas

sean verdaderas, la conclusión también debe serlo, pero nada importa si las premisas son falsas.

Si nos detenemos a pensar la semántica de un razonamiento es donde podemos comprender la idea completa de lo que estamos hablando. Salgamos de las tablas de verdad y vayamos a un ejemplo sencillo “*Si yo soy Superman, entonces puedo volar. Y resulta que efectivamente soy Superman. Por lo tanto, es cierto que puedo volar.*” Con ese razonamiento en mente, subo a un noveno piso vestido con una capa roja, y salto desde el balcón... Por supuesto, la caída va a ser fatal.

El problema no está en el razonamiento en sí, el cual tiene una estructura perfectamente válida⁴, sino en las premisas. Si contrastamos con la realidad, sabemos que Superman es un personaje de ficción, que no existe, y que cualquier afirmación sobre que alguien sea Superman es falsa. **Partir de premisas falsas lleva a conclusiones incorrectas, incluso ante un razonamiento perfectamente válido.** La lógica no se ocupa sobre la realidad de las premisas en contraste con la realidad, sino sobre el análisis del razonamiento en sí, y queda en quien enuncia el razonamiento, asegurarse que las premisas tengan sentido.

Podemos entonces concluir que en un razonamiento hay una relación (lógica) entre premisas y conclusión. Esa relación está dada por la forma del razonamiento, y no por el contenido del mismo. Sin embargo, aunque la lógica estudia la relación dada por la forma, no analiza en ningún momento el contenido, el cual deberá ser sometido a contrastaciones empíricas si fuera necesario.

6.8 Álgebra de Boole

En 1857 George Boole publicó “*An investigation of the Laws of Thought*” (Una investigación sobre las Leyes del Pensamiento), una obra que postula procesos de análisis sobre elementos lógicos, con una completa explicación sobre los procesos de inferencia y deducción. La publicación es considerada como una obra fundacional, ya que establece procesos matemáticos (formales) para trabajar con la lógica, algo novedoso para la época.

A posteriori de su obra, otros autores ampliaron los conocimientos sobre la lógica, se formalizaron y descubrieron teoremas, nuevas formas de análisis, etc. elaborando un completo sistema formal al que se suele llamar **Álgebra de Boole**.

Los principios básicos de este álgebra versan sobre lo ya charlado en esta unidad, valores de verdad, proposiciones y conectivas, y análisis lógico de los valores de verdad expresados en proposiciones compuestas o razonamientos. Sin embargo, el álgebra de Boole equematiza las operaciones lógicas, permitiendo establecer teoremas y realizar pruebas sobre las fórmulas.

No vamos a meternos en la definición del álgebra de Boole, que es compleja y requiere un buen entendimiento matemático, sino más bien en los corolarios que de esta surgen, y como estos aplican a la lógica proposicional, por lo que esta sección será solo un vistazo simple a esta parte de la lógica.

6.8.1 Equivalencia de fórmulas

Decimos que dos fórmulas son equivalentes cuando, ante una misma valuación, se obtiene el mismo resultado en ambas formulas, y esto ocurre para toda valuación.

⁴Puede el lector realizar el pasaje a fórmulas y analizar la tabla de verdad del mismo como ejercicio.

Definición 6.8 Dos fórmulas se dicen **equivalentes** cuando producen el mismo valor de verdad ante una misma valuación, para toda valuación posible.

Un ejemplo podrían ser las fórmulas $p \wedge q$ y la fórmula $q \wedge p$. Equivalencia que podemos probar por tabla de verdad.

| p | q | $p \wedge q$ | $q \wedge p$ |
|-----|-----|--------------|--------------|
| V | V | V | V |
| V | F | F | F |
| F | V | F | F |
| F | F | F | F |

Si analizamos las últimas dos columnas, vemos como los valores que resultan son idénticos, en cada fila de la tabla. Esto nos habla de la equivalencia, la cual también puede probarse verificando que dos fórmulas unidas con una conectiva de equivalencia producen una tautología. En este ejemplo, verificar que $(p \wedge q) \leftrightarrow (q \wedge p)$ produce efectivamente una tautología.

| p | q | $p \wedge q$ | $q \wedge p$ | $(p \wedge q) \leftrightarrow (q \wedge p)$ |
|-----|-----|--------------|--------------|---|
| V | V | V | V | V |
| V | F | F | F | V |
| F | V | F | F | V |
| F | F | F | F | V |

De hecho, lo que acabamos de probar no es otra cosa que la **propiedad conmutativa** en la conjunción.

El álgebra de Boole se basa en este tipo de equivalencias para establecer sus **axiomas**, es decir, características que debe cumplir un sistema formal para ser un álgebra de Boole, en donde se definen relaciones entre las operaciones lógicas y los valores de verdad, y desde donde surgen los **teoremas**, postulados formales que surgen de los axiomas o de otros teoremas ya probados.

6.8.2 Axiomas y Leyes del Álgebra de Boole

El álgebra de Boole presenta 10 axiomas (5 en dos sabores diferentes), que son los siguientes.

- | | |
|---|---|
| 1a Ley de identidad en la conjunción $\Phi \wedge V \leftrightarrow \Phi$ | 1b Ley de identidad en la disyunción $\Phi \vee F \leftrightarrow \Phi$ |
| 2a Existencia del elemento complementario en la conjunción $\Phi \wedge \neg\Phi \leftrightarrow F$ | 2b Existencia del elemento complementario en la disyunción $\Phi \vee \neg\Phi \leftrightarrow V$ |
| 3a Ley asociativa de la conjunción $(\Phi \wedge \Psi) \wedge T \leftrightarrow \Phi \wedge (\Psi \wedge T)$ | 3b Ley asociativa de la disyunción $(\Phi \vee \Psi) \vee T \leftrightarrow \Phi \vee (\Psi \vee T)$ |
| 4a Ley conmutativa de la conjunción $\Phi \wedge \Psi \leftrightarrow \Psi \wedge \Phi$ | 4b Ley conmutativa de la disyunción $\Phi \vee \Psi \leftrightarrow \Psi \vee \Phi$ |
| 5a Ley distributiva de la conjunción respecto a la disyunción $\Phi \wedge (\Psi \vee T) \leftrightarrow (\Phi \wedge \Psi) \vee (\Phi \wedge T)$ | 5b Ley distributiva de la disyunción respecto a la conjunción $\Phi \vee (\Psi \wedge T) \leftrightarrow (\Phi \vee \Psi) \wedge (\Phi \vee T)$ |

Partiendo de los axiomas anteriores se pueden probar algunas leyes fundamentales.

- 6a Ley de **involución**
 $\neg(\neg\Phi) \leftrightarrow \Phi$

| | |
|--|--|
| 7a Ley de idempotencia en la conjunción $\Phi \wedge \Phi \leftrightarrow \Phi$ | 7a Ley de idempotencia en la conjunción $\Phi \vee \Phi \leftrightarrow \Phi$ |
| 8a Ley de absorción en la conjunción $\Phi \wedge F \leftrightarrow F$ | 8a Ley de absorción en la conjunción $\Phi \vee V \leftrightarrow V$ |
| 9a Ley de De Morgan (para la conjunción) $\neg(\Phi \wedge \Psi) \leftrightarrow \neg\Phi \vee \neg\Psi$ | 9a Ley de De Morgan (para la disyunción) $\neg(\Phi \vee \Psi) \leftrightarrow \neg\Phi \wedge \neg\Psi$ |
| 10a Ley de complemento (caso 1) $\neg V \leftrightarrow F$ | 10a Ley de complemento (caso 2) $\neg F \leftrightarrow V$ |

Sabías qué

El álgebra de Boole tiene una definición mucho más formal, y aplica no solo a la lógica proposicional, sino a la matemática binaria o la teoría de conjuntos. Cada rama emplea una notación distinta para referirse a los mismos elementos, por ejemplo, el lógica digital se suelen emplear los signos *0* y *1* para lo que nosotros estamos mencionando como *F* y *V*, y en conjuntos usan a idea de \emptyset y de U . También cambian los signos que se usan para los operaciones.

La aplicación sucesiva de estas leyes permite solucionar o simplificar fórmulas lógicas, como se muestra a continuación:

Ejemplo 6.17

$$\begin{aligned}
 & \neg p \wedge (\neg(\neg p \wedge q)) && \text{(Aplicando De Morgan)} \\
 & \neg p \wedge ((\neg\neg p) \vee (\neg q)) && \text{(Aplicando distributiva)} \\
 & (\neg p \wedge (\neg\neg p)) \vee (\neg p \wedge \neg q) && \text{(Aplicando involución)} \\
 & (\neg p \wedge p) \vee (\neg p \wedge \neg q) && \text{(Aplicando elemento complementario)} \\
 & F \vee (\neg p \wedge \neg q) && \text{(Aplicando comutativa)} \\
 & (\neg p \wedge \neg q) \vee F && \text{(Aplicando identidad)} \\
 & \neg p \wedge \neg q && \text{(Aplicando De Morgan, invertida)} \\
 & \neg(p \vee q) && \text{(6.2)}
 \end{aligned}$$

■

6.9 Reglas de inferencia

Cuando Aristóteles comenzó a analizar la lógica y formalizarla, desarrolló también las **reglas de inferencia**. En su concepción más simple, podemos decir que una regla de inferencia es un razonamiento de estructura bien conocida a la que se le da un nombre particular. Su definición formal, sin embargo, es la de una función que permite obtener una conclusión dada una serie de premisas.

Definición 6.9 Una **regla de inferencia**, o también **regla de transformación** es una función que toma premisas como entrada, analiza su sintaxis y describe una conclusión como salida.

Es decir, si tenemos una serie de premisas que cumplen una estructura particular, que cae dentro de uno de los casos de inferencia, podemos determinar fácilmente su conclusión. Por supuesto, las reglas de inferencia pueden combinarse entre sí, y con las reglas de reemplazo del álgebra de Boole, para obtener conclusiones más complejas.

A continuación veremos algunas de las reglas que fueron determinadas por Aristóteles en la antiguedad, y que es bueno tener en consideración a la hora de pensar razonamientos. Por supuesto no son las únicas reglas existentes, pero trabajaremos sobre otras en las actividades finales de este capítulo.

6.9.1 Modus ponendo ponens

El *modus ponendo ponens*, latín para **el modo que, al afirmar, afirma**, también llamado **afirmación del antecedente** es una regla con la siguiente forma:

$$\begin{array}{c} \Phi \rightarrow \Psi \\ \Phi \\ \hline \Psi \end{array}$$

Podemos analizar fácilmente este razonamiento. Si el antecedente es verdadero, la única forma de que la implicación sea verdadera es que el consecuente sea también verdadero. Así, si sabemos que vale la implicación, y que vale el antecedente, podemos concluir que necesariamente debe valer el consecuente.

Un ejemplo de este tipo de razonamientos es el siguiente:

Si hay fuego, entonces habrá humo
Hay fuego

Hay humo

Podríamos analizar por tabla de verdad este razonamiento, verificando su validez, aunque esto se deja como ejercicio para el lector.

6.9.2 Modus tollendo tollens

El *modus tollendo tollens*, latín para **el modo que, al negar, niega**, también llamado **negación del consecuente** es una regla con la siguiente forma:

$$\begin{array}{c} \Phi \rightarrow \Psi \\ \neg \Psi \\ \hline \neg \Phi \end{array}$$

Podemos nuevamente analizar fácilmente que este razonamiento es válido. Si sabemos que el consecuente es falso, pero la implicación es verdadera, entonces no hay forma alguna de que el antecedente sea verdadero, pues si lo fuera la implicación sería falsa. Solo puede darse entonces como conclusión que el antecedente es necesariamente falso.

Si seguimos la misma implicación del ejemplo anterior, el razonamiento sería:

Si hay fuego, entonces habrá humo
No hay humo

No hay fuego

Es importante destacar que la premisa es la negación del consecuente, y no del antecedente, ya que de la negación de este último no podría elaborarse ninguna conclusión, algo que analizamos en la Sección 6.10.

6.9.3 Modus ponendo tollens

El *modus ponendo tollens*, latín para **el modo que, al afirmar, niega**, es una regla que establece que, si no es posible que dos proposiciones sean verdaderas al mismo tiempo, y una de ellas es verdadera, entonces la otra debe ser necesariamente falsa. La fórmula de este razonamiento es:

$$\begin{array}{c} \neg(\Phi \wedge \Psi) \\ \Phi \\ \hline \neg\Psi \end{array}$$

Un ejemplo en lenguaje natural podría ser el siguiente:

No es cierto que Gómez y Rodriguez puedan ser
ámbos presidentes al mismo tiempo
Gómez es presidente

No es cierto que Rodriguez sea presidente

Nuevamente el análisis es sencillo. La única forma de que la conjunción sea verdadera, es cuando ámbas proposiciones son verdaderas. La negación antes de la conjunción es verdadera siempre, salvo en el caso de la conjunción verdadera (Es decir, afirma que no pueden ocurrir Φ y Ψ al mismo tiempo). El hecho de que ocurra uno de ellos, hace obligatorio que el otro no ocurra para que la premisa sea verdadera, y por tanto la única conclusión posible es que la otra proposición debe ser falsa.

6.9.4 Modus tollendo ponens

El *modus tollendo ponens*, latín para **el modo que, al negar, afirma**, es una regla que establece que, si una de dos proposiciones debe ser verdadera, y una de ellas es falsa, entonces la otra debe ser necesariamente verdadera. La fórmula de este razonamiento es:

$$\begin{array}{c} \Phi \vee \Psi \\ \neg\Phi \\ \hline \Psi \end{array}$$

Para que la disyunción sea verdadera, uno de los elementos debe ser verdadero. Si sabemos que uno de ellos es falso (por la segunda premisa), entonces podemos concluir que la otra proposición es verdadera.

Un ejemplo en lenguaje natural podría ser el siguiente:

O bien Gomez es presidente o bien Rodriguez es
presidente
Gómez no es presidente

Rodriguez es entonces el presidente

Todas estas reglas de inferencia nos permiten entender mejor las conectivas y los razonamientos, y entender de mejor forma cuándo un razonamiento es válido y cuándo no. Si bien podemos siempre probar un razonamiento mediante tablas de verdad, la realidad es que si nos encontramos con un razonamiento que sigue una de las reglas de inferencia, podríamos simplemente afirmar su validez, ya que la misma fue probada hace más de 2000 años.

6.10 Falacias bien conocidas

Así como hay formas bien conocidas de razonamientos válidos, también hay formas bien conocidas de razonamientos inválidos, llamados **falacias**. Las falacias tienen estructuras que, para el ojo no entrenado, resultan válidas, pero que realmente no lo son.

Conocer las falacias nos permite entender mejor las estructuras de los razonamientos válidos, pero también los inválidos. Conocerlas nos evita caer en errores comunes. Veremos entonces algunas de las falacias más comunes que podemos encontrar.

6.10.1 Afirmación del consecuente

La *afirmación del consecuente* es tal vez uno de los errores más comunes. Su estructura es la siguiente:

$$\begin{array}{c} \Phi \rightarrow \Psi \\ \Psi \\ \hline \Phi \end{array}$$

Es decir, sabiendo verdadera la implicación, y verdadero su consecuente, se afirma que vale el antecedente. Un ejemplo en lenguaje natural sería:

Si hay fuego, entonces habrá humo
Hay humo

Hay fuego

Se puede ver en la tabla de verdad, si analizamos las valuaciones donde las premisas son verdaderas, que la conclusión no necesariamente lo es en toda valuación:

| Premisa 1 | Premisa 2 | Conclusión |
|-------------------------|-----------|------------|
| $\Phi \rightarrow \Psi$ | Ψ | Φ |
| V | V | V |
| F | F | V |
| V | V | F |
| V | F | F |

De hecho ya hemos analizado anteriormente este caso, el humo en realidad podría haberse producido por otros motivos, que no son la presencia de fuego. La implicación solo dice que sucede cuando el antecedente se cumple, no cuando no se cumple, y el consecuente podría ser verdadero incluso con el antecedente falso, y la implicación seguiría siendo verdadera.

6.10.2 Negación del antecedente

La implicación suele ser lo que más dolores de cabeza trae a los recién iniciados, y por eso la *negación del antecedente* es otra forma de falacia bien conocida, cuya fórmula para el razonamiento es:

$$\begin{array}{c} \Phi \rightarrow \Psi \\ \neg\Phi \\ \hline \neg\Psi \end{array}$$

Surge de la confusión de que, si sucede el antecedente entonces sucederá el consecuente, y por tanto afirmar su inversa (si no sucede el antecedente entonces tampoco sucederá el consecuente). Como vimos, esto no es cierto, incluso si el antecedente es falso, el consecuente podría ser

perfectamente verdadero, manteniendo verdadera a la implicación. Un ejemplo sería

Si hay fuego, entonces habrá humo
No hay fuego

No hay humo

Nuevamente, ya hemos discutido como el humo puede producirse por otros motivos que no sean la presencia de fuego.

6.11 Límites de la lógica proposicional

Vamos a volver sobre nuestros pasos para formalizar un razonamiento que ya hemos visto.

■ Ejemplo 6.18

Todos los humanos son mortales. Sócrates es un humano. Por lo tanto, Sócrates es mortal.



En primer lugar, podemos identificar que “*Por lo tanto*” es el indicador de conclusión. Esto hace que “*Sócrates es mortal*” sea la conclusión, mientras que las otras dos oraciones sean las premisas.

Si intentamos pasar todo a fórmula veremos que se requieren tres variables proposicionales, ya que las tres oraciones afirman cosas diferentes. Es decir, el diccionario y la fórmula resultantes serían:

■ Ejemplo 6.19

Diccionario:

p = Todos los humanos son mortales.

q = Sócrates es humano.

r = Sócrates es mortal.

Fórmula:

p

q

r



Inmediatamente podemos apreciar que este razonamiento es falso, pues la conclusión es algo completamente independiente de las premisas, y estas no son contradictorias.

Sabías qué

En los razonamientos deductivos la conclusión siempre se desprende de las premisas. Es decir, tras el pasaje a fórmula siempre la conclusión debe incluir proposiciones que estén mencionadas en las premisas, aunque éstas pueden estar combinadas de formas diferentes a como lo estaban en las premisas.

La única situación en donde la conclusión puede no desprenderse de las premisas es la reducción al absurdo, es decir, cuando las premisas se prueban necesariamente falsas y absurda. Un ejemplo es afirmar que vale p y al mismo tiempo vale $\neg p$ en las premisas. Por ser una situación imposible, cualquier conclusión que surja de dichas premisas es válida, incluso si nada tiene que ver con las premisas.

Sin embargo, cuando leemos el razonamiento, nos resulta intuitivo pensar en su validez. ¿Es acaso realmente inválido? Bueno, si, lo acabamos de analizar. Pero, ¿Dónde está el truco? Si vemos

la estructura interna vamos a apreciar que, si bien no aparece dos veces la misma proposición, si hay elementos que se repiten internamente entre las distintas proposiciones.

■ **Ejemplo 6.20**

Todos los humanos son mortales
Sócrates es un humano

Sócrates es mortal

Sócrates es siempre el mismo, tanto en la premisa como en la conclusión. La idea de “ser un humano” o de “ser mortal” también aparece en más de una ocasión. Sin embargo la lógica proposicional no permite reflejar esa estructura gramatical.

El gran problema entonces es que este tipo de razonamientos caen fuera de lo que la lógica proposicional permite modelar. Es decir, no se pueden formalizar adecuadamente razonamientos en donde la semántica está relacionada a la estructura gramatical interna de las proposiciones. Este tipo de razonamientos no tiene solución en la lógica proposicional, hemos llegado al límite de la misma.

Deberemos entonces recurrir a un nuevo formalismo para poder modelar este razonamiento de forma apropiada. En el Capítulo 7 se propone otro sistema lógico, que permite solucionar el problema de este razonamiento, pero también de otros.

6.12 Actividades

Ejercicio 6.1 Identifique cuales de las siguientes son proposiciones y cuales no lo son.

- a Microsoft es la empresa detrás del sistema operativo Windows.
- b ¿Hay vida en Marte?
- c Las células eucariotas tienen núcleo y las procariotas no lo tienen.
- d ¡Bienvenidos chicos!
- e ¿Es un celular inteligente?
- f No es cierto que haya empresas que beneficien a algunos políticos.
- g Las leyes más importantes.

Ejercicio 6.2 Identifique las conectivas y las proposiciones atómicas involucradas, luego realice la traducción al lenguaje de la lógica proposicional de las siguientes oraciones:

- a Safari es un navegador y viene instalado en macOS.
- b Safari es un navegador o bien es un sistema operativo.
- c Safari es un navegador, y no es cierto que viene instalado en Windows.
- d O bien vamos al cine o bien al teatro, más no ambos, pero compraremos pochoclos.
- e Linux puede funcionar en computadoras de escritorio, en dispositivos móviles o en otros dispositivos.
- f GIMP no es ni un sistema operativo ni un navegador.
- g Manjaro es una distribución de Linux basada en Arch y viene con el escritorio Gnome, el escritorio KDE y con otros escritorios populares.
- h No es cierto que Debian sea la distribución más vieja de Linux, pero Slackware si lo es.
- i El software son los programas y el SO de una PC, o bien, son los componentes físicos y periféricos de una PC.

- j Solo encontraremos vida en marte si vamos a dicho planeta.
- k Si viene Richard Stallman a la Argentina, iré a verlo y le pediré un autógrafo.
- l Me encanta ver jugar a Messi ya que es un gran jugador de fútbol y a mi me gusta ese deporte.
- m Argentina solo saldrá de su penosa situación económica si llega al poder un gobierno fuerte con ideas nuevas.

■

Ejercicio 6.3 A partir de los siguientes enunciados, determine cuáles son **proposiciones** solamente y cuáles son **razonamientos**.

En caso de que sean proposiciones diga además si es atómica o compuesta, encontrando las conectivas.

En caso de ser un razonamiento, diga cuáles son las premisas y cual la conclusión, identificando el indicador de premisa/conclusión que se está utilizando.

- a Empezaré karate y judo.
- b O bien me inscribo a judo y a karate, o bien a jiu jitsu. Me inscribí en jiu jitsu. Por lo tanto, no me inscribí a karate y a judo.
- c Voy a cursar los martes y los viernes. Ya que, o bien iba a cursar los lunes y jueves, o bien los martes y viernes. Pero no voy a cursar los lunes.
- d Me inscribí a jiu jitsu.
- e Si me inscribo a karate me volveré muy fuerte. Pero no me volví muy fuerte. Por lo tanto, no me inscribí a karate.
- f No es cierto que uno se vuelva fuerte ya que se inscribió a jiu jitsu.

■

Ejercicio 6.4 Determine cuáles de las siguientes son fórmulas bien formadas, y cuáles no lo son.

- a $p \rightarrow q \wedge r$
- b $p \vee q \neg p$
- c $p \wedge (\neg q \leftrightarrow \neg r)$
- d $p \wedge (\wedge q \rightarrow r)$
- e $p \wedge (q(\rightarrow r))$
- f $p \wedge \neg q \leftrightarrow \neg r$

■

Ejercicio 6.5 Analice los siguientes razonamientos y extrapole la fórmula lógica de cada proposición. Luego, pruebe si el razonamiento es válido o inválido.

- a Si hay bananas o hay manzanas entonces hay fruta. No hay manzanas. Por tanto, no hay fruta.
- b Si se comienza a tratar el calentamiento global se podrá detener a tiempo. Pero no se comienza a tratar el calentamiento global. Por tanto, no se podrá detener a tiempo.
- c No es cierto que se deba detener el ascensor. Dado que, el ascensor se debe detener si y solo si está frente a la puerta de un piso. Y no es cierto que el ascensor esté frente a la puerta de un piso.
- d Se han movilizado tropas aliadas al norte. Por lo tanto, hay ejércitos enemigos al norte.

■

- Ya qué si no hay ejércitos enemigos al norte entonces no es necesario movilizar tropas aliadas en esa dirección.
- e Si un software es libre, entonces tiene una licencia libre. Si un software tiene licencia libre entonces garantiza las cuatro libertades del software libre. Por lo tanto, si un software es libre, entonces garantiza las cuatro libertades del software libre.
 - f La teoría de cuerdas une la gravedad con la mecánica cuántica, por tanto, es una teoría de como funciona nuestro universo. La teoría de cuerdas requiere de diez dimensiones para funcionar. En consecuencia, nuestro universo cuenta con diez dimensiones.
 - g Hay que verificar su existencia o hay que tener fé ciega en su existencia. Es posible verificar su existencia si y solo si se cuenta con el equipo adecuado. No se cuenta con el equipo adecuado. En consiguiente, hay que tener fé ciega en su existencia.
 - h Si y solo si se cuenta con suficiente dinero se podrá construir el edificio. Si se ha vendido suficiente cantidad de mineral, entonces se contará con suficiente dinero. Por ende, se puede construir el edificio.
 - i Si cambia el panorama político de la AFA entonces la situación futbolística se revierte. Adicionalmente, el panorama político de la AFA no va a cambiar si el presidente de la AFA no toma una decisión radical o no sucede que alguien más accede a la dirección. Por consiguiente, si el presidente toma una decisión política radical o alguien más accede a la dirección es que cambiará el panorama político de la AFA.
 - j La Tierra es plana o es redonda. Si la Tierra es redonda entonces no nos caeremos por el borde. En cambio, si la Tierra es plana, si nos caeremos por el borde. Pero la Tierra no es plana. En consecuencia, no nos caeremos por el borde.
 - k El ascensor está en movimiento. Esto es así ya que si el ascensor no abre sus puertas es porque o bien está en movimiento, o bien se está preparando para moverse. Y la puerta está cerrada y el ascensor no se está preparando para moverse.
 - l Si la celda actual está pintada de rojo y la celda siguiente a la derecha está pintada de negro, entonces el cabezal se moverá dos lugares a la derecha. El cabezal no se movió dos lugares a la derecha. En consiguiente, o bien la celda actual no estaba pintada de rojo, o bien la celda siguiente a la derecha no estaba pintada de negro.
 - m Los pasajeros no murieron. Por tanto, la U.S.S Enterprise los transportó con éxito en la superficie. Ya qué, si la U.S.S. Enterprise no los transportaba con éxito en la superficie, entonces los pasajeros morirían.

Ejercicio 6.6 Analice las siguientes implicaciones y determine en cada situación si

Ejercicio 6.7 Dadas las siguientes estructuras lógicas formalizadas de razonamientos, analizar si los mismos son válidos o inválidos.

1. $p \wedge q \vdash p$
2. $p \vee q, \neg q \vdash p$
3. $p \vee q, \neg q \vdash \neg p$
4. $p \vee q, \neg q \vdash p$
5. $p \rightarrow q, p \vdash q$
6. $\neg p \rightarrow q, \neg q \vdash \neg p$
7. $p \rightarrow q, \neg q \vdash \neg p$
8. $p \rightarrow q, q \rightarrow r, p \vdash r$
9. $(p \vee q) \rightarrow r, p \wedge \neg q \vdash r$

10. $(p \vee q) \rightarrow r, \neg p \wedge \neg q \vdash r$

■

TODO Ejercicio sobre probar el resto de las reglas de inferencia y/o falacias

Ejercicio 6.8 A partir de las siguientes estructuras lógicas de razonamientos, piense distintas proposiciones en lenguaje natural o ejemplos que se pueden aplicar a los mismos:

1. $p \vee q$

$\neg q$

2. p

$p \vee q$

3. $\neg q$

$p \rightarrow q$

4. q

4. $(p \wedge q) \rightarrow \neg s$

$p \wedge q$

$\neg s$

■

Ejercicio 6.9 Dados los siguientes razonamientos, identifique los indicadores de premisa, los indicadores de conclusión, y estructure el razonamiento en premisas y conclusión.

1. Si hay bananas o hay manzanas, entonces hay fruta. No hay manzanas. Pero hay fruta. Por lo tanto, hay bananas.
2. Será Santino quien venga a la fiesta. Dado que a la fiesta iba a venir o bien Guadalupe o bien Santino. Pero Guadalupe no va a venir.
3. Argentina clasificó para el mundial FIFA 2022. Esto es así porque o bien Perú clasifica para el mundial o bien Argentina clasifica para el mundial. Pero Perú no clasificó.
4. Instagram es la red social más utilizada. Dado que Instagram es la red social más utilizada y Facebook se utiliza cada vez menos.
5. Me compro un pantalón o me compro una remera, pero no ambas. Me compro una remera. Por lo tanto, me compro un pantalón o no me compro un pantalón.

■

Ejercicio 6.10 Determine si las siguientes expresiones son equivalentes. Para ello primero traduzca del lenguaje natural al lenguaje formal de la lógica proposicional, y utilice la conectiva correspondiente para unir ambas expresiones y comprobar si son equivalentes o no.

1. ¿Será lo mismo la frase “**Hay vida en Marte y hay vida en Ganimedes**” a la frase “**Hay vida en Ganimedes y hay vida en Marte**”?
2. ¿Será lo mismo la frase “**Hay vida en Marte o hay vida en Ganimedes**” a la frase “**Hay vida en Ganimedes o hay vida en Marte**”?
3. ¿Será lo mismo la frase “**Si hay vida en Marte entonces hay vida en Ganimedes**” a la frase “**Si hay vida en Ganimedes entonces hay vida en Marte**”?

4. ¿Será lo mismo la frase “**No es cierto que no hay vida en Marte**” a la frase “**Hay vida en Marte**”?
5. ¿Será lo mismo la frase “**No es cierto que, hay vida en Marte o hay vida en Ganimedes**” a la frase “**No hay vida en Marte y no hay vida en Ganimedes**”?
6. ¿Será lo mismo la frase “**No es cierto que, hay vida en Marte y hay vida en Ganimedes**” a la frase “**No hay vida en Marte o no hay vida en Ganimedes**”?

■

Ejercicio 6.11 Pruebe que los axiomas del álgebra de Boole se cumplen, verificando las equivalencias expresadas.

■



7. Lógica de predicados

Red Lógica de un Enrutador con LEDs.

Fotografía de tschoenemeyer.

En el capítulo anterior vimos que la lógica proposicional no nos permite formalizar todos los razonamientos que se nos puedan ocurrir. Así, el pobre Sócrates podría terminar herido de muerte por alguien que solo concibe la lógica proposicional como el único formalismo lógico.

Por suerte para Sócrates, y para el resto de nosotros, con el tiempo varios pensadores se dieron cuenta de las problemáticas asociadas a la lógica proposicional, y elaboraron otros formalismos capaces de modelar razonamientos que antes no eran posibles. La **lógica de predicados**, también llamada **lógica de primer orden** o **lógica de orden uno** es uno de tales formalismos.

La lógica de predicados tiene la particularidad de que permite hablar sobre los diversos elementos gramaticales internos a las proposiciones, es decir, sobre los individuos que menciona, sus propiedades y relaciones. Además nos va a permitir versar sobre el universo, y todos los elementos del mismo, o algunos de ellos, sin ponerles a estos nombres específicos.

Si la lógica proposicional tenía a la **proposición** como el elemento portador de verdad, la lógica de predicados tendrá al **predicado** como el elemento mínimo capaz de ser verdadero o falso. En la siguiente sección veremos en qué consiste esto de los predicados, pero para entenderlo, deberemos primero poder “desmenuzar” una proposición

7.1 Individuos, Propiedades y Relaciones

Pensemos en algunas proposiciones que podíamos enunciar e intentemos comprender mejor la estructura interna de dichas proposiciones. Vamos con nuestro ejemplo clásico, que menciona a Sócrates.

■ **Ejemplo 7.1**

Sócrates es humano

■

Estamos ante algo de lo que podemos decir que es verdadero o falso, es decir, frente a una proposición. Pero como vimos, lo que deseamos es poder comprender la estructura interna de estas

oraciones. Aquí es donde el comprender sujeto y predicado se vuelve útil. ¿Sobre quién habla la oración? Es decir, ¿Quién es el sujeto? Sócrates, claro.

7.1.1 Individuos

Los **individuos** son los elementos sobre los que versa la lógica de predicados. Pueden ser personas, como “Sócrates”, planetas como “*La Tierra*”, objetos como “*la silla*”, elementos ideales “*el color rojo*” o no tangibles como “*el alma*”. Son, en definitiva, los **sujetos** en las oraciones que antes mencionábamos como proposiciones, que podrían incluso incluir modificadores directos o indirectos.

Hay una particularidad interesante, y es que un individuo es un único elemento. Si decimos algo como “*las personas*”, “*los astronautas*” o “*los colores*” no estamos hablando de un individuo, sino de varios. Veremos como lidiar con la situación de nombrar varios elementos al mismo tiempo más adelante. De momento, tenga presente el lector que para que algo sea un individuo debería en principio estar en enunciado singular y no en plural.

Los individuos no tienen un valor de verdad asociado, es decir, “Sócrates” no es ni verdadero ni falso, es una persona, una entidad sobre la cual vamos a mencionar cosas, pero en sí misma, no tiene valor de verdad alguno.

Al formalizar en la lógica de predicados, vamos a elaborar un **diccionario**, de forma similar a lo que hacíamos para lógica proposicional. Sin embargo, este diccionario es bastante diferente, ya que uno de los elementos que debemos incluir en el mismo son los individuos. A cada individuo se le asignará una letra en minúscula en el diccionario. En general optamos por elegir una letra relacionada al individuo en cuestión, por ejemplo “*s*” para “Sócrates” o “*t*” para “*El planeta Tierra*”. Incluso podríamos asignar combinaciones de letras, como “*sm*” para “*San Martín*”.

A estas letras se las conoce como **constantes**, y cada una representa un individuo específico del discurso.

7.1.2 Propiedades

Sobre los individuos vamos a querer decir diferentes cosas. Por ejemplo, podemos querer mencionar algunas características que pueden llegar a tener, como el “*ser un humano*” o “*ser mortal*”. Llamamos a estas características **propiedades**. Una propiedad no es más que una característica que puede aplicar (o no) a un individuo particular. Por ejemplo, la propiedad “*ser un color primario*” es una característica que aplica a “*el color rojo*”, pero no a “*el color violeta*”.

Por supuesto, también vamos a querer escribir propiedades en nuestro diccionario. Las mismas van a estar identificadas por una letra en mayúscula, y también buscaremos que esté relacionada a la propiedad, por ejemplo “*P*” para la idea de color primario, o “*H*” para la idea de ser un humano.

Como las propiedades versan sobre una característica de un elemento, y como el lenguaje natural en ocasiones no nos permite expresar claramente la característica sin mencionar al sujeto, vamos a mencionar al individuo en la oración, con un nuevo concepto. Por ejemplo, vamos a decir “*x es humano*”. Al momento de definir el diccionario, no sabemos a priori quien es “*x*”, por lo que actúa de forma similar a las incógnitas en matemáticas. Decimos que es una **variable**, que tiene sentido mencionar dentro de la propiedad. Definiremos entonces la propiedad en términos de esa variable, como se muestra en los siguientes ejemplos:

■ Ejemplo 7.2

$$H(x) = x \text{ es humano} \quad M(x) = x \text{ es mortal} \quad P(x) = x \text{ es un color primario}$$

■

Notar que para vamos a definir una propiedad “ A ” como “ $A(x)$ ” (se lee “ A de equis”). En el texto mencionaremos esa “ x ” en el punto en donde mencionaríamos un individuo. Así, la interpretación que debe hacerse es que “ A ” es algo que espera un individuo cualquiera, y dice si ese individuo es es humano.

Al momento de escribir la fórmula, vamos a dar un valor concreto a ese “ x ”, por ejemplo, “*Sócrates*”, por lo que usaremos la constante que asignamos a dicho individuo. Veamos un ejemplo concreto.

■ Ejemplo 7.3

Texto a pasar a fórmula:

Sócrates es humano

Diccionario:

$s = Sócrates$

$H(x) = x \text{ es humano}$

Fórmula:

$H(s)$

■

Notar que la idea de la fórmula es tomar el texto de la propiedad, y reemplazar la “ x ” con el individuo concreto que se ha aplicado, o más bien, con la interpretación del diccionario de la constante aplicada. En este ejemplo, se usa la constante “ s ” aplicada a la propiedad “ H ”, por lo que se debe reemplazar la “ x ” en “ $x \text{ es humano}$ ” (la interpretación de “ H ”), con la interpretación de “ s ”, “*Sócrates*”. Esto da como resultado la frase “*Sócrates es humano*”, el texto original a pasar a fórmula.

Notar que una propiedad, en principio es algo abstracto. Es decir, “ $x \text{ es humano}$ ” es una propiedad, pero no podemos decir ni que sea verdadera ni falsa. Sin embargo, al aplicar la propiedad a un individuo, se transforma en un valor de verdad. Decir que “*Sócrates es humano*” si es algo de lo que podamos decir que sea verdadero o falso.

Para quienes conozcan de matemática, una propiedad no es más que una función de individuos en valores de verdad. Es decir, si le damos a la función un individuo concreto, nos describirá verdadero o falso dependiendo de si ese individuo cumple o no dicha propiedad. Así, en el diccionario tendremos la definición de dicha función (Ej. “ $H(x) = x \text{ es humano}$ ”), y en la fórmula, la función aplicada a un individuo concreto (Ej. “ $H(s)$ ”).

TODO Hablar de alfa-conversión de variables

TODO: Relaciones

7.2 Predicados

TODO

7.2.1 Aridad

TODO

7.2.2 Predicados y conectivas

TODO

7.3 Universo de discurso

TODO

7.4 Cuantificadores

TODO

7.4.1 Cuantificador universal

TODO

7.4.2 Cuantificador existencial

TODO

7.4.3 Cuantificador existencial negado

TODO

7.4.4 Equivalencia de cuantificadores

TODO

7.5 Actividades

Ejercicio 7.1 Dados los siguientes textos, determine si se trata de individuos o no, y justifique su respuesta en caso negativo.

- a) Lisa
- b) Los astronautas
- c) José de San Martín
- d) Paquetes
- e) La selección argentina
- f) Los jugadores de la selección

Ejercicio 7.2 Pasar cada una de las siguientes oraciones a fórmula, identificando claramente el diccionario, los individuos, las propiedades y las relaciones entre los mismos. Recuerde identificar adicionalmente las conectivas que pudieran aparecer.

- a) La Tierra es plana
- b) Los Beatles es un grupo musical inglés
- c) Lisa quiere a Nelson
- d) El gato de Schrödinger está vivo o muerto

- e) El verde es un color secundario, pero el azul no lo es.
- f) Freddy Mercury fue el vocalista de Queen, y Kurt Cobain el de Nirvana.
- g) Freddy Mercury no cantaba con Kurt Cobain.

Ejercicio 7.3 Dados los siguientes dominios, pensar para cada uno al menos 3 individuos que formen parte del mismo, dos propiedades y dos relaciones que apliquen a los elementos de dicho dominio.

Si conoce poco del tema, puede investigar en Internet sobre el mismo para determinar cosas sobre el mismo.

- a) El espacio exterior
- b) Las partículas subatómicas
- c) Los deportes
- d) Las computadoras
- e) El software libre

Ejercicio 7.4 Dados los siguientes diccionarios, elabore al menos dos proposiciones compuestas en el lenguaje natural que utilicen los elementos del diccionario.

a) $j = \text{Juan}$

$m = \text{María}$

$a = \text{Ana}$

$l = \text{Luis}$

$A(x) = x \text{ vive en Argentina.}$

$B(x) = x \text{ vive en Brasil.}$

b) $a = \text{Argentina}$

$b = \text{Brasil}$

$u = \text{Uruguay}$

$J(x,y) = x \text{ jugó contra } y.$

$G(x,y) = x \text{ le ganó a } y.$

$C(x) = x \text{ salió campeón del torneo.}$

c) $me = \text{Messi}$

$ma = \text{Maradona}$

$p = \text{Pel}$

$a = \text{La selección argentina}$

$b = \text{La selección brasileña}$

$J(x,y) = x \text{ jugó en } y$

$P(x,y) = x$ tiene en su plantel a y

■

Ejercicio 7.5 Pase cada una de las siguientes oraciones en el lenguaje natural a una fórmula de la lógica de predicados. Para cada una indique su diccionario y fórmula.

- El 7 es un número natural y es impar.
- Windows es un sistema operativo privativo y Linux es un sistema operativo libre.
- Algunas computadoras usan Linux.
- Todos los programas de software libre garantizan las 4 libertades.
- No todos los perros tienen pelo.
- Ningún animal es más inteligente que el pulpo.
- Todo número es igual a si mismo.
- Algunos astronautas son de Rusia, otros son de Estados Unidos.
- Ningun lenguaje de programación soluciona todos los problemas.
- No hay números que sean primos y sean pares y que además sean mayores a 2.
- Si un número es primo entonces no hay ningún número que sea su divisor, salvo el 1 y el mismo.

■

Ejercicio 7.6 Dado que el dominio son los productos de software y las empresas que los producen, y se cuenta con el siguiente diccionario.

$w = \text{WhatsApp}$

$f = \text{Facebook}$

$i = \text{Instagram}$

$g = \text{Gmail}$

$m = \text{Meta}$

$a = \text{Alphabet}$

$S(x) = x$ es un software.

$E(x) = x$ es una empresa que produce software.

$B(x) = x$ tiene más de un billón de dólares.

$U(x) = x$ es muy usado.

$D(x,y) = x$ es dueña de y

$F(x,y) = x$ fabricó y

$C(x,y) = x$ compró y

Se pide que pase al lenguaje natural las siguientes fórmulas de la lógica de predicados.

- $D(m,f) \wedge D(m,i) \wedge D(m,w)$
- $\forall x.E(x) \rightarrow B(x)$
- $\exists x.\neg E(x) \wedge B(x)$
- $\exists x.\forall y.E(x) \wedge D(x,y) \rightarrow \neg F(x,y)$

Ejercicio 7.7 Considere las siguientes expresiones que representan una famosa variación del juego piedra-papel-tijeras:

- Las tijeras cortan al papel.
- El papel envuelve a la piedra.
- La piedra aplasta al lagarto.
- El lagarto envenena a Spock.
- Spock destruye las tijeras.
- Las tijeras decapitan al lagarto.
- El lagarto se come al papel.
- El papel desautoriza a Spock.
- Spock vaporiza la roca.
- La piedra aplasta las tijeras.

Tenga en cuenta que la expresión “tijera corta al papel” representa que la tijera vence al papel. Es decir, toda expresión, cualquiera sea, puede ser reformulada en término de, **el primer elemento vence al segundo**.

Se pide entonces complete la tabla a continuación para expresar quien vence a quien en dicho juego, completando con **V** o **F**.

| x vence a y | Piedra | Papel | Tijeras | Lagarto | Spock |
|-----------------|--------|-------|---------|---------|-------|
| Piedra | | | | | |
| Papel | | | | | |
| Tijeras | | | | | |
| Lagarto | | | | | |
| Spock | | | | | |

Programación

IV

V

Índice y Anexos

Índice 169

Anexos

A Markdown 177

A.1 Sintaxis y Semántica
A.2 Actividades

B HTML 185

B.1 Browsers
B.2 W3C y estándares de la web
B.3 Sistema de etiquetas como marcas
B.4 Etiquetas básicas de HTML
B.5 Documentos HTML válidos
B.6 CSS
B.7 Formularios y respuestas del servidor
B.8 Actividades

C Simbología Lógica y Tablas de Verdad 213

C.1 Simbología de las conectivas y sus palabras asociadas en lenguaje natural en la lógica proposicional
C.2 Simbología de la lógica de predicados
C.3 Tablas de verdad de conectivas

Índice alfabético

Red Lógica de un Enrutador con LEDs.
Fotografía de tschoenemeyer.

A

| | |
|-----------------------------------|------------|
| Ábaco | 28 |
| Actividades | 25, 47 |
| Ad-hoc | 23 |
| Algoritmo..... | 28, 32 |
| Altair 8800 | 39 |
| Amiga OS | 41 |
| Analisis de expresiones | 109 |
| Android | 22 |
| Antivirus | 25 |
| Análisis mediante semántica | 110 |
| Aplicación | 22 |
| Apple..... | 39, 41 |
| Apple II | 40 |
| Apple Macintosh | 41 |
| Archivo | 23, 24, 66 |
| Archivos Informáticos | 65 |
| Aristóteles..... | 98 |
| ARPANET | 43 |
| Arquitectura de Computadora | 35 |
| Arquitectura de Harvard | 35 |
| Arquitectura de Von Neumann..... | 35 |
| ASCC | 34 |
| AT&T | 36 |
| Atari 2600..... | 40 |
| Automatización | 46 |

B

| | |
|------------------------|----------------|
| Babbage, Charles | 30 |
| Bajo nivel | 51 |
| BBC Micro..... | 40 |
| Bell Labs..... | 35 |
| BeOS | 41 |
| Big Data | 45 |
| Binario | 28, 55, 56, 59 |
| Bit..... | 60 |
| Boole, George | 98 |
| Browsers | 83 |
| BSD | 88 |
| Byte | 60 |

C

| | |
|----------------------------------|--------|
| Cables | 53 |
| Caja Negra | 61 |
| Calculadoras Mecánicas | 30 |
| Carruaje que apunta al sur | 28 |
| Chip | 38 |
| Church-Turing, Tesis de | 32 |
| Church, Alonzo | 32 |
| Ciencia formal..... | 100 |
| Cilindro de Leibniz..... | 30 |
| Circuito Electrónico..... | 53 |
| Circuito Integrado | 38 |
| Circuitos Integrados | 38 |
| Codificación | 56, 58 |

| | |
|------------------------------------|--------|
| Código | 53 |
| Código Binario | 55, 56 |
| Código Fuente | 79 |
| Código Objeto | 79 |
| Colecovision | 40 |
| Commodore 64 | 40 |
| Compilador | 79 |
| Compilar | 79 |
| Computación Biológica | 47 |
| Computación Cuántica | 46 |
| Computadora | 11 |
| Computadora Electromecánica | 33 |
| Computadora Mecánica | 30 |
| Computadoras Humanas | 29 |
| Comunicación | 51 |
| Conclusión | 99 |
| Conectiva binaria | 106 |
| Conectiva unaria | 106 |
| Conjunción | 105 |
| Contingencia | 115 |
| Contradicción | 115 |
| Copyleft | 42, 87 |
| Copyright | 84 |
| CP/M | 41 |
| CPU | 13 |
| Cuarta Revolución Industrial | 46 |

D

| | |
|---------------------------|------------|
| De Morgan, Augustus | 98 |
| Debuggear | 34 |
| Debugging | 34 |
| Deducción | 99, 100 |
| Derecho de autor | 84 |
| Diccionario | 123 |
| Directorio | 23, 25, 69 |
| Disyunción | 106 |
| Domótica | 45 |
| DOS | 41 |

E

| | |
|------------------------------|----|
| Editor | 76 |
| Editor de Texto | 76 |
| EDVAC | 34 |
| Electricidad | 53 |
| ENIAC | 34 |
| Exabit | 60 |
| Exabyte | 60 |
| Extensiones de Archivo | 67 |

F

| | |
|--------------------------------|-----|
| Falso | 101 |
| Firmware | 19 |
| Formato | 66 |
| Fórmula lógica | 122 |
| FOSS | 89 |
| Free Software Foundation | 42 |
| Freeware | 86 |
| Frege, Gottlob | 98 |
| Fuente de Alimentación | 13 |

G

| | |
|----------------------------|----|
| Gates, Bill | 39 |
| Gigabit | 60 |
| Gigabyte | 60 |
| Gödel, Kurt | 31 |
| GPL | 87 |
| Grandes Computadoras | 35 |

H

| | |
|------------------------------------|---------|
| Hardware | 12, 17 |
| Harvard Mark I | 34 |
| Harvard, Arquitectura de | 35 |
| Herramientas de desarrollo | 25 |
| Hillbert, David | 31 |
| Historia de las computadoras | 27 |
| Homebrew Computer Club | 39 |
| HTML | 83, 173 |
| Hueso de Ishango | 28 |

I

| | |
|---|--------|
| IBM | 35 |
| Identificador de Recursos Universal | 75 |
| Indicador de conclusión | 125 |
| Indicador de premisa | 128 |
| Información | 56 |
| Informática | 65 |
| Intel 8080 | 39 |
| Inteligencia Artificial | 46 |
| Intellivision | 40 |
| Internet | 17, 43 |
| Internet de las Cosas | 44 |
| Interpretar | 79 |
| Interprete | 79, 80 |
| Introducción a la lógica | 97 |
| IoT | 44 |

ISP 17

J

Jaquard, Joseph Marie 30
Jerarquía 24
Jobs, Steve 39

K

Kilby, Jack S. 38
Kildall, Gary 41
Kilobit 60
Kilobyte 60

L

La prehistoria de la computación 28
LAN 17
Leibniz, Cilindro de 30
Leibniz, Gottfried 30
Lenguaje 77
Lenguaje Compilado 80
Lenguaje de Alto Nivel 80
Lenguaje de Bajo Nivel 80
Lenguaje de Marcado 81
Lenguaje de Medio Nivel 80
Lenguaje de Programación 78
Lenguaje Formal 78
Lenguaje Interpretado 80
Lenguaje natural 104
Lenguajes de Programación 36
Libertades del Software Libre 86
Licencia 84
Licencia de Software 42
Licenciatario 84
Linux 21, 88
Lógica 97, 99
Lógica de orden cero 121
Lógica de predicados 151
Lógica proposicional 121
Lovelace, Ada 30
Lógica 99

M

Macintosh 41
Magnavox Odyssey 40
Maquina AnalíticaMáquina Analítica 30
Máquina de Turing 32

Máquina Diferencial 30
Markdown 165
Mecanismo de Anticintera 28
Megabit 60
Megabyte 60
Memoria RAM 13
Micral N 39
Microchip 38
Microcomputadoras 39
Microprocesador 39
Microsoft 39
Microsoft Windows 41
MIT 88
Motherboard 13
Motorola 6800 39
MS-DOS 41

N

Navegador de Archivos 25
Negación 106
NORSAR 43
Números 59

O

Open Source 88
Open Source Initiative 42, 88

P

PAN 17
Paradigma 80
Paradigma Funcional 80
Paradigma Imperativo 80
Paradigma Orientado a Objetos 80
Parte Física 12
Parte Lógica 18
Pasaje a fórmula 122
Pascal, Blaise 30
Pascalina 30
PC 41
Peano, Giuseppe 98
Periférico 13
Periféricos 13
Periféricos de Almacenamiento 16
Periféricos de Entrada 14
Periféricos de Entrada/Salida 15
Periféricos de Salida 15
Petabit 60

| | |
|-----------------------------|--------|
| Petabyte | 60 |
| Pierce, Charles | 98 |
| Piratería | 84 |
| precedencia | 109 |
| Premisa | 99 |
| Principia Mathematica | 98 |
| Programa | 22, 76 |
| Proposición | 121 |
| Proposición atómica | 122 |
| Proposición compuesta | 122 |
| Proposición molecular | 122 |
| Proposición simple | 122 |
| Proyecto GNU | 42 |

R

| | |
|------------------------------|---------|
| Razón | 97 |
| Razonamiento | 99 |
| Razonamiento deductivo | 100 |
| Razonamiento inductivo | 100 |
| Razonamientos | 99, 125 |
| Razonar | 99 |
| Red de Computadoras | 17 |
| Redes | 17, 43 |
| Relé | 33 |
| Router | 17 |
| Rusell, Bertrand | 98 |
| Ruta | 70, 72 |
| Ruta Absoluta | 70 |
| Ruta Relativa | 72 |

S

| | |
|---|------------|
| SaaS | 86 |
| SAGE | 43 |
| Schröder, Ernst | 98 |
| Semántica | 78 |
| Shareware | 85 |
| Simbología Lógica y Tablas de Verdad .. | 201 |
| Sintaxis | 78 |
| Sistema Binario | 56 |
| Sistema Binario | 28, 55 |
| Sistema de Archivos | 69 |
| Sistema Operativo | 20, 40 |
| SMP80/88 | 39 |
| Software | 17, 18 |
| Software a Medida | 86 |
| Software como Servicio | 86 |
| Software Libre | 41, 86, 87 |
| Software Privativo | 87 |

| | |
|----------------------------|--------|
| Software Propietario | 87 |
| Stallman, Richard | 42, 86 |
| Suanpan | 28 |
| Switch | 17 |

T

| | |
|-----------------------------------|-----|
| Tabla semántica | 106 |
| Tarjeta Perforada | 30 |
| Tautología | 115 |
| Teorema de la Incompletitud | 31 |
| Teoría de la Computabilidad | 32 |
| Teoría de la Computación | 31 |
| Terabit | 60 |
| Terabyte | 60 |
| Tesis de Church-Turing | 32 |
| Texas Instruments | 38 |
| Torvalds, Linus | 88 |
| Transistores | 38 |
| Tubo de Vacío | 34 |
| Turing, Alan | 32 |
| Turing, Máquina de | 32 |

U

| | |
|------------------|--------|
| UNIX | 41 |
| URI | 74, 75 |
| URL | 74 |
| Utilidades | 25 |

V

| | |
|------------------------------------|-----|
| Válido | 134 |
| Valor de verdad | 101 |
| Valores de verdad | 101 |
| Valuación | 110 |
| Variable proposicional | 123 |
| Verdadero | 101 |
| Visualizador | 76 |
| Von Neumann, Arquitectura de | 35 |
| Von Neumann, John | 35 |

W

| | |
|-------------------------|--------|
| WAN | 17 |
| Whitehead, Alfred | 98 |
| Windows | 21, 41 |
| Wozniak, Steve | 39 |

X

| | |
|------------------|----|
| XEROX PARC | 41 |
|------------------|----|

XML 82

Z

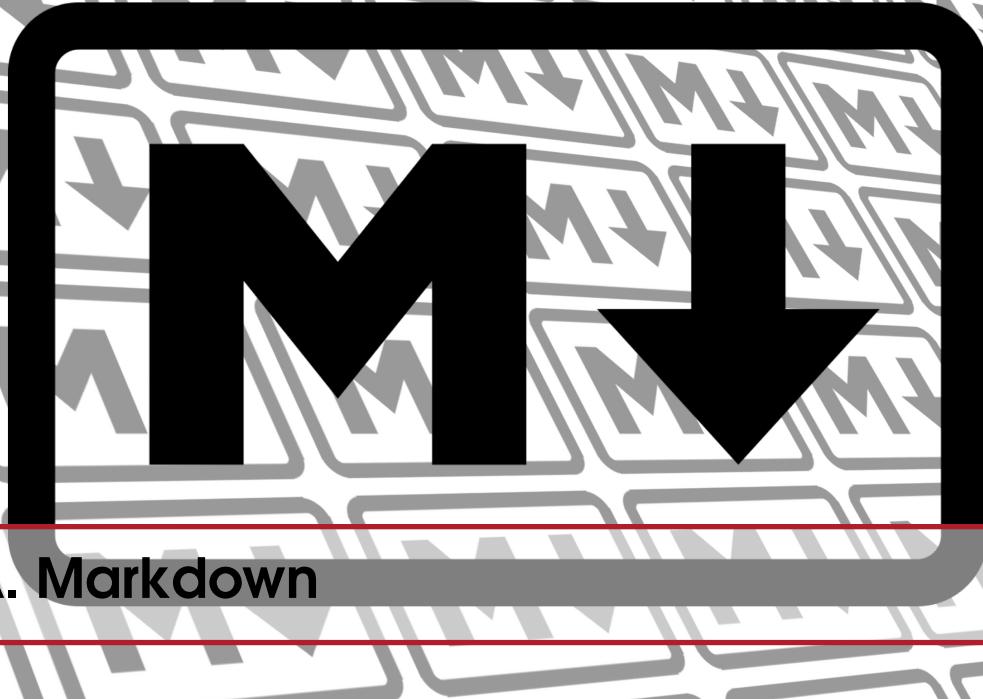
Z1 33

Z2 33

Z3 33

Z80 39

Anexos



A. Markdown

Logo de Markdown sobre fondo con logos tridimensionales.
Diseño propio.

Markdown es un lenguaje de marcado ligero creado por **John Gruber**, con ayuda de **Aaron Swartz**, que trata de conseguir la máxima legibilidad y facilidad de publicación tanto en su forma de entrada como de salida. Gruber deseaba que la gente pudiera escribir un documento de forma sencilla, y que pudiera comprenderlos, sin necesidad de saber demasiado sobre el lenguaje, y aún así producir documentos válidos que pudieran ser exportados luego a documentos HTML válidos.

Si bien su lanzamiento oficial fue en el 2004, se inspira en una gran cantidad de convenciones existentes para marcar mensajes de correo electrónico en texto plano, muy utilizadas en la década de 1980 y principios de 1990. Markdown es un lenguaje libre, y también lo son sus herramientas, que se distribuyen bajo una **licencia BSD**. La versión original fue creada en el lenguaje de programación Perl, pero con el tiempo se ha portado a muchos otros lenguajes.

No hay un estándar definido para Markdown, aparte de la implementación original de John Gruber, que algunos consideran obsoleto. Esto ha ocasionado una gran fragmentación, con diversas implementaciones que agregan funciones, o remueven otras.

Markdown se ha posicionado como un importante lenguaje en diversos ámbitos, por ejemplo, varios sistemas de blogs permiten escribir artículos utilizando Markdown.

En el área de programación es ampliamente utilizado para escribir documentación de los programas, convirtiéndose en el estándar de facto. Por ejemplo, sitios que albergan repositorios de código, como **GitHub**, **GitLab** y **Bitbucket** entre otros, incluyen visualizadores para Markdown que muestran la documentación del programa de forma automática. Cómo estos sitios albergan proyectos de software libre, leer la documentación de estos programas es una buena forma de aprender Markdown, ya que se puede ver fácilmente la visualización y su código fuente.

También cabe destacar que, si bien se puede escribir código Markdown con cualquier editor de texto simple, existen editores especializados o plugins para editores que permiten visualizar la salida como HTML, resaltan la sintaxis y proveen funciones de autocompletado. También existen editores con visualizadores que corren directamente online, sin necesidad de instalar nada en el equipo, como el sitio web <http://dillinger.io/>.

Los archivos que contienen código markdown suelen guardarse con la extensión **.md**, o con extensión **.markdown**, según la herramienta que se utilice para producirlos.

Cabe destacar que Markdown permite exportar el contenido a HTML, un lenguaje de marcado que se utiliza para diseñar páginas web. Sin embargo, sus características están muy reducidas con respecto a este último, por lo que el diseño de páginas no es su función. En cambio, está pensado para redactar correos electrónicos, artículos simples y documentación.

A.1 Sintaxis y Semántica

Párrafos y texto simple

Todo texto escrito en un documento Markdown que no contenga marcas específicas aparece tal cual, como un párrafo. Un salto de línea en el texto original no se ve reflejado en el resultado visualizado.

Los saltos de línea en la salida se dan cuando se insertan dos espacios consecutivos en la entrada (donde un salto de línea es considerado un espacio).

```
1 Esto es un párrafo único, incluso  
2 si hay un salto de línea en el medio.  
3  
4 Esto aparece en otro párrafo, pues hay dos  
5 saltos de línea desde el párrafo anterior.
```

Énfasis

Existen dos formas de enfatizar una palabra en Markdown, un énfasis simple (que en general se visualiza como texto en itálica), y un énfasis doble (para palabras o frases un poco más importantes que las anteriores, y que se termina visualizando en negrita).

El énfasis consiste en colocar una marca (para énfasis simple) o dos (para énfasis doble) al inicio del texto a resaltar, y la misma cantidad al final de dicho texto. Pueden utilizarse tanto guiones bajos como asteriscos para marcar, pero en general se estila utilizar guion bajo para énfasis simple, y asteriscos para negrita.

```
1 Esto texto tiene estas palabras con énfasis simple,  
2 pero estas que están a continuación **Presentan énfasis doble**
```

La salida del texto anterior se visualizará de forma similar a lo siguiente:

Esto texto *tiene estas palabras con énfasis simple*,
pero estas que están a continuación **Presentan énfasis doble**

Títulos

Los títulos permiten delimitar secciones bien diferenciadas en el texto, y ya son por si mismos texto resaltado, por lo que no deben rodearse de marcas de énfasis.

Los títulos vienen en diversos niveles, de acuerdo a la importancia del mismo. Por ejemplo, los títulos de primer nivel son los más importantes, y por tanto deberían ser utilizados solo como el título general del documento. Los títulos de nivel dos actúan como subtítulo, y permiten diferenciar secciones grandes del documento. Títulos de tercer, cuarto, quinto y sexto nivel pueden ser utilizados para marcar secciones más específicas.

Los títulos, si bien tienen un estilo destacado, tienen una connotación semántica clara. No deberían ser utilizados para enfatizar palabras.

Los títulos comienzan con uno o más signos de numeral, llamado también almohadilla, o hash (#). La cantidad de almohadillas expresa el nivel del título, siendo una almohadilla los títulos de primer nivel, dos almohadillas de segundo nivel, y así siguiendo.

```
1 # Título de primer nivel
2 ## Título de segundo nivel
3 ### Título de tercer nivel
4 ##### Título de cuarto nivel
5 ##### Título de quinto nivel
6 ##### Título de sexto nivel
```

Lo que se termina visualizando como:

Título de primer nivel

Título de segundo nivel

Título de tercer nivel

Título de cuarto nivel

Título de quinto nivel

Título de sexto nivel

También existe una forma distinta de escribir títulos de primer y segundo nivel, que consiste en “subrayar” las palabras que conforman el texto. Un subrayado doble indica títulos de primer nivel, y se logra utilizando signos de igual (=) para realizar el subrayado. Un subrayado simple indica títulos de segundo nivel, y se logra utilizando guiones medios, o signos de resta (-). El subrayado debe tener al menos dos signos consecutivos, pero lo ideal para maximizar la lectura en la entrada es que abarque tanto como el texto que está marcando.

```
1 Título de primer nivel  
2 =====  
3  
4 Título de segundo nivel  
5 -----
```

Listas no ordenadas

Las listas no ordenadas marcan cada uno de sus elementos como viñetas. Se pueden utilizar asteriscos (*), signos de suma (+), o guiones o signos de resta (-) para marcar los elementos de la lista. Incluso pueden combinarse, aunque esto atenta contra la lectura en la entrada y no es recomendado.

```
1 + Item 1  
2 + Item 2  
3 + Item 3
```

Resulta en:

- Item 1
- Item 2
- Item 3

También pueden colocarse listas dentro de otras listas, tabulando los elementos

```
1 + Item 1  
2 + Item 2 tiene los siguientes elementos  
3     * Item 2A  
4     * Item 2B  
5 + Item 3
```

Resultando en:

- Item 1
- Item 2 tiene los siguientes elementos
 - Item 2A
 - Item 2B
- Item 3

Listas ordenadas

Las listas ordenadas, también llamadas listas numeradas, presentan números en lugar de viñetas. Se logran colocando un número al comienzo de la línea y siguiéndolo de un punto.

- ```
1 1. Primer elemento
2 2. Segundo elemento
3 3. Tercer elemento
```

1. Primer elemento
2. Segundo elemento
3. Tercer elemento

De igual forma que con las listas desordenadas, estas se pueden anidar. Más aún, se pueden anidar listas ordenadas con no ordenadas en diversas formas.

### Enlaces

Una característica interesante es la posibilidad de insertar enlaces en cualquier punto del texto, que apuntan a otros documentos o sitios web. Un enlace se compone de dos partes: la dirección a la cual se quiere acceder, y el texto que se le mostrará al lector. El texto se coloca entre corchetes, los cuales van seguidos de unos paréntesis que contienen la dirección a la cual se enlazará.

- ```
1 Lo siguiente es un enlace: [Ir a Google](https://google.com)
```

Lo siguiente es un enlace: [Ir a Google](https://google.com)

Imágenes

En ocasiones es necesario agregar algún tipo de elemento visual, en particular, imágenes. Las imágenes funcionan de forma similar a un enlace, pero se debe anteponer un signo de admiración (!) a los corchetes. El texto que se coloca entre corchetes corresponde con la descripción de la imagen, mientras que la dirección entre paréntesis corresponde con la ubicación de la misma.

- ```
1 ! [Logo de Markdown Here](http://cor.to/mdhere)
```



## Código

Por haber sido pensado para escribir documentación de código de programación, Markdown incluye una forma de mostrar código. Para ello se utilizan comillas agudas, llamadas también backsticks (`). El texto entre backsticks se corresponde a código fuente. Si el mismo tiene varias líneas, se deben incluir tres backsticks en lugar de uno, pudiendo colocar al final de los backsticks de apertura, en el inicio y entre corchetes, el lenguaje en el que se está escribiendo el código.

```
1 'let x = 5'
2
3 ''' [JavaScript]
4 let x = 5;
5 let y = 7;
6 let z = x + y;
7'''
```

Lo cual se vería así:

```
let x = 5
```

```
let x = 5;
let y = 7;
let z = x + y;
```

## Separaciones

También es posible colocar líneas de separación entre diferentes secciones del texto. Una línea de separación se ve como una fina línea horizontal que abarca todo el ancho del documento, y puede hacerse con una serie de signos de resta o guiones (-) colocados de forma consecutiva.

También pueden utilizarse asteriscos o signos de igual para la separación.

```
1 Esta sección se encuentra separada de la que le sigue.
2
3 -----
4
5 La línea superior marca esa separación de forma clara.
```

## A.2 Actividades

**Ejercicio A.1** Dado el siguiente código en Markdown, modifique el mismo de forma tal de dar énfasis simple a los periféricos que aparecen mencionados y énfasis doble a las palabras informática, software y hardware en todos los lugares en donde aparece.

```
1 La informática es la disciplina que que estudia el
2 tratamiento automatizado de información por parte
3 de las computadoras, las cuales se componen de un
4 conjunto de hardware y software.
5
6 El hardware se compone de una placa base, una
7 fuente de alimentación, memoria, un microprocesador,
8 y un conjunto de periféricos tales como mouse,
9 teclado, pantalla, parlantes, impresora, micrófono
10 y placas de expansión.
11
12 El software consiste en los programas que corremos
13 en el equipo, como Word, GIMP, Age of Empires y otros.
```

**Ejercicio A.2** Redacte en Markdown una receta para hacer tallarines caseros, utilizando una lista no ordenada para los mencionar los ingredientes y una lista ordenada para los pasos a seguir en la preparación. Puede valerse de Internet para buscar una receta, en caso de que desconozca los pasos a seguir.

**Ejercicio A.3** Busque en Internet imágenes de tres lugares del mundo que desearía visitar. En un documento Markdown, inserte las imágenes de forma consecutiva, colocando previo a cada una el nombre del lugar como título de nivel 3, y el país en el que se encuentra como subtítulo de nivel 5. El título debe consistir en un enlace a la página de Wikipedia sobre ese lugar.

**Ejercicio A.4** El siguiente código Markdown tiene algunos errores, tanto semánticos como de sintaxis.

```
1 ##### Sobre Markdown
2
3 ## El lenguaje
4
5 El lenguaje Markdown fue creado en 2004 por John Gruber.
6 Es un lenguaje de marcado ligero diseñado para maximizar la
7 facilidad de escritura y lectura tanto en la entrada como
8 en la salida.
9
10 -----
11
12 ## _La sintaxis_
13
14 La sintaxis del lenguaje es sencilla, y consiste en signos
15 como asteriscos, almohadillas, y corchetes para lograr
16 estructurar un documento. En general se pueden escribir
17 los elementos de la siguiente lista:
18
19 Títulos
20 Palabras con énfasis
21 Listas
22 Enlaces
23 Código
24
25 -----
26
27 Copyright 2019
```

Se pide entonces que detecte los errores y proponga un código que los corrija.

**Ejercicio A.5** Elabore un documento Markdown que funcione como su curriculum vitae. Para ello, coloque su nombre completo como título principal, y siga al mismo con un subtítulo de nivel dos indicando su profesión. Agregue una foto de su persona, y seccione el contenido en datos personales, trabajos, formación y otros elementos utilizando títulos de tercer nivel en adelante y separadores según lo considere necesario. Utilice una lista no ordenada para sus datos personales.

No hace falta que el CV sea real, basta con que sea verosímil, si no cuenta con experiencia laboral o estudios, puede inventarlos para realizar esta actividad.

**Ejercicio A.6** Realice en Markdown una breve carta de presentación donde usted solicita un puesto de “Desarrollador de Software” en una empresa. Utilice títulos, énfasis y listas de forma apropiada en los lugares que crea conveniente.



Imagen de un diseñador web pensando en código.  
Diseño de Mudassar Iqbal con retoques propios.

**HyperText Markup Language (HTML)** es un lenguaje de marcado que sirve para describir páginas web, por lo que se ha transformado en el lenguaje de marcado más extendido del mundo.

HTML es un lenguaje basado en XML, por lo que su principio fundamental son las etiquetas, que actúan de marcas para indicar distintos tipos de contenidos en la página web que describen.

Un archivo de HTML es simplemente un archivo de texto, que en general se identifican con la extensión **.html**, y que puede editarse con cualquier editor de texto. Para visualizar el contenido como un sitio web, se debe utilizar un programa visualizador especial, que en general se conoce como **browser** o **navegador web**.

## B.1 Browsers

El browser tiene una doble funcionalidad, por un lado, permite conectarse a dispositivos remotos, los cuales se identifican con una URL específica, y descargar de este uno o más archivos. Por otro lado permite visualizar los archivos HTML, interpretando su contenidos y dando una muestra visual de los contenidos (Esto incluye la interpretación de CSS y de JavaScript). Cuando uno coloca una URL como “”, el browser ingresa a la máquina identificada como “google.com” y descarga por defecto el archivo “index.html” del mismo. Otras URLs, como “” descargan el archivo específico “archivo.html”.

Existen muchos browsers en el mercado, algunos privativos, otros libres. Entre los más destacados están:

- Internet Explorer
- Edge
- Google Chrome
- Mozilla Firefox
- Opera
- Safari

## B.2 W3C y estándares de la web

A diferencia de otros lenguajes, HTML se encuentra claramente especificado en lo que se conoce como el estándar HTML, el cual está definido por el **World Wide Web Consortium (W3C)**, una organización encargada de definir estándares para la web, y promover su correcto uso.



Logos de diferentes browsers. Existe una amplia diversidad, con navegadores que se enfocan en mercados sumamente específicos.

Logos del proyecto [github.com/alrra/browser-logos](https://github.com/alrra/browser-logos).

Cabe destacar que HTML define una estructura para el contenido, y son los navegadores los que determinan de que forma mostrar dicho contenido (el estilo). Si se desea especificar estilos más avanzados, debe recurrirse al uso de otro lenguaje de la web, conocido como “Hojas de Estilo en Cascada”, o en inglés **Cascade Style Sheets (CSS)**. Este lenguaje complementa a HTML, permitiendo definir cosas como los colores de texto, tamaños de fuentes, ubicación de imágenes, colores de fondo, distribución de contenidos en la página, entre una gran serie de otros elementos. CSS también está especificado por la W3C.

Finalmente, si lo que se desea es lograr una interactividad en el sitio (por ejemplo, que se muestre cierto contenido cuando el usuario selecciona una opción, o cuando presiona un botón), se debe recurrir a un lenguaje de programación conocido como **JavaScript** (o **ECMAScript**), el cual permite manipular las etiquetas del documento HTML, agregando o quitando contenido, atributos, e incluso nuevas etiquetas en cualquier lugar del documento. JavaScript también está definido por la W3C, complementando así los otros lenguajes.

La W3C también define otros estándares que permiten la definición de imágenes vectoriales para la web, o formatos de transmisión de datos entre equipos, entre otra serie de cosas.

A lo largo del tiempo se han definido varias versiones de HTML, siendo las primeras bastante primitivas, permitiendo poca interacción con el usuario, mientras que actualmente se puede definir aplicaciones enteras usando HTML, CSS y JavaScript. La versión actual de HTML es la versión 5, y es una versión “viva”, es decir, no hay una versión 5 final, sino que el estándar se va actualizando a medida que surge la necesidad de utilizar nuevas etiquetas. Así, algunas etiquetas clásicas desaparecen, y otras nuevas aparecen. Junto con HTML5 aparecen nuevos estándares para CSS (CSS 3, también una versión “viva”), y JavaScript (En versiones cada vez más modernas adhiriendo

al estándar ECMAScript que publica una nueva versión cada año).

Por ser tan longevo, y por las cualidades de ser un lenguaje vivo, la documentación que uno puede encontrar en Internet sobre HTML puede estar desactualizada y haberse vuelto obsoleta. A esto se debe sumar el hecho de que mucha gente aprende por su cuenta, y luego hace público su conocimiento, el cuál no siempre es correcto. Por eso es importante para los desarrolladores web seguir los estándares de la W3C y asegurarse de que su código siga dichos estándares.

¿Qué ocurre si nuestro código no sigue el estándar? En principio, nada. Cualquier “buen browser” (entiéndase casi todos los navegadores web modernos, a excepción de Internet Explorer) garantiza que los códigos que siguen el estándar de la W3C se verán correctamente. Sin embargo, cada uno tiene su propia implementación, y su propia idea de “que hacer” ante código que no sigue la norma. Así, código que funciona en Google Chrome puede no verse bien en Firefox o en Internet Explorer.

Si nos encontramos realizando un sitio que va a ser utilizado por el público en general, debemos seguir necesariamente el estándar, pues no podemos garantizar cual es el navegador web que va a utilizar la persona que ingrese. Internet Explorer, por gozar de una situación monopólica en la mayoría de los equipos domésticos, pudo no seguir los estándares. Esto hacía que cualquier desarrollador que hiciera una página, debía realizar dos versiones de la misma, una para Internet Explorer, y otra para el resto de los navegadores. La situación se mantuvo hasta la salida de Edge por parte de Microsoft, reemplazando a Internet Explorer, forzada por las decisiones de múltiples empresas y desarrolladores de dejar de dar soporte a Internet Explorer.

### B.3 Sistema de etiquetas como marcas

En los lenguajes basados a etiquetas, la etiqueta es el elemento fundamental que permite marcar una porción de texto, y es, en sí mismo, un elemento que puede ser marcado por otras etiquetas.

Toda etiqueta está dada por una marca que comienza por el signo “<” (menor), se sigue de un **nombre**, y cierra con el signo “>” (mayor). El nombre de la etiqueta consiste en una única palabra, que puede contener letras, números, guiones bajos o guiones medios (no puede tener espacios).

Además, la mayoría de las etiquetas consisten en dos marcas, una como la mencionada anteriormente, conocida como marca de “**apertura**”, y otra que es conocida como marca de “**cierre**”, y que a diferencia de las anteriores utiliza “</” (menor seguido de barra) para el comienzo de la marca.

Así, la siguiente es la marca de apertura:

```
1 <nombre_de_etiqueta>
```

Y la siguiente una etiqueta de cierre:

```
1 </nombre_de_etiqueta>
```

El nombre de la etiqueta debe ser el mismo tanto en la marca de apertura como en la de cierre. La etiqueta marca el texto que se encuentra entre la marca de apertura y la de cierre, llamado “**contenido**” de la etiqueta.

```
1 <nombre_etiqueta> Contenido </nombre_etiqueta>
```

El contenido de una etiqueta pueden ser otras etiquetas:

```
1 <etiqueta1>
2 <etiqueta2>
3 Contenido de la etiqueta
4 </etiqueta2>
5 </etiqueta1>
```

También puede darse el caso de que se requiera información adicional para marcar un texto que el solo nombre de la etiqueta. Esa información adicional puede ser brindada a la etiqueta mediante un “**atributo**”. Un atributo consiste en un par de elementos, conocidos como “**clave**” (el nombre del atributo) y “**valor**” (el valor del atributo). Estos se colocan únicamente en la etiqueta de apertura, separando la clave del valor mediante un signo igual (=), y donde el valor siempre debe escribirse entre comillas.

```
1 <nombre_de_etiqueta atributo="valor">
```

En una misma etiqueta pueden colocarse tantos atributos como se desee, separando cada atributo del otro con un espacio. Note como el primer atributo también se separa del nombre de la etiqueta mediante un espacio.

```
1 <nombre attr1="val1" attr2="val2">
```

Finalmente, en ocasiones es necesario agregar contenido que sea ignorado por el visualizador. Este contenido sirve para que el mismo autor del código pueda estructurar sus ideas, o dejar mensajes a otros posibles lectores del código fuente, y se conocen como “**comentarios**”.

Un comentario comienza cuando se encuentran los signos “`<!--`” y termina cuando se encuentran los signos “`-->`”. Un comentario puede abarcar varias etiquetas, e incluso varias líneas de texto.

```
1 <!--
2 Esto es un comentario,
3 puede ponerse lo que se desee aquí,
4 incluso etiquetas como <hola> o </hola>
5 y serán ignoradas
6 -->
```

Además, vale aclarar que en HTML, así como en otros lenguajes de etiquetas, un espacio, múltiples espacios, una o más tabulaciones, o incluso uno o más saltos de línea son equivalentes a un único espacio. Así, estas cuatro etiquetas son exactamente equivalentes:

```
1 <etiqueta attr1="val1" attr2="val2">
2 <etiqueta attr1="val1" attr2="val2">
3 <etiqueta attr1="val1"
```

```
4 attr2="val2">
5 <etiqueta
6 attr1="val1"
7 attr2="val2"
8 >
```

Note como el signo “>” de cierre de la marca, puede también estar en otra línea. También vale para el de apertura.

Por último, considere el siguiente código:

```
1 <etiqueta1><etiqueta2>Contenido de la etiqueta
2 </etiqueta2><etiqueta2 attr="val1"><etiqueta3>Contenido de la
3 etiqueta 3
4 </etiqueta3>
5 </etiqueta2></etiqueta1>
```

Observe como se vuelve complicado identificar que etiqueta contiene a que otra, y qué marca tiene cada uno. Considere ahora la siguiente versión con el mismo código, pero estructurado de una forma diferente:

```
1 <etiqueta1>
2 <etiqueta2>
3 Contenido de la etiqueta
4 </etiqueta2>
5 <etiqueta2 attr="val1">
6 <etiqueta3>
7 Contenido de la etiqueta 3
8 </etiqueta3>
9 </etiqueta2>
10 </etiqueta1>
```

Observe como esta forma de estructurar el contenido facilita su lectura y su entendimiento. Esto se conoce en programación como “**indentación del código**”, es decir, tabular y estructurar el código de forma tal que sea fácil identificar las partes del mismo. La palabra “indentación” viene de la palabra inglesa “*indent*”, y su equivalente en español sería “tabulado”. Es deseable que todo el código se encuentre correctamente indentado. En los lenguajes de etiquetas, cada marca de cierre debería colocarse a la misma altura horizontal que su marca de cierre. Luego, el contenido dentro de una etiqueta se debe colocar en una nueva línea y siempre tabulado dos o cuatro espacios (un ancho de tabulación) más a la derecha que la etiqueta que lo contiene.

## B.4 Etiquetas básicas de HTML

HTML, como lenguaje de etiquetas, sigue la misma estructura mencionada en la sección anterior, pero el conjunto de etiquetas que pueden utilizarse y la forma en la que deben estructurarse las mismas. También, el efecto que produce cada etiqueta al ser visualizada está definido en la especificación del lenguaje.

Existen etiquetas para una gran cantidad de elementos, y este apunte no pretende ser una guía comprensiva de los mismos, así como tampoco estar necesariamente actualizado. Por esto se

recomienda tener a mano los siguientes enlaces para aprender sobre nuevas etiquetas o validar la información sobre las aquí expuestas.

- <https://developer.mozilla.org/es/>
- <https://www.w3schools.com>

## Párrafos y énfasis

La etiqueta más sencilla que permite definir un párrafo de texto es “**<p>**”. Dentro de “**<p>**” solamente debería haber texto, el cuál puede o no estar enfatizado en algún lugar.

```
1 <p>
2 El texto aquí presente compone un párrafo, el
3 cuál empieza en la marca de apertura y termina
4 en la marca de cierre.
5 </p>
6 <p>
7 Al abrir una nueva etiqueta, el contenido de
8 esta se encuentra en otro párrafo.
9 </p>
```

Cada párrafo se encuentra separado del siguiente por un pequeño espacio, cuyo tamaño inicial es delimitado por el browser. Si se requiere una mayor distancia entre un párrafo y el siguiente (u otros elementos que lo rodean) se debe utilizar CSS.

También se puede dar énfasis a palabras o frases enteras dentro de un párrafo. Para ello existen dos etiquetas, “**<em>**” que da un énfasis simple, y “**<strong>**”, el cuál da un énfasis doble. El énfasis simple suele ser mostrado por defecto en itálica, mientras que el doble suele ser mostrado en negrita. Nuevamente, este comportamiento puede variarse con CSS.

```
1 <p>
2 En este párrafo éste texto está siendo
3 enfatizado, y el que sigue también,
4 pero con un énfasis aún mayor
5 </p>
```

Si bien es posible colocar una etiqueta “**<strong>**” dentro de una etiqueta “**<em>**”, esto no es recomendado, ya que solo se obtiene un efecto visual, el cuál debería ser logrado con CSS, semánticamente la frase ya está siendo más relevante que otras al encontrarse con énfasis.

## Títulos

Para los títulos se usa el grupo de etiquetas “H”, el cuál abarca seis diferentes nivel, desde “**<h1>**” hasta “**<h6>**”. La etiqueta “**<h1>**” representa el título principal del documento, y por tanto no debería haber más de una en un mismo sitio. Etiquetas “**<h2>**” pueden ser utilizadas para subtítulos, o títulos de secciones o de artículos. Los “**<h3>**” pueden ser utilizados para subtítulos dentro de una sección o artículo. Así siguiendo el mismo criterio, se debería utilizar títulos para marcar aquellos textos que identifican partes importantes del sitio.

```

1 <h1>Título de primer nivel</h1>
2 <h2>Título de segundo nivel</h2>
3 <h3>Título de tercer nivel</h3>
4 <h4>Título de cuarto nivel</h4>
5 <h5>Título de quinto nivel</h5>
6 <h6>Título de sexto nivel</h6>
```

## Listas

HTML también soporta listas, tanto ordenadas (se muestran los elementos como una serie de elementos numerados) como no ordenadas (se muestran los elementos con viñetas). Para declarar una lista se debe “abrir la lista”, luego declarar los elementos como contenido de la lista, y finalmente “cerrar la lista”. Para abrir la lista se utiliza la etiqueta “`<ol>`” (si se trata de una lista ordenada) o “`<ul>`” (si es una lista no ordenada). Se utiliza la misma etiqueta para el cierre, como marca de cierre. Los elementos de la lista se declaran con la etiqueta “`<li>`”, la cuál se marca con apertura y cierre.

```

1 <h4>La siguiente es una lista ordenada</h4>
2
3 Primer elemento
4 Segundo elemento
5 Tercer elemento
6 Cuarto elemento
7
8
9 <h4>Y ahora una lista no ordenada</h4>
10
11 Primer elemento
12 Segundo elemento
13 Tercer elemento
14 Cuarto elemento
15
```

Dentro de la etiqueta “`<li>`” puede colocarse algo más que texto, por ejemplo, etiquetas de énfasis, e incluso algunas otras etiquetas, pero no debería contener párrafos enteros o títulos.

Dentro de las etiquetas “`<ol>`” y “`<ul>`” solo pueden haber etiquetas “`<li>`”, es decir, no puede haber un párrafo o título en medio de la lista de elementos. Tampoco pueden existir elementos “`<li>`” fuera de una estructura de lista.

## Tablas

Las tablas son una forma de generar información tabulada, o de doble entrada (fila y columna). Es una buena forma de mostrar datos que dependen de dos elementos, por ejemplo, países con algunas de sus propiedades geográficas o políticas, como muestra la tabla a continuación:

| País      | Población       | Superficie                | Densidad                  |
|-----------|-----------------|---------------------------|---------------------------|
| Argentina | 44.494.502 hab. | 2.780.400 km <sup>2</sup> | 16 hab/km <sup>2</sup>    |
| Uruguay   | 3.519.014 hab.  | 176.215 km <sup>2</sup>   | 19,97 hab/km <sup>2</sup> |
| Paraguay  | 7.152.703 hab.  | 406.752 km <sup>2</sup>   | 17,58 hab/km <sup>2</sup> |
| Bolivia   | 11.383.094 hab. | 1.098.58 km <sup>2</sup>  | 10,36 hab/km <sup>2</sup> |

Las tablas deben definirse con la etiqueta “`<table>`”, y dentro de ésta declarar las filas, y en cada fila, las celdas (las columnas terminan siendo declaradas en base a las celdas).

La tabla anterior puede ser generada con el siguiente código HTML:

```
1 <table>
2 <tr>
3 <td>País</td>
4 <td>Población</td>
5 <td>Superficie</td>
6 <td>Densidad</td>
7 </tr>
8 <tr>
9 <td>Argentina</td>
10 <td>44.494.502 hab.</td>
11 <td>2.780.400 km2</td>
12 <td>16 hab/km2</td>
13 </tr>
14 <tr>
15 <td>Uruguay</td>
16 <td>3.519.014 hab.</td>
17 <td>176.215 km2</td>
18 <td>19,97 hab/km2</td>
19 </tr>
20 <tr>
21 <td>Uruguay</td>
22 <td>7.152.703 hab.</td>
23 <td>406.752 km2</td>
24 <td>17,58 hab/km2</td>
25 </tr>
26 <tr>
27 <td>Uruguay</td>
28 <td>11.383.094 hab.</td>
29 <td>1.098.58 km2</td>
30 <td>10,36 hab/km2</td>
31 </tr>
32 </table>
```

Como se puede apreciar, la etiqueta “`<tr>`” define una nueva fila, y va ubicada dentro de la etiqueta “`<table>`”. Por otro lado, la etiqueta “`<td>`” se ubica dentro de la etiqueta “`<tr>`”, y define una celda de la tabla. Los estilos como bordes, dimensiones y otros detalles deben ser definidos utilizando CSS.

Otros casos de uso más complejos pueden requerir que un conjunto de celdas se encuentren agrupadas de alguna forma, algo que puede lograrse mediante el uso de diversos atributos en las etiquetas “`<tr>`” o “`<td>`”, dependiendo de cuál sea el agrupamiento deseado.

## Enlaces

El nombre de HTML significa en español “lenguaje de marcado de hipertexto”. El concepto de **hipertexto** remite a la idea de poder navegar el texto de una forma no lineal, mediante la utilización de enlaces, que pueden llevarnos delante y atrás en un conjunto de documentos.

Precisamente, uno de los elementos más importantes es entonces la capacidad de agregar

enlaces a otros documentos. Un enlace siempre debe componerse de dos partes fundamentales: el texto a mostrarle al lector y la dirección al cuál se desea enlazar.

Por esto, la etiqueta “`<a>`” (llamado así por “anchor”, ancla en inglés), utilizada para definir enlaces, requiere de forma obligatoria un atributo llamado “`href`” (de hiper-referencia). Mientras que el texto que se le mostrará al lector del documento al visualizarlo en un browser será el contenido de la etiqueta, el lugar al que enlaza será la dirección colocada como valor en el atributo “`href`”.

```
1 Ir a Google
```

Es importante destacar que el enlace puede ser una URL cualquiera, pero también pueden utilizarse rutas relativas para enlazar a otros documentos en la misma máquina (es decir, del mismo sitio web).

```
1
2 Ver las noticias de Mayo del 2015
3
```

## Imágenes

Las imágenes en HTML están dadas por la etiqueta “`<img>`”, una de las etiquetas más extrañas de HTML, pues tiene la particularidad de requerir únicamente su marca de apertura, no teniendo que ser cerrada. Es decir, es una etiqueta que no posee contenido alguno.

Lleva sin embargo, de forma obligatoria, dos atributos. El primero es `src`, el cuál espera como valor una URL a la ubicación de la imagen (también podría ser la ruta a un archivo local). El otro atributo es `alt`, que espera como valor un texto, el cual se muestra en caso de que la imagen no pueda ser cargada, o cuando el usuario deja el puntero del mouse sobre la imagen. Este texto consiste en la “descripción” de la imagen, y también puede ser utilizado por visualizadores específicos para no videntes como información adicional.

```
1
```

El tamaño de la imagen se debe determinar por CSS.

## Audio y Video

Las etiquetas “`audio`” y “`video`” se introdujeron en HTML5 con la intención de soportar contenido multimedia avanzado (algo que antes de esta versión solo podía lograrse mediante complementos externos del navegador, como Adobe Flash o Microsoft Silverlight). Las etiquetas mencionadas pueden incluir un atributo `controls`, que no lleva valor alguno, y que le indica al navegador que debe mostrarle los controles de reproducción al usuario.

El contenido de esas etiquetas debe ser una o más etiquetas “`<source>`”, las cuales mediante el atributo `src` para indicar la ubicación del archivo, y el atributo `type` para indicar el formato en que

se encuentra (En los sitios web, muchos archivos pueden publicarse sin sus extensiones, por lo que indicar el formato resulta importante).

```
1 <audio controls>
2 <source src="caballo.ogg" type="audio/ogg">
3 <source src="caballo.mp3" type="audio/mpeg">
4 </audio>
5
6 <video controls>
7 <source src="caballo.ogg" type="video/ogg">
8 <source src="caballo.mp4" type="video/mp4">
9 </video>
```

Se debe tener en cuenta que estas etiquetas solamente son soportadas en navegadores web modernos, y no se visualizarán correctamente en máquinas antiguas no actualizadas.

## Contenido embebido

Una etiqueta también interesante y muy utilizada es “**iframe**”. Ésta permite incrustar contenido de otro sitio web, y hacer que el mismo se encuentre disponible en nuestro documento. Un ejemplo muy común es el caso de los videos de plataformas como YouTube, los cuales se incrustan utilizando esta etiqueta.

En general uno no suele incluir el contenido de otros sitios, y cuando lo hace, utiliza una porción de código predefinida, con atributos especiales ya completados por quien provee el código.

```
1 <iframe
2 width="560"
3 height="315"
4 src="https://www.youtube.com/embed/dQw4w9WgXcQ"
5 frameborder="0"
6 allow="accelerometer; autoplay; encrypted-media; gyroscope;
7 picture-in-picture"
8 allowfullscreen>
9 </iframe>
```

## Etiquetas semánticas de HTML

HTML define todo un conjunto de etiquetas que permiten agrupar contenido de forma semántica. Así, por ejemplo, un título, una serie de párrafos e imágenes que pertenezcan a una misma agrupación conceptual pueden ser marcadas e identificadas como tales.

Estas etiquetas no tienen una contraparte visual, y solamente permiten definir bloques lógicos. Sin embargo, herramientas de procesamiento automático de datos, como los “robots” de Google o Bing utilizan este agrupamiento para “comprender” la información del documento y brindar resultados de búsqueda de calidad para los usuarios (Y un diseñador siempre quiere que su sitio sea bien entendido por estas empresas para aparecer en los primeros lugares de los resultados de búsqueda).

Las agrupaciones conceptuales, si bien suman un nivel semántico no visual, pueden tener una contrapartida visual si se utiliza CSS para darles estilo.

Dentro de las etiquetas de agrupamiento encontramos:

- **<header>**, utilizada para la cabecera de un sitio (por ejemplo, la parte que contiene el título principal del sitio, el logo, algunos posibles datos adicionales y tal vez algún menú de navegación principal).
- **<main>**, utilizada para agrupar todo el contenido principal del sitio, es decir, la parte relevante que interesa leer a los usuarios.
- **<footer>**, utilizada para definir el pie de página (una sección al final del sitio que suele contener datos como el copyright, enlaces a redes sociales, enlaces al mapa del sitio, y otros datos del dueño del sitio o del mismo sitio).
- **<section>**, utilizada para agrupar lógicamente secciones grandes del sitio que se muestran en un mismo documento. Cada sección tiene que contener de forma obligatoria un título, de “**<h2>**” en adelante.
- **<article>**, utilizada para agrupar contenido que lógicamente conforma un artículo, y que podría leerse de forma independiente del resto del contenido, y aún así, tener sentido.
- **<nav>**, utilizada para definir barras de navegación, en general con enlaces a uno o más documentos del sitio.
- **<aside>**, utilizada para agrupar elementos que no refieren a una parte importante del sitio. Por ejemplo, publicidad, o elementos de barras laterales suelen agruparse bajo ésta etiqueta.
- **<div>**, utilizada para agrupar elementos que no entran semánticamente en ninguna de las etiquetas anteriores.

Cada una de estas etiquetas abre cuando se comienza a agrupar y cierra cuando se deja de agrupar. A continuación se da una muestra de dos secciones de un sitio, que a su vez contienen artículos.

```
1 <section>
2 <h2>Noticias deportivas</h2>
3
4 <article>
5 <h3>La preocupación de los argentinos</h3>
6 <p>
7 Todos los argentinos siguen preocupados
8 por la indecisión de los expertos sobre
9 quién es el mejor jugador de fútbol del
10 mundo. Maradona y
11 Messi son dos de los
12 candidatos más votados.
13 </p>
14 </article>
15 <article>
16 <h3>Se puede jugar al basquet con 1.50?</h3>
17 <p>
18 El basquet siempre ha sido un deporte
19 dominado por personas de gran altura,
20 pero una nueva ola de jugadores de baja
21 estatura y alta performance están dando
22 muestra de que la altura no es un
23 condicionante a la hora de jugar.
24 </p>
25 </article>
26 </section>
27 <section>
```

```

28 <h2>Noticias del espectáculo</h2>
29
30 <article>
31 <h3>Mar del Plata o Villa Carlos Paz</h3>
32 <p>
33 Cuáles son las obras de teatro que
34 podemos encontrar este verano en los
35 puntos turísticos?
36 </p>
37 </article>
38 <article>
39 <h3>Una nueva de Ricardo Darín</h3>
40 <p>
41 El actor nos cuenta de su nuevo
42 trabajo, una película que no sabemos
43 de que se trata, pero que seguro es
44 buena, porque actúa Darín, y si es
45 mala, no importa, porque actúa Darín.
46 </p>
47 </article>
48 </section>

```

## B.5 Documentos HTML válidos

Hasta ahora vimos la forma en la que las etiquetas se escriben y pueden anidarse, y también vimos algunas etiquetas básicas de HTML. Sin embargo, un documento HTML no consiste en etiquetas sueltas en el texto, sino que debe contar con una estructura protocolar obligatoria.

Todo documento HTML debe comenzar con una pseudo-etiqueta, llamada “declaración del doctype”. Esta declaración es leída por el navegador web previo a procesar cualquier elemento del documento, y le indica al browser qué es lo que debe esperar del mismo. Nosotros escribiremos lo siguiente:

```
1 <!DOCTYPE html>
```

Dicha sentencia le indica al browser que el documento se trata de un archivo HTML en su versión 5. Sentencias similares para HTML en versiones anteriores también existen, así como otros estándares como XHTML o XML, que también pueden ser comprendidos por el navegador, pero trabajados de forma distinta por el mismo.

Luego, puede comenzar el contenido del sitio, el cual debe estar siempre dentro de una etiqueta que engloba a la totalidad de los elementos, y que se llama “**html**”. Dicha etiqueta no puede contener cualquier cosa, sino que solamente dos etiquetas pueden estar dentro de ella, las etiquetas “**head**” y la etiqueta “**body**”. Es decir, la declaración de un documento debería ir quedando algo como:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 </head>

```

```

5 <body>
6 </body>
7 </html>
```

No debería haber ninguna otra etiqueta fuera de html, ni directamente bajo esta, sino que toda otra etiqueta debería estar dentro de las etiquetas “<head>” o “<body>”.

La etiqueta “<head>” sirve para darle información al navegador sobre características adicionales del documento. Estos datos son utilizados de diferentes formas por el navegador, y también por los robots de Google o Bing. En particular hay dos etiquetas importantes a colocar dentro de “<head>”, la etiqueta “<title>” que indica el título del sitio (el navegador suele mostrarlo como título de la ventana o pestaña) y la etiqueta “<meta>” que carga datos adicionales al navegador, pudiendo haber más de una etiqueta meta en el head.

Cada etiqueta meta define, mediante atributos, información adicional para el navegador. Uno de los atributos más importantes es el “charset”, el cuál indica al browser que vamos a utilizar símbolos “especiales” (es decir, caracteres que no son del idioma inglés, como acentos o la letra eñe). Para esto, el atributo “charset” debe tener el valor “utf-8”.

También es recomendable agregar como atributo el idioma en el cuál va a estar escrito nuestro documento. Para ello se utiliza el atributo “lang”, el cuál se coloca directamente sobre la etiqueta “<html>”. Para español se debe utilizar el valor “es”.

Adicionalmente podemos agregar etiquetas meta con información como el autor del sitio, descripción, palabras clave, e incluso información sobre como debería mostrarse el documento en un dispositivo. Para ello, la etiqueta “<meta>” soporta los “name” y “content”, que deben llenarse con valores específicos en cada ocasión. Nuestro documento HTML comienza a verse así:

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4 <title>Título del sitio</title>
5 <meta charset="utf-8">
6 <meta name="author" content="Autor del sitio">
7 <meta name="description"
8 content="Un sitio de prueba para los que se inician en
9 HTML">
10 <meta name="keywords"
11 content="HTML, Web, CSS, JavaScript">
12 </head>
13 <body>
14 </body>
15 </html>
```

Ahora si podemos centrarnos en el contenido, el cuál estará declarado en la etiqueta “<body>”, la cuál contendrá todos los elementos visibles del sitio. Por ejemplo, un *header* con el título principal del sitio, una etiqueta main con el contenido, y un *footer* con información de copyright.

A continuación se muestra un ejemplo de una página HTML básica.

```

1 <!DOCTYPE html>
2 <html lang="es">
```



```
59 de que se trata, pero que seguro es
60 buena, porque actúa Darín, y si es
61 mala, no importa, porque actúa Darín.
62 </p>
63 </article>
64 </section>
65 </main>
66 <footer>
67 <p>
68 Copyright© 2019 - Juan de los Palotes
69 </p>
70 </footer>
71 </body>
72 </html>
```

Por supuesto que “<body>” admite cualquier combinación compleja de etiquetas de contenido, y no solo la aquí expuesta. El contenido dependerá, en última instancia, de lo que se desea mostrar.

Para saber si un documento es o no es válido, la W3C, encargada de regular el estándar de HTML, provee a los desarrolladores una página web que permite validar sus archivos, para detectar si estos efectivamente cumplen con el estándar. La dirección es:

<https://validator.w3.org/nu/#file>

Hay que tener en cuenta que el validador de la W3C solamente analiza si las etiquetas utilizadas son correctas, y si se cumple con todos los campos protocolares, así como si las etiquetas están anidadas correctamente. No valida, por otro lado, cuestiones de estilo, como la indentación, ni el uso correctamente semántico de las etiquetas. Por esto es que el validador es útil, pues nos brinda una idea clara de si nuestro documento cumple al menos con una estructura correcta, y por tanto, va a ser procesado correctamente por los navegadores, pero no nos asegura que el código sea “bueno”.

## B.6 CSS

Es interesante comprender que en general, un sitio web no se compone únicamente de código HTML, sino que es un conjunto de archivos, que incluye tanto a uno como más archivos HTML, como también a las imágenes del sitio, iconos, fuentes, estilos, y código para interactividad local. Es decir, cuando uno realiza un sitio web, no realiza un archivo, sino un directorio lleno de archivos.

Las rutas de los archivos HTML siempre deberían ser rutas relativas. Esto permite que el directorio que involucra el proyecto, y que contiene a todos los archivos, pueda ser copiado de una máquina a otra, e incluso subido a un servidor para ser compartido.

Uno de estos tipos de archivo resulta particularmente interesante, el CSS. Los estilos por defecto de HTML resultan sumamente básicos, y prácticamente no existe sitio hoy en día que no tenga CSS (Aunque si puede carecer de otros) El código CSS consiste en una serie de reglas que se le brindan a los elementos HTML. Estas reglas incluyen detalles como:

- El tamaño de fuente.
- El estilo del texto (italica, negrita, subrayado, etc).
- El color del texto y del fondo de los elementos.
- Bordes, con sus tamaños, colores, estilo, redondeo de bordes, etc.
- Tamaños de las imágenes, videos, reproductores de audio, etc.
- Imágenes de fondo y transparencias.
- Ubicación de los elementos, tamaños y formas.

- Distancia del elemento a otros elementos y a su contenido interno.

En las últimas versiones de CSS se pueden definir incluso cosas como animaciones, transformaciones de contenido (Por ejemplo, poner el texto en mayúsculas), e incluso estilos que se aplican únicamente en ciertas situaciones (por ejemplo, si ve el contenido en un celular, o si se va a imprimir el sitio).

Aprender CSS y todas sus particularidades es un trabajo complejo, y hay gente que se especializa en esta área, conocidos como “Diseñadores web”. Un diseñador web se diferencia de un programador. El primero nada sabe de lenguajes de programación, y solamente se enfoca en HTML y CSS. El segundo debe tener una idea de HTML y CSS, pero no es su fuerte, ni debe conocerlos en profundidad.

El código CSS consiste en una serie de “reglas”, las cuales se aplican una tras otra sobre el los elementos del documento. El navegador web aplica las reglas, y tras hacerlo, muestra el resultado. Cada regla consiste en un “selector” (el o los elementos sobre el cuál se aplicará la regla) y una serie de “propiedades” (que son las cualidades a dar al elemento sobre el cuál se aplica la regla), cada una con su correspondiente “valor”.

```
1 selector {
2 propiedad1: valor1;
3 propiedad2: valor2;
4 propiedad3: valor3;
5 }
```

Cada propiedad se separa de las que le siguen mediante un punto y coma, que no debe olvidarse. El objetivo aparece al comienzo, y las propiedades que se aplicarán irán entre llaves.

Como selector, puede colocarse el nombre de cualquier etiqueta HTML. Las propiedades que acepta una regla dependen del tipo de elemento a dar estilo. Por ejemplo, el siguiente código coloca el color de texto de todos los títulos h2 en rojo.

```
1 h2 {
2 color: red;
3 }
```

También pueden definirse más de un selector mediante la separación de los mismos utilizando comas. Por ejemplo, el siguiente código pone en rojo sobre fondo azul tanto los títulos de nivel dos, como los de nivel tres y cuatro.

```
1 h2, h3, h4 {
2 color: red;
3 background-color: blue;
4 }
```

Existen formas de apuntar a un elemento específico del documento HTML, en lugar de “a todos los elementos h1”, así como de apuntar a un grupo de elementos particular (por ejemplo, el grupo de elementos que deberían tener texto en rojo). Para ello, el HTML debe cumplir ciertos requisitos, agregando atributos a los elementos a dar estilo (Los atributos “class” y “id”). También existen

formas de apuntar a los conocidos como pseudo-elementos (por ejemplo, los enlaces que tienen el cursor del mouse sobre ellos), dándoles estilo.

El código CSS se incluye en la etiqueta “`<head>`”, pues si bien tiene resultados visibles, no involucran contenido a disponer en la página. Así, dentro de “`<head>`” se puede agregar código CSS utilizando la etiqueta “`<style>`”, que abre cuando inicia el código CSS y cierra cuando termina.

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4 <title>...</title>
5 <style>
6 h2, h3, h4 {
7 color: red;
8 background-color: blue;
9 }
10 </style>
11 </head>
12 <body>
13 ...
```

También es posible colocar el estilo en un archivo externo, y adjuntarlo al documento HTML mediante la etiqueta “`<link>`”, algo que no trataremos aquí. Versiones anteriores de HTML permitían incluir estilos directamente en el HTML, como atributos del elemento a dar estilo. Esta práctica se encuentra hoy en día obsoleta, y se desalienta su uso en HTML5, pues se removerá la posibilidad de hacerlo en el futuro.

## B.7 Formularios y respuestas del servidor

Si ha navegado por Internet, habrá notado sitios como Gmail, Facebook o Twitter, que solicitan al usuario que suministre un correo electrónico y contraseña, previo a dejarlo visualizar la página principal. ¿Cómo hace HTML para validar que el usuario y la contraseña son correctos? La respuesta es, no lo hace.

Un servidor especializado, encargado de validar la información recibe los datos que el usuario ingresó en el HTML y luego determina si la contraseña coincide con los registros de la empresa o no, mostrándole al usuario el contenido que solicitaba, o indicándole que verifique sus credenciales y vuelva a intentarlo.

HTML provee funcionalidad para realizar este tipo de acciones, es decir, enviar información a un servidor. Para ello, utiliza una etiqueta llamada “`<form>`”, y etiquetas que van dentro de esta, como “`<input>`”, “`<select>`”, “`<textarea>`” y “`<button>`”.

Una etiqueta “`<form>`” debe contener dos atributos, “method” y “action”. El atributo “method” puede tener un valor de “GET” o de “POST”, y le indica al navegador la forma en la que los datos serán enviados al servidor (El servidor deberá estar configurado para esperar lo mismo que el navegador envía). La etiqueta “action”, por su parte, espera como valor una URL, que es la dirección a donde se enviarán los datos.

Todo formulario debe contar con alguna forma en que el usuario confirme los valores ingresados, y mande la información al servidor de forma efectiva. Esta acción se conoce como “submit”, y suele ser un botón el que provee tal característica.

Los botones se crean con la etiqueta “<button>”, y muestran como texto del mismo, lo que tengan en su contenido. Además, suelen llevar un atributo llamado “type” que indica qué es lo que se espera que haga el botón. Existen dos tipos que el browser trata de forma especial: “submit” y “reset”. Si el usuario da clic sobre un botón cuyo “type” es “submit”, la información cargada en el formulario será enviada al servidor. Si da clic sobre un botón con “type” igual a “reset”, limpiará toda la información cargada al momento en el formulario, dejándolo en blanco.

Existen diversos elementos que pueden aparecer en un formulario. La mayoría de ellos se ven representados en la etiqueta “<input>”, que posee dos atributos obligatorios. El primero “type” indica que tipo de input se trata, pudiendo ser:

- **text** Un campo de texto genérico
- **password** Un campo de texto, pero donde el contenido no se muestra o aparece como puntos o asteriscos.
- **checkbox** Un campo de marca de selección, en donde el usuario puede marcar, o desmarcar la opción.
- **radio** Un campo de selección entre varias opciones. Si el usuario marca una opción, las otras se desmarcan.
- **hidden** Un campo que no se muestra en el formulario, y que puede ser utilizado para datos de control.
- **file** Un campo en el que el usuario puede seleccionar un archivo de su máquina para la carga.

Desde HTML5, tipos de input con valores más semánticos se han creado, lo que permite una mayor diferenciación sobre los datos que se están manipulando, dando lugar por ejemplo, a herramientas capaces de completar automáticamente formularios enteros. Así “type” también puede ser:

- **color** Permite al usuario elegir un color mediante un selector de colores.
- **date** El usuario puede ingresar una fecha, con un selector de fechas estilo calendario (sin ingresar hora).
- **email** Un campo de texto, pero donde se espera que el usuario agregue un correo electrónico.
- **number** Un campo de texto, pero donde se espera que el usuario ingrese un número.
- **tel** Un campo de texto, pero donde se espera que el usuario ingrese un número de teléfono.
- **url** Un campo de texto en donde se espera una dirección URL válida.

También existen otras opciones, para horas, meses, rangos de elementos, campos de búsqueda, etc.

El otro atributo importante de los input es “name”, el nombre. Este atributo es la forma de identificar al valor de forma inequívoca por parte del servidor. Por tanto, cada input debe tener un valor distinto. El valor consiste en un nombre, que puede estar formado por letras, números, guiones bajos y guiones medios, pero que no puede contener espacios.

Un caso particular es el de los inputs de tipo “radio”. En este caso, se espera que el nombre sea el mismo para aquellas opciones que conformen un conjunto (es decir, las opciones tales que si el usuario selecciona una, las otras se des-seleccionan).

A continuación hay un ejemplo de un pequeño formulario que solicita al visitante su correo electrónico y contraseña.

```

1 <form action="servidor/procesar_ingreso" method="POST">
2 <label> Correo electrónico:
3 <input type="email" name="correo_de_usuario">
4 </label>
```

```
5 <label> Contraseña:
6 <input type="password" name="password_de_usuario">
7 </label>
8 <button type="submit">
9 Ingresar
10 </button>
11 </form>
```

Esto resulta en algo como lo siguiente:

Correo electrónico:  Contraseña:  Ingresar

Note el uso de la etiqueta “<label>”. Esta se utiliza para englobar a los elementos “<input>”, permitiendo mostrar un texto asociado a los mismos que el visitante verá en el sitio, dandole indicaciones sobre qué es lo que se espera que ingrese. Pueden usarse en su lugar párrafos de texto simple, pero “<label>” brinda una ventaja adicional de usabilidad, cuando el usuario da clic sobre el texto, el “<input>” se selecciona, y el usuario ya puede comenzar a escribir, esto no ocurre con un párrafo. En particular es una característica muy útil cuando se trabaja con elementos de tipo “radio” o “checkbox”.

Las etiquetas “<input>” son muy útiles, pero en ocasiones no son suficientes. Si se desea por ejemplo, que el usuario ingrese texto muy largo, se debe recurrir a la etiqueta “<textarea>”. Esta etiqueta presenta un cuadro de texto de múltiples líneas, el cual se agranda automáticamente a medida que el usuario escribe. Al igual que los “<input>”, espera un atributo “name” para poder ser procesado, pero no lleva “type”.

También puede que se desee elegir de una serie de opciones de una lista, algo que se logra con la etiqueta “<select>” (también espera un “name” como atributo), la cual lleva dentro una o más etiquetas “<option>”. Esta etiqueta muestra una lista desplegable, en donde el usuario puede seleccionar una de las opciones (dadas cada una por un “<option>” distinto).

A continuación hay un formulario que muestra el uso de etiquetas “<textarea>” y “<select>”. También presenta un “checkbox” y un “radio” El formulario permite al usuario ingresar un reclamo en el sitio de una empresa proveedora de internet.

```
1 <form action="servidor/procesar_ingreso" method="POST">
2 <div>
3 <label for="nombre">Su nombre:</label>
4 <input type="text" name="nombre" id="nombre">
5 </div>
6
7 <div>
8 <label for="motivo">Motivo del reclamo:</label>
9 <select name="motivo" id="motivo">
10 <option>No tengo conexión a internet</option>
11 <option>Mi conexión anda lenta</option>
12 <option>Mi conexión es intermitente</option>
13 <option>Otro tipo de reclamo</option>
14 </select>
15 </div>
16
17 <div>
18 <label for="problema">Cuéntenos más sobre el problema:</label>
```

```
19 <textarea name="problema" id="problema"></textarea>
20 </div>
21
22 <div>
23 <input type="checkbox" name="reclamos_previos"
24 id="reclamos_previos">
25 <label for="reclamos_previos">¿Ya ha realizado reclamos en el
26 pasado?</label>
27
28 <p>¿Desea que nos comuniquemos con usted?</p>
29 <div>
30 <input type="radio" name="comunicacion" id="comunicacion1">
31 <label for="comunicacion1"></label>Comunicarse por correo
32 electrónico</label>
33
34 <div>
35 <input type="radio" name="comunicacion" id="comunicacion2">
36 <label for="comunicacion2"></label>Comunicarse por
37 teléfono</label>
38
39 <div>
40 <input type="radio" name="comunicacion" id="comunicacion3">
41 <label for="comunicacion3">No comunicarse</label>
42
43 <div>
44 <button type="submit">
45 Enviar Reclamo
46 </button>
47 <button type="reset">
48 Limpiar el formulario
49 </button>
50 </div>
51 </form>
```

Note como en este ejemplo las etiquetas “`<label>`” no envuelven al campo que denotan, sino que se usa otra técnica, la cual consiste en utilizar un atributo “`for`” en “`<label>`” cuyo valor coincide con el valor del atributo “`id`” del “`<input>`” que se encuentra marcando.

También se envuelve cada conjunto de “`<label>`” y “`<input>`” en una etiqueta “`<div>`” que los agrupa semánticamente. Esto también logra el efecto de mostrar un elemento debajo de otro en el formulario, ya que por defecto, los “`<div>`” se muestran uno bajo el otro, mientras que los “`<label>`” y “`<input>`” se muestran todos en el mismo renglón.

El formulario anterior se vería algo así:

Su nombre:

Motivo del reclamo:  No tengo conexión a internet

Cuéntenos más sobre el problema:

¿Ya ha realizado reclamos en el pasado?

¿Desea que nos comuniquemos con usted?

- Comunicarse por correo electrónico
- Comunicarse por teléfono
- No comunicarse

Los formularios pueden embellecerse con CSS. Existen incluso conjuntos de código pre-hechos por terceros que uno puede utilizar para embellecer sus formularios (y sus sitios en general). Estos códigos se agrupan en lo que se llama “biblioteca”. Con un poco de CSS el formulario anterior puede verse de la siguiente forma:

Su nombre:

Motivo del reclamo:  No tengo conexión a internet

Cuéntenos más sobre el problema:

¿Ya ha realizado reclamos en el pasado?

¿Desea que nos comuniquemos con usted?

- Comunicarse por correo electrónico
- Comunicarse por teléfono
- No comunicarse

Existen también otros atributos y particularidades que hacen a los formularios, así como al procesamiento de los mismos. Estos detalles no son parte del presente, y se recomienda al lector retomar estos temas cuando posea un conocimiento más amplio de programación y de redes de computadoras.

## B.8 Actividades

Los siguientes ejercicios se realizar de forma continuada, es decir, para realizar el ejercicio 2 deberá realizar el ejercicio 1 previamente. Recomendamos hacerlos en orden y no saltarse ninguno.

**Ejercicio B.1** Cree un documento HTML un sitio web que actuará como portal de noticias. Recuerde de que el documento debe tener todas las partes protocolares necesarias. Luego agregue los siguientes elementos a su sitio:

- a) Un título principal con el texto “Últimas Noticias”.
- b) Una primer sección bajo el título “Noticias deportivas”.
- c) Una segunda sección bajo el título “Noticias del espectáculo”.
- d) Un párrafo descriptivo en cada sección indicando el tipo de noticias que el lector podrá encontrar.

El resultado debería verse algo como:

# Últimas Noticias

## Noticias Deportivas

Toda la información del fútbol, basquet, y mucho más.

## Noticias del Espectáculo

Los famosos, la téle, el teatro, y todo el chisme.

**Ejercicio B.2** Agregue dos artículos de noticias a cada sección del sitio anterior. Puede buscar noticias en un medio de su preferencia, o inventarlas. Cada noticia debe contar con un título (de menor jerarquía que los de las secciones), y un copete (resumen de la noticia) en donde aparezcan algunas palabras o frases clave con énfasis. El resultado debería ser similar al siguiente:

# Últimas Noticias

## Noticias Deportivas

Toda la información del fútbol, basquet, y mucho más.

### La preocupación de los argentinos

Todos los argentinos siguen preocupados por la indecisión de los expertos sobre quién es el mejor jugador de futbol del mundo. **Maradona** y **Messi** son dos de los candidatos más votados.

### Se puede jugar al básquet con 1.50?

El básquet siempre ha sido un deporte dominado por *personas de gran altura*, pero una nueva óla de jugadores de **baja estatura** y **alta performance** están dando muestra de que la estatura no es un condicionante a la hora de jugar.

## Noticias del Espectáculo

Los famosos, la téle, el teatro, y todo el chisme.

### Mar del Plata o Villa Carlos Paz

¿Cuáles son las obras de teatro que podemos encontrar este verano en los *puntos turísticos*?

### Una nueva de Ricardo Darín

El actor nos cuenta de su nuevo trabajo, una película que no sabemos de que se trata, pero que seguro es buena, porque **actúa Darín**, y si es mala, no importa, porque **actúa Darín**.

**Ejercicio B.3** Añada el link de referencia a los sitios de donde tomo las noticias (o coloque un enlace a sitios de su preferencia si las inventó). Los enlaces deben leerse con el texto “Leer más”. El resultado debería ser similar a este:

## Últimas Noticias

### Noticias Deportivas

Toda la información del fútbol, basquet, y mucho más.

#### La preocupación de los argentinos

Todos los argentinos siguen preocupados por la indecisión de los expertos sobre quién es el mejor jugador de futbol del mundo. **Maradona** y **Messi** son dos de los candidatos más votados.

#### Se puede jugar al básquet con 1.50?

El básquet siempre ha sido un deporte dominado por *personas de gran altura*, pero una nueva ola de jugadores de **baja estatura y alta performance** están dando muestra de que la estatura no es un condicionante a la hora de jugar.

### Noticias del Espectáculo

Los famosos, la téléc, el teatro, y todo el chisme.

#### Mar del Plata o Villa Carlos Paz

¿Cuáles son las obras de teatro que podemos encontrar este verano en los *puntos turísticos*?

#### Una nueva de Ricardo Darín

El actor nos cuenta de su nuevo trabajo, una película que no sabemos de que se trata, pero que seguro es buena, porque **actúa Darín**, y si es mala, no importa, porque **actúa Darín**.

**Ejercicio B.4** Sume ahora, antes del título del diario, en una cabecera, una imagen que actúe de logo del sitio. Puede tomar de internet una imagen que guste como logo. Incluya también un pie de página con la información de copyright del sitio. La misma debe contener el nombre del autor y el año en que realizó el sitio. Tenga en cuenta que lo que antes eran posiblemente simples secciones deberían ser ahora el contenido principal del sitio.

**Ejercicio B.5** Si realizó los ejercicios hasta ahora correctamente (es decir, ha colocado todas las etiquetas de estructura semánticas que corresponde) debería ser posible que su sitio luzca como la siguiente imagen:

The screenshot shows a website layout with a header 'UltimasNoticias'. Below it are two main content areas:

- Noticias Deportivas**
  - Toda la información del fútbol, basquet, y mucho más.
  - La preocupación de los argentinos**
    - Todos los argentinos siguen preocupados por la indecisión de los expertos sobre quién es el mejor jugador de futbol del mundo. **Maradona** y **Messi** son dos de los candidatos más votados.
    - [Leer Más](#)
  - Se puede jugar al básquet con 1.50?**
    - El básquet siempre ha sido un deporte dominado por *personas de gran altura*, pero una nueva ola de jugadores de **baja estatura y alta performance** están dando muestra de que la estatura no es un condicionante a la hora de jugar.
    - [Leer Más](#)
- Noticias del Espectáculo**
  - Los famosos, la téléc, el teatro, y todo el chisme.
  - Mar del Plata o Villa Carlos Paz**
    - ¿Cuáles son las obras de teatro que podemos encontrar este verano en los *puntos turísticos*?
    - [Leer Más](#)
  - Una nueva de Ricardo Darín**
    - El actor nos cuenta de su nuevo trabajo, una película que no sabemos de que se trata, pero que seguro es buena, porque **actúa Darín**, y si es mala, no importa, porque **actúa Darín**.
    - [Leer Más](#)

At the bottom of the page, there is a footer bar with the text 'Copyright© Juan de los Palotes - 2019'.

Para ello necesitaremos agregar en el “<head>” del sitio la siguiente definición de CSS dentro de una etiqueta “<style>”:

```
1 header, footer {
2 background-color: #222;
3 color: lightgray;
4 }
5 footer {
6 height: 40px;
7 }
8 header {
9 height: 85px;
10 }
11 header, footer p {
12 padding: 10px;
13 }
14 header h1, header h2, header h3,
15 header h4, header h5, header h6 {
16 display: none;
17 }
18 header img {
19 height: 35px;
20 width: auto;
21 }
22 header nav {
23 width: 100%;
24 background-color: #222;
25 margin-bottom: 20px;
26 }
27 header li {
28 display: inline-block;
29 padding: 10px;
30 }
31 header li:hover {
32 background-color: #555;
33 cursor: pointer;
34 }
35 header a {
36 color: white;
37 text-decoration: none;
38 }
39 main {
40 margin-top: 10px;
41 display: flex;
42 flex-direction: row;
43 }
44 section {
45 justify-content: space-between;
46 width: 40%;
47 margin-left: 5%;
48 border: 3px solid rgb(150, 25, 25);
49 border-radius: 10px;
50 padding: 20px;
```

```
51 }
52 section>h1, section>h2, section>h3,
53 section>h4, section>h5, section>h6,
54 section>p {
55 color: rgb(150, 25, 25);
56 }
57 article p {
58 text-align: justify;
59 }
60 article a {
61 padding: 10px 20px;
62 background-color: rgb(150, 25, 25);
63 border-radius: 5px;
64 color: white;
65 text-decoration: none;
66 }
67 article a:hover {
68 background-color: rgb(200, 25, 25);
69 text-decoration: none;
70 }
71 aside {
72 text-align: center;
73 background-color: cornflowerblue;
74 color: darkred;
75 width: 50%;
76 margin: 0 auto;
77 font-weight: bolder;
78 }
79 aside h1, aside h2, aside h3, aside h4,
80 aside h5, aside h6 {
81 animation: blinker 2s linear infinite;
82 }
83 @keyframes blinker {
84 50% { opacity: 0; }
85 }
```

**Ejercicio B.6** Agregue en la cabecera una barra de navegación, que contenga una lista no ordenada de enlaces a las secciones del sitio. Las mismas son: Política, Policiales, Deportes y Espectáculos. El resultado debe ser como el siguiente:

The screenshot shows a news website template with a dark header bar containing the logo 'ÚltimasNoticias' and navigation links for 'Política', 'Policiales', 'Deportes', and 'Espectáculos'. Below the header are two main content boxes. The left box is titled 'Noticias Deportivas' and contains a paragraph about Argentina's concern over their footballer's height, followed by a section on basketball with a height requirement of 1.50 meters. The right box is titled 'Noticias del Espectáculo' and discusses theater performances, featuring a section on Ricardo Darín. Each news item includes a 'Leer Más' button.

**Ejercicio B.7** Agregue justo sobre el pie de página una publicidad. Coloque en ella un título o dos títulos de nivel dos en adelante, y un párrafo de texto. El resultado se verá algo así:

The screenshot shows the same news website template as before, but with a blue advertisement banner at the bottom. The banner has a title 'Don barredora' and a subtitle '¡Me llama usted, entonces voy, don barredora es quien yo soy!'. It also includes the code 'KLS-3226' at the bottom. The rest of the page structure remains the same, with the news columns and footer.

**Ejercicio B.8** Valide el código que ha generado con el validador oficial de la W3C. Si el código pasa exitosamente las pruebas, significa que el código no viola los estándares (aunque no significa que siga buenas prácticas de estilo, por lo que no necesariamente es un buen código).

De encontrarse con errores al momento de validar, corrija su código hasta que este sea válido. Puede encontrar el validador en:

<https://validator.w3.org/nu/#file>

**Ejercicio B.9** Se pide que diseñe una página web que contenga un formulario web capaz de cargar ordenes médicas para pacientes que requieren reposo laboral. El formulario debe contener los siguientes campos:

- Un campo de texto para el nombre del médico que firma la orden.
- Un campo de texto para el número de matrícula del médico que firma la orden.
- Un campo de selección radial que permita elegir si el número de matrícula corresponde a la matrícula nacional o provincial.
- Un campo de texto para el nombre del paciente.
- Un campo de selección que permita elegir la cantidad de horas de reposo. Deben haber 4

- opciones: “24 horas”, “48 horas”, “72 horas”, “hasta alta médica”.
- Un campo de selección (checkbox) que se pueda marcar si el paciente tiene obra social, y desmarcar en caso negativo.
  - Labels para cada uno de los campos.
  - Un botón para enviar el formulario.

El formulario deberá utilizar el método GET, y su acción deberá indicar como valor “#” (solo el signo numeral).

Puede utilizar el siguiente código CSS:

```
1 label {
2 display: block;
3 margin: 5px 0;
4 }
```

Esto logrará que los elementos aparezcan como en la imagen a continuación:

## Carga de datos para orden médica

Nombre del médico

Matrícula del médico

Matricula Nacional

Matricula Provincial

Nombre del Paciente

Cantidad de horas de reposo

El paciente tiene obra social

**Ejercicio B.10** Diseñe una página web que contenga un título principal con el texto “Mundiales de la FIFA”, y luego una tabla de doble entrada, donde las filas corresponden cada una a un mundial distinto. Deben haber tres columnas, la primera indicando en qué lugar del mundo se realizó el evento, y la segunda y tercera columna indicando el campeón y subcampeón respectivamente. Incluya al menos los mundiales de los siguientes años:

- 2002
- 2006
- 2010
- 2014
- 2018

Puede utilizar el siguiente código CSS para que la tabla quede más fácil de visualizar:

```
1 table {
```

```
2 border: 3px solid rgb(150, 25, 25);
3 border-collapse: collapse;
4 border-spacing: 0;
5 }
6 table tr:nth-child(odd) {
7 background-color: rgb(255, 175, 175);
8 }
9 table tr:nth-child(1) {
10 background-color: rgb(150, 25, 25);
11 color: white;
12 }
13 table td {
14 padding: 3px 5px;
15 }
```

El resultado de la página debería ser el siguiente.

## Mundiales de la FIFA

| Año  | Sede                | Campeón    | Subcampeón     |
|------|---------------------|------------|----------------|
| 1930 | Uruguay             | Uruguay    | Argentina      |
| 1934 | Italia              | Italia     | Checoslovaquia |
| 1938 | Francia             | Italia     | Hungría        |
| 1950 | Brasil              | Uruguay    | Brasil         |
| 1954 | Suiza               | Alemania   | Hungría        |
| 1958 | Suecia              | Brasil     | Suecia         |
| 1962 | Chile               | Brasil     | Italia         |
| 1966 | Inglaterra          | Inglaterra | Alemania       |
| 1970 | México              | Brasil     | Italia         |
| 1974 | Alemania            | Alemania   | Paises Bajos   |
| 1978 | Argentina           | Argentina  | Paises Bajos   |
| 1982 | España              | Itália     | Alemania       |
| 1986 | México              | Argentina  | Alemania       |
| 1990 | Italia              | Alemania   | Argentina      |
| 1994 | Estados Unidos      | Brasil     | Italia         |
| 1998 | Francia             | Francia    | Brasil         |
| 2002 | Corea del Sur/Japón | Brasil     | Alemania       |
| 2006 | Alemania            | Italia     | Francia        |
| 2010 | Sudafrica           | España     | Paises Bajos   |
| 2014 | Brasil              | Alemania   | Argentina      |
| 2018 | Rusia               | Francia    | Croacia        |



## C. Simbología Lógica y Tablas de Verdad

Escultura en bronce de Aristóteles frente a la Universidad de Freiburg.  
Foto de pixabay.

Este capítulo presenta una pequeña ayuda de memoria para la simbología lógica utilizada en este libro, así como también de las tablas de verdad y otras referencias útiles al momento de realizar actividades de lógica.

Esta ayuda memoria no descarta la necesidad de que el lector incorpore a su saber cotidiano y de forma memorística estos elementos.

### C.1 Simbología de las conectivas y sus palabras asociadas en lenguaje natural en la lógica proposicional

Los símbolos usados en la lógica proposicional son:

| Símbolo           | Nombre                | Idea en lenguaje natural      | palabras clave en lenguaje natural                                                                                                                                                                                                                                                                                                                                            |
|-------------------|-----------------------|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $V$               | Verdadero             | El valor de verdad verdadero. |                                                                                                                                                                                                                                                                                                                                                                               |
| $F$               | Falso                 | El valor de verdad falso.     |                                                                                                                                                                                                                                                                                                                                                                               |
| $\neg$            | Negación              | No                            | no, no es cierto que,<br>no es verdad que,<br>ni (también indica conjunción)                                                                                                                                                                                                                                                                                                  |
| $\wedge$          | Conjunción            | Y                             | ”, ”, y, también, además,<br>adicionalmente, en adición, ahora, incluso,<br>inclusive, así mismo, de igual forma,<br>del mismo modo, igualmente, sin embargo,<br>no obstante, pero, pese, empero, aunque,<br>aún así, a pesar de, tanto como,<br>al igual que, por otra parte, más,<br>“no... ni...” (el “ni” también es una negación),<br>aparte, así mismo, “por otro lado” |
| $\vee$            | Disyunción            | O                             | o, “tal vez ...tal vez ...”,<br>“o de pronto”, aunque de pronto,<br>“puede... o ... puede”, aunque puede                                                                                                                                                                                                                                                                      |
| $\vee$            | Disyunción excluyente | O uno u otro, pero no ambos   | .º bien ..., o bien..."                                                                                                                                                                                                                                                                                                                                                       |
| $\rightarrow$     | Implicación           | Entonces                      | “... entonces ...”, “... implica ...”,<br>“si... entonces...”, “... sólo sí ...”,<br>“... es condición necesaria de ...”,<br>“... es condición suficiente para ...”                                                                                                                                                                                                           |
| $\leftrightarrow$ | Equivalencia          | Si y solo si, es igual a      | “... es equivalente a ...”, “... si y solo si ...”                                                                                                                                                                                                                                                                                                                            |

Las proposiciones utilizadas de forma genérica comienzan con  $p$  (en minúscula) y continúan con las letras subsiguientes ( $q, r, s, \dots$ ). Las proposiciones podrían tener otras letras e incluso palabras si tienen un significado asociado.

El símbolo  $\vdash$  es utilizado para indicar que una conclusión puede deducirse de una o más premisas.

En el caso de tener que asociar la fórmula a elementos semánticos se debe definir un diccionario.

#### ■ Ejemplo C.1

Diccionario:

$f$  = hay fuego

$h$  = hay humo

Fórmula:

$f \rightarrow h, \neg h \vdash \neg f$

■

## C.2 Simbología de la lógica de predicados

Los siguientes son los símbolos asociados a la lógica de predicados.

| Símbolo    | Nombre       |
|------------|--------------|
| $\forall$  | Para todo    |
| $\exists$  | Existe algún |
| $\nexists$ | Ningún       |

Los predicados genéricos suelen enumerarse con la letra  $P$  (en mayúscula) y subsiguientes ( $Q, R, S$ , etc.). Las letras minúsculas desde  $x$  en adelante, y asociadas a una premisa, representan variables proposicionales ( $x, y, z$ , etc.) mientras que las letras  $a$  en adelante ( $a, b, c$ , etc.) representan constantes. Obviamente pueden usarse otras letras según la connotación semántica asociada. Las premisas y las constantes deben estar declaradas en el diccionario.

### ■ Ejemplo C.2

Diccionario:

$P(x) = x$  es una persona

$a =$  Juan

$m =$  María

Fórmula:

$P(a) \wedge P(m)$

■

### C.3 Tablas de verdad de conectivas

Las siguientes son las tablas de verdad elementales de las conectivas lógicas:

| $\Phi$ | $\Psi$ | $\Phi \wedge \Psi$ | $\Phi$ | $\Psi$ | $\Phi \vee \Psi$ |
|--------|--------|--------------------|--------|--------|------------------|
| $V$    | $V$    | $V$                | $V$    | $V$    | $V$              |
| $V$    | $F$    | $F$                | $V$    | $F$    | $V$              |
| $F$    | $V$    | $F$                | $F$    | $V$    | $V$              |
| $F$    | $F$    | $F$                | $F$    | $F$    | $F$              |

| $\Phi$ | $\Psi$ | $\Phi \vee \Psi$ | $\Phi$ | $\Psi$ | $\Phi \rightarrow \Psi$ | $\Phi$ | $\Psi$ | $\Phi \leftrightarrow \Psi$ |
|--------|--------|------------------|--------|--------|-------------------------|--------|--------|-----------------------------|
| $V$    | $V$    | $F$              | $V$    | $V$    | $V$                     | $V$    | $V$    | $V$                         |
| $V$    | $F$    | $V$              | $V$    | $F$    | $F$                     | $V$    | $F$    | $F$                         |
| $F$    | $V$    | $V$              | $F$    | $V$    | $V$                     | $F$    | $V$    | $F$                         |
| $F$    | $F$    | $F$              | $F$    | $F$    | $V$                     | $F$    | $F$    | $V$                         |