ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE MADRID

Introducción a la programación

Santiago Higuera de Frutos

Septiembre 2023

«***Preguntarse cuándo podrán pensar los ordenadores es como preguntarse cuándo podrán nadar los submarinos***»
Edsger W. Dijkstra

Introducción

Históricamente, el cálculo numérico ha sido una herramienta matemática de gran utilidad para los ingenieros y científicos en general. Para ellos, es importante aprender a resolver problemas de todo tipo por métodos numéricos. La llegada de los ordenadores a mediados del siglo pasado posibilitó la aplicación de los métodos numéricos con mucha eficiencia. En ese sentido, las posibilidades que ofrecen los ordenadores son enormes. Hoy en día, cualquier pequeña calculadora es capaz de dar los valores de las funciones trigonométricas, o de complejos cálculos exponenciales. Se trata simplemente de aproximaciones a dichas funciones mediante desarrollos en serie.

Programar un ordenador es una disciplina que requiere distintos tipos de conocimientos, entre los que se incluyen el manejo del sistema de archivos del ordenador y la comprensión de la lógica matemática básica. Además, con el fin de poder aplicar la programación a la resolución de problemas concretos, habrá que entender bien el problema que se quiere resolver y, por último, ser capaz de codificar instrucciones en algún lenguaje de programación concreto.

No hay que confundir programar con aprender un lenguaje de programación concreto. Los principios por los que se rige la programación de ordenadores son independientes del lenguaje de programación que se utilice. Pero, para aprender a programar, es necesario utilizar algún lenguaje concreto. Se podría utilizar el símil de aprender mecánica del automóvil. Los principios por los que se rige un motor diésel, una trasmisión, o un embrague son independientes de una marca comercial concreta. Pero para hacer las prácticas de taller, será necesario utilizar un motor, una trasmisión o un embrague concreto. Luego, con los principios de funcionamiento aprendidos y con un pequeño entrenamiento, se podrán utilizar dichos conocimientos para resolver problemas de motores, trasmisiones o embragues de distintas marcas.

Con la programación pasa algo parecido. Utilizando un primer lenguaje se aprenden los principios de la programación. Luego, una vez que se han comprendido dichos principios, se podrán resolver problemas concretos con diferentes lenguajes. Solo será necesario un pequeño entrenamiento en el nuevo lenguaje.

CAPÍTULO 1

Primeros pasos

Contenido

Los ordenadores son una herramienta de cálculo muy valiosa. No es indispensable saber cómo funcionan para poder utilizarlos, pero un pequeño conocimiento acerca de los sistemas de numeración, de los componentes de un ordenador o de cómo se organiza el sistema de archivos, favorece el aprendizaje posterior de las técnicas de programación.

Este documento no se plantea como un conjunto de materias evaluables para los estudiantes, sino como un conjunto de conocimientos de cultura general y también como un capítulo de consulta posterior, durante el aprendizaje de la programación.

1.1 Sistemas de numeración

Un sistema de numeración es un conjunto de símbolos y reglas de generación que permiten construir todos los números válidos. Los números se representan mediante una cadena compuesta por varios símbolos. Cada símbolo individual se denomina dígito.

Los sistemas de numeración pueden clasificarse en dos grandes grupos: posicionales y no-posicionales:

- **No posicionales:** cada dígito tiene el valor del símbolo utilizado, que no depende de la posición (columna) que ocupa en el número.
- **Posicionales:** el valor de un dígito depende tanto del símbolo utilizado, como de la posición que ese símbolo ocupa en el número.

Como ejemplo de sistema de numeración no posicional puedes recordar el sistema de numeración de los romanos, donde cada símbolo representaba un valor independientemente de la posición que ocupara el dígito en la cifra total[1]. Así, el número romano *XVII* equivale a *10+5+1+1 = 17*.

1.1.1 Sistema de numeración decimal

El sistema de numeración decimal, que es el que se utiliza actualmente en la escritura, es un sistema posicional. Es de procedencia indo-arábiga y lo introdujo en Europa Leonardo de Pisa, también conocido como Fibonacci, a principios del siglo XIII, en su libro *Liber Abaci* (*Libro de cálculo*).

Se utilizan 10 símbolos diferentes: *0, 1, 2, 3, 4, 5, 6, 7, 8* y *9*. Cuando un número está compuesto por más de un dígito, se interpreta como una suma en la que cada dígito equivale a su valor multiplicado por la potencia de diez de la posición que ocupa, siendo la primera posición la *0*, que es la situada a la derecha. Así, por ejemplo:

[1] Hay cierto carácter posicional en el sistema de numeración romano. Por ejemplo, en el número XIV y el número XVI, la I vale $+1$ o -1 según su posición respecto de la V

$$321 = 3 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$$

Como se utilizan diez símbolos, se dice que el sistema de numeración decimal es un sistema de base 10.

1.1.2 Sistema de numeración binario

A raíz de la implantación de los ordenadores, se han comenzado a utilizar otros sistemas de numeración. Por ejemplo, el sistema de numeración *binario* utiliza solo dos símbolos: el 0 y el 1. Se trata de un sistema de numeración posicional de base 2. A cada uno de los dígitos de un número expresado en el sistema binario se le denomina *BIT*, que es el acrónimo de *BInary digiT*.

La razón de utilizar el sistema binario está en la facilidad de implementarlo mediante dispositivos electrónicos que solo admiten dos estados: con corriente eléctrica o sin ella, como se muestra en la Figura 1.1.

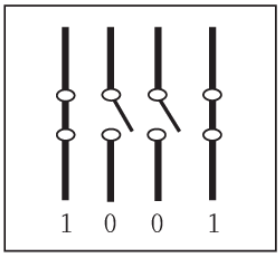


Figura 1.1: Representación de un número binario mediante interruptores abiertos o cerrados

La interpretación decimal de un número binario se hace de manera similar a lo visto para números decimales: cada dígito equivale a su valor multiplicado por la potencia de 2 correspondiente a su posición, empezando en cero por la derecha. Así, se tiene:

$$101 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

Si se realizan las operaciones se verá que el número binario *101* equivale al decimal *5*. A este desarrollo, que se utiliza para calcular el valor decimal de un número expresado en cualquier base, se le denomina *representación polinómica exponencial*.

Aunque pueda parecer difícil, la utilización de los números binarios no es complicada, por el hecho de utilizar solo dos cifras diferentes. Las operaciones suma, multiplicación o exponenciación se realizan de manera muy sencilla, aunque no se va a detallar aquí cómo hacerlas.

Los dispositivos electrónicos facilitan la utilización de números binarios agrupados en conjuntos de 1, 2, 4, 8, 16, 32, 64,... Se trata de grupos correspondientes a potencias de 2:

$$1 = 2^0;\ 2 = 2^1;\ 4 = 2^2;\ 8 = 2^3;\ 16 = 2^4;\ 32 = 2^5;\ 64 = 2^6$$

Los procesadores de los ordenadores manejan grupos de bits en cada ciclo. Los primeros procesadores manejaban grupos de 4 bits, pero actualmente todos manejan grupos de al menos 8 bits.

La agrupación de bits por excelencia es la de 8 bits, que recibe el nombre de *byte*. El byte y sus múltiplos se han convertido en la unidad de medida estándar para la capacidad de los dispositivos de almacenamiento. Los múltiplos del byte más utilizados son el *kilobyte* (kB, mil bytes), el *megabyte* (MB, un millón de bytes), el *gigabyte* (GB, mil megabytes) y el *terabyte* (TB, , mil gigabytes). Hoy día son comunes las memorias USB del orden de algunas decenas de gigabytes (4 GB, 16 GB, 32 GB, 64 GB) y los discos duros desde 500 GB hasta varios terabytes.

Si el prefijo se entiende en términos del Sistema Internacional, *kilo* equivale a 10^3 (miles), *mega* a 10^6, *giga* a 10^9 y *tera* a 10^{12}. Pero lo cierto es que es más habitual entenderlo en términos de potencias de 2, como se indica en la siguiente tabla[2]:

Tabla 1.1: Múltiplos del byte

Prefijo	Sistema Internacional de Medida	Norma ISO/IEC 80000-13
byte	1 byte = 10^0 bytes	1 byte = 2^0 bytes
kilo	1 KB = 10^3 bytes	1 KiB = 2^{10} bytes = 1024 bytes
mega	1 MB = 10^6 bytes = 1000 MB	1 MiB = 2^{20} bytes = 1024 KiB
giga	1 GB = 10^9 bytes = 1000 Mb	1 GiB = 2^{30} bytes = 1024 MiB
tera	1 TB = 10^12 bytes = 1000 GB	1 TiB = 2^{40} bytes = 1024 GiB

1.1.3 Sistema de numeración hexadecimal

El sistema hexadecimal es un sistema de numeración posicional de base 16 (2^4). Utiliza 16 símbolos:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$$

[2]La norma ISO/IEC 80000-13 establece que, cuando los múltiplos del byte se entienden en potencias binarias, los prefijos deben ser *KiB* para el kilobyte, *MiB* para el megabyte, *GiB* para el gigabyte y *TiB* para el terabyte; pero es habitual ver la notación *KB, MB, GB, TB* cuando se refieren a múltiplos de byte como potencia de 2

.

La A hexadecimal es el número 10 decimal, la B el 11, y así hasta la F que es el 15. Se pueden utilizar en mayúsculas o minúsculas. Cuando se está escribiendo un número hexadecimal se suele poner una x delante, para especificar que es hexadecimal; también se usa $0x$ delante.

El sistema hexadecimal actual fue introducido en el ámbito de la computación por primera vez por IBM en 1963. Se utiliza mucho para simplificar la escritura de números binarios, pues cada dígito equivale a un número binario de 4 bits. Así, un número hexadecimal de dos dígitos puede representar números decimales entre el 0 y el 255, o lo que es lo mismo, un *byte*.

La conversión de hexadecimal a decimal se realiza de manera similar a como se ha hecho con los números binarios, pero usando base 16. Algunos ejemplos podrían ser los siguientes:

$$\begin{aligned} A &= 10 \times 16^0 = 10 \\ 1A &= 1 \times 16^1 + 10 \times 16^0 = 26 \\ b3 &= 11 \times 16^1 + 3 \times 16^0 = 179 \\ ff &= 15 \times 16^1 + 15 \times 16^0 = 255 \\ FFFF &= 15 \times 16^3 + 15 \times 16^2 + 15 \times 16^1 + 15 = 65535 \end{aligned}$$

1.2 El Ordenador

Un *ordenador* es un dispositivo electrónico programable capaz de realizar operaciones aritméticas o lógicas. El ordenador recibe unos datos, que se denominan *entrada*, los procesa y devuelve el resultado de dicho procesamiento, que se denomina *salida*.

La palabra ordenador es el término que se utiliza frecuentemente en el español que se habla en España. Proviene del término francés *ordinateur*, introducido en Francia por IBM a mediados del siglo XX. En el español que se habla en América se suele utilizar el término *computador* o *computadora*, que proviene del término inglés *computer*.

El ordenador, como máquina, está formado por una serie de dispositivos electrónicos. Al conjunto de estos dispositivos se le denomina el *hardware* del ordenador. Para que un ordenador pueda funcionar necesita *software*. El software del ordenador es el conjunto de instrucciones que dirige el funcionamiento del

mismo. A estos conjuntos de instrucciones se les suele denominar también *programas*.

Básicamente, la capacidad de un ordenador depende de sus componentes *hardware*, en tanto que, la diversidad de tareas que puede realizar, depende mayormente del software que tenga instalado.

Hay muchos tipos de máquinas programables. Las hay dedicadas al manejo de maquinaria o de dispositivos electrónicos y electromecánicos. A estas máquinas programables se les suele denominar *controladores*, más que ordenadores.

En el lenguaje habitual, se suele reservar el término ordenador a las máquinas programables provistas de teclado, pantalla y unidades de almacenamiento. Los ordenadores más habituales son los servidores, los ordenadores de sobremesa, los portátiles, las tabletas y los teléfonos móviles.

La denominación *servidor* se suele utilizar para denominar los ordenadores que dan servicio a otros ordenadores a través de una red de telecomunicaciones. Un ejemplo serían los servidores que ofrecen páginas web a través de Internet, o el ordenador central que da servicio a los ordenadores individuales en una empresa, a través de una red cableada.

1.2.1 Componentes *hardware* del ordenador

Los componentes *hardware* de un ordenador de sobremesa suelen ser la unidad central, el monitor, el teclado, el dispositivo apuntador y las unidades de almacenamiento. El dispositivo apuntador más frecuente en la actualidad es el ratón. Los ordenadores portátiles suelen tener los mismos componentes que los de sobremesa. Las tabletas y los teléfonos disponen de pantallas táctiles que integran el teclado y permiten utilizar los dedos como dispositivo apuntador. Los servidores suelen tener solo la unidad central y los dispositivos de almacenamiento; para manejarlos, se utilizan uno o más ordenadores de la red.

La unidad central consta en realidad de varios dispositivos que conviene conocer. El núcleo central lo constituye el *procesador*, también denominado *Unidad de Procesamiento* o *CPU (Central Processing Unit)*. La CPU es la que controla el resto de dispositivos: unidades de memoria, puertos de entrada-salida y controladoras. La alimentación eléctrica se hace a través de una unidad de alimentación que se conecta a la red electrica.

Entre las controladoras, destaca la placa que controla las salidas por pantalla. Hoy en día las pantallas tienen resoluciones de varios millones de puntos, que se actualizan a frecuencias superiores a 60 Hz. Eso hace necesarios muchos

cálculos por segundo, lo que ha llevado a desarrollar placas específicas para la gestión de la salida gráfica por pantalla. Se trata en realidad de un auténtico ordenador dedicado en exclusiva al manejo de la información gráfica, liberando al procesador principal de dicha tarea. Se le suele llamar la *tarjeta gráfica* o GPU (*Graphic Processing Unit*) y su coste es elevado. Es frecuente que los programas utilicen la potencia de cálculo de las tarjetas gráficas para otros tipos de procesamientos o para utilizar el procesamiento *en paralelo*.

El *bus de datos* es el componente que comunica el procesador con el resto de componentes. Es un elemento fundamental para el funcionamiento del ordenador. La Figura 1.2 muestra el esquema de la comunicación del procesador con el resto de componentes a través del bus datos.

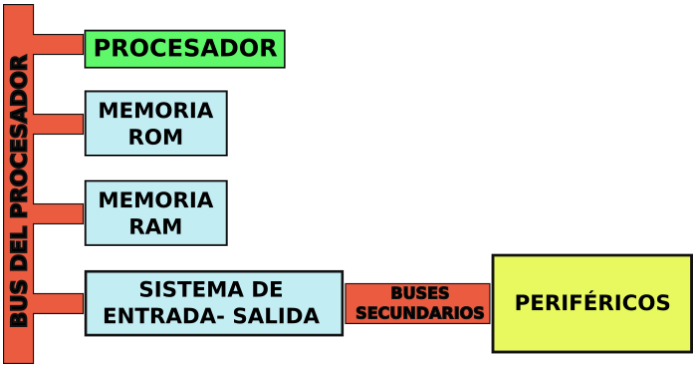


Figura 1.2: Esquema de comunicación del procesador a través del bus de datos

La *memoria* de los ordenadores está formada por dispositivos electrónicos que permiten almacenar información, ya sean datos o instrucciones de programas. Se comunican con el procesador a través del bus de datos. En los ordenadores hay dos tipos de memoria:

- **Memoria fija:** también conocida como memoria *ROM* (*Read Only Memory*), contiene instrucciones que no se borran aunque se apague el ordenador. Estas instrucciones fijas son las encargadas de gestionar el arranque de la máquina, la activación del resto de dispositivos y la carga del sistema operativo.

- **Memoria volátil:** denominada memoria *RAM* (*Random Access Memory*), es una cantidad de memoria que se borra cada vez que se apaga el ordenador. En ella se cargan los datos, los programas y los resultados de los procesamientos, durante el funcionamiento del ordenador.

1.2.2 La capacidad del procesador

El procesador es el componente básico de cualquier ordenador. A día de hoy, dos marcas copan el mercado de procesadores para ordenadores de sobremesa: Intel y AMD. Ambas ofrecen procesadores de alta calidad. Apple incorpora a sus dispositivos sus propios procesadores. En el caso de los teléfonos móviles, hay distintos fabricantes de procesadores, por ejemplo Qualcomm o Mediatek, aunque cada vez más, los propios fabricantes de móviles (Samsung, Huawey, Google, Apple,...) incorporan sus propios procesadores.

Los datos que permiten estimar la calidad de un determinado procesador son los siguientes:

- **Velocidad de reloj:** internamente, el procesador dispone de una señal de reloj que permite sincronizar las diferentes tareas. Se mide en Hz, y hoy día son habituales frecuencias de algunos GHz. No coincide con la velocidad que percibe el usuario pues, dependiendo de cómo esté diseñado el procesador, puede realizar más o menos tareas en un ciclo de reloj.
- **Anchura en bits del bus:** es el número de bits que puede trasmitir en un ciclo de reloj. Actualmente, en los ordenadores de sobremesa es habitual una anchura de 64 bits (8 bytes).
- **Velocidad del bus:** es la velocidad de reloj a la que funciona el bus. Suele estar por debajo de la del procesador.
- **Memoria caché:** es una memoria muy rápida que almacena una copia de los datos que necesitará a continuación el procesador. El rendimiento de un procesador mejora con el aumento de la caché.
- **Tipo de núcleo:** un procesador con varios núcleos equivale a disponer de varios procesadores trabajando en paralelo. Esto permite la ejecución de varios programas de manera simultánea, con lo que se puede incrementar el rendimiento. Son habituales las denominaciones comerciales *dual core* (dos núcleos) o *quad core* (cuatro núcleos).

La velocidad real de un determinado ordenador depende de su eslabón más débil. Un procesador muy rápido con un bus o unas memorias lentas verá afectado su rendimiento por la menor capacidad de esos componentes.

1.2.3 Ejecución de los programas

Inicialmente, los programas están almacenados en las unidades de almacenamiento (en los discos). Para poderlos ejecutar, se tienen que cargar primero en la memoria RAM y desde ahí, es desde donde se ejecutan y transmiten sus instrucciones al procesador. La tarea de cargar los programas en memoria corresponde al sistema operativo.

El ordenador mantiene en ejecución numerosos programas de manera aparentemente simultánea: controladores de las memorias, discos, WIFI, Bluetooth, salida por pantalla, entrada por teclado, etc. A esto, habrá que sumar los programas que ordene ejecutar el usuario: correo, navegador web, reproductor de música, etc.

En realidad, la ejecución no es simultánea. Se reparte el tiempo de reloj del procesador entre los diferentes programas, en lo que se conoce como *ejecución concurrente*. Pero sucede que, dada la gran velocidad del reloj, la percepción del usuario es la de que los programas se ejecutan de manera simultánea.

Si el procesador dispone de más de un núcleo, entonces sí que se pueden realizar varias tareas de manera *más* simultánea. Es lo que se conoce como *ejecución en paralelo*.

Un mismo programa, para que pueda beneficiarse de la ejecución en paralelo, tiene que haber sido programado de manera adecuada. Es el caso de algunos juegos de ordenador o de determinados programas científicos.

1.3 El sistema operativo

El *sistema operativo* de un ordenador es el software encargado de interactuar con el usuario, de gestionar el sistema de archivos, las comunicaciones y de ejecutar el resto de programas. Actualmente existen varios sistemas operativos con mayor o menor implantación, según el tipo de ordenador de que se trate.

En los ordenadores de sobremesa y en los ordenadores portátiles, el sistema operativo más extendido es Windows, de la empresa Microsoft. En los ordenadores fabricados por la empresa Apple se utiliza el sistema operativo MacOS. Entre científicos, programadores y analistas de sistemas es frecuente utilizar alguna variante del sistema operativo Linux.

En los servidores se utilizan casi en exclusiva sistemas operativos basados en Linux. En los teléfonos móviles y tabletas básicamente hay dos sistemas operativos: Android, de la empresa Google e IOS de Apple.

Los sistemas operativos de Microsoft y de Apple son de código propietario (cerrado) y sujetos a una licencia que hay que pagar para poderlos utilizar. Linux, en cambio, es software de código abierto disponible de manera gratuita. Tanto Android, como MacOS o IOS están basados en Linux.

Cuando se arranca el ordenador, la memoria ROM contiene las instrucciones necesarias para cargar el sistema operativo en memoria. En realidad no se carga en memoria todo el sistema operativo, sino solo la parte indispensable. A medi-

da que se solicitan acciones al sistema operativo, se van cargando en memoria distintos programas, que son los encargados de gestionar dispositivos, cargar y ejecutar otros programas, leer y escribir los ficheros de los discos o cualquier otra tarea de las que realiza el sistema.

1.4 El sistema de archivos

El *sistema de archivos* o *sistema de ficheros* (*file system*) es el elemento del sistema operativo que controla cómo se almacenan y recuperan los datos. Es el encargado de administrar y estructurar la información en las unidades de almacenamiento: discos, memorias USB, etc. El sistema de archivos se encarga de asignar espacio a los nuevos archivos, administrar el espacio libre y conceder acceso a los datos.

Cada sistema operativo suele tener su propio sistema de archivos pero, de cara al usuario, casi todos muestran una organización basada en *directorios* y *archivos* organizados de manera jerárquica en una estructura en forma de árbol[3]

En una estructura en árbol, la información se organiza en *ramas* y *hojas*. Cada rama puede contener otras ramas u hojas. En el sistema de archivos, los directorios son las ramas y los archivos las hojas.

Un *archivo* es una unidad de información independiente que alberga datos o instrucciones de programa. Una *carpeta* o *directorio* es una agrupación que se hace de la información almacenada en un dispositivo de almacenamiento que está compuesta de otros directorios o archivos. Cada dispositivo de almacenamiento tiene un *directorio raíz*, que es donde empieza la estructura de árbol del dispositivo. Dentro del directorio raíz, hay otros directorios o archivos.

La Figura 1.3 muestra una estructura jerárquica en forma de árbol que empieza en la parte superior de la figura con el directorio o carpeta raíz y se va extendiendo hacia abajo mediante carpetas que contienen otras carpetas o archivos.

[3] El nombre *directorio* es equivalente al nombre *carpeta*. Originalmente se llamaban directorios. En los años 80 del siglo pasado, la empresa Apple Inc. introdujo la *Interfaz Gráfica de Usuario*, GUI (*Graphic User Interface*), en su popular ordenador Macintosh. En esta interfaz gráfica, los directorios quedaban simbolizados por el dibujo de una carpeta. A partir de ahí, la mayoría de sistemas operativos incorporaron sus GUI manteniendo el símbolo de la carpeta asociado a los directorios. A día de hoy, se puede utilizar indistintamente el nombre *carpeta* o *directorio* para referirse a las agrupaciones de archivos que se utilizan en la estructura del sistema de archivos.

Figura 1.3: Estructura en árbol del sistema de archivos de una unidad de almacenamiento

1.4.1 Nombre de los archivos

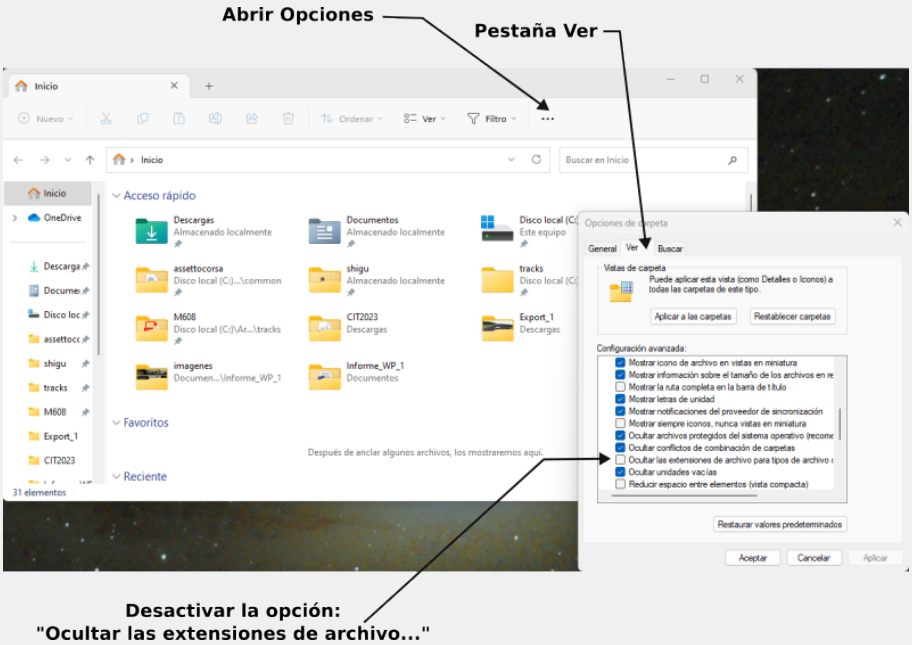
Cada archivo o directorio tiene un nombre asociado. La regla general es que no puede haber dos elementos con el mismo nombre dentro de un mismo directorio. El nombre suele constar de dos partes separadas por un punto, que se denominan *nombre* y *extensión*. Hay que respetar las siguientes convenciones:

- Los *nombres* deben estar formados por cualquier combinación de números, letras y símbolos permitidos.
- La extensión del archivo suele estar formada por tres o cuatro caracteres, que se separan del nombre propiamente dicho por un punto, y sirven para indicar al sistema operativo qué tipo de archivo es. Por ejemplo, como se verá más adelante, los nombres de archivo que contienen instrucciones de programas en C llevan la extensión *.c*. Otros tipos de fichero llevan otras extensiones que identifican el tipo de archivo. Ejemplos habituales son *.txt* para archivos de texto plano, *.docx* para los ficheros de los procesadores de texto, *.xlsx* para los ficheros de hojas de cálculo, *.png* o *.jpg* para formatos de imagen y muchos más.

Consejo

En su configuración por defecto, el explorador de archivos de Windows oculta la extensión de los archivos y solo muestra su nombre. Propone identificar el tipo de archivo por el icono que muestra a la izquierda del nombre. Esto hace que sea frecuente ver en el mismo directorio ficheros que se llaman igual. En realidad no se llaman igual, lo que sucede es que la extensión del nombre de los archivos, que es lo que los diferencia, está oculta. Esto es muy incómodo, sobre todo, cuando se está programando.

Se recomienda desactivar esta opción, lo que evitará quebraderos de cabeza en muchas situaciones mientras se programa. Para ello, hay que desactivar la opción *Ocultar extensiones de archivos...* en el diálogo *Opciones de carpeta* y luego pulsar el botón para aplicar esta configuración a todas las carpetas. La figura siguiente muestra el explorador de archivos de Windows 11 y da indicaciones de cómo desactivar esta opción.



1.4.2 La ruta de acceso al archivo

Cada archivo o directorio, además de tener un nombre determinado, está situado dentro de un directorio determinado. Para identificar el directorio en el que está guardado un elemento concreto se pone el nombre de todos los directorios que hay que recorrer, desde el directorio raíz, hasta el directorio en el que está el elemento de que se trate.

Los nombres de cada carpeta se separan con el separador que tenga establecido el sistema de archivos, que en Windows es la barra invertida, \, y en Linux y Mac la barra inclinada normal, /.

El directorio raíz también tiene distinta nomenclatura según el sistema operativo. En Windows, se pone una letra mayúscula denominada *nombre de la unidad*, seguida de dos puntos. En Linux y Mac se utiliza la barra inclinada /.

A esta lista de directorios que hay que seguir hasta llegar a un archivo o directorio concreto se le suele llamar *ruta* del archivo (en inglés, el *file path*). Observe el siguiente ejemplo:

C:\MisCarpetas\Carpeta_2

La línea anterior hace referencia a una carpeta llamada *Carpeta_2*, que está situada dentro de una carpeta llamada *MisCarpetas* del directorio raíz del disco *C:* de un sistema operativo Windows.

Observe ahora la siguiente línea:

/MisCarpetas/Carpeta_2

Esta línea hace referencia a una carpeta llamada *Carpeta_2*, que está situada dentro de una carpeta llamada *MisCarpetas* situada en el directorio raíz de una unidad de disco de un sistema operativo Linux o Mac.

Como puede ver, la descripción de una ruta es muy similar en los tres sistemas operativos mencionados, variando solo el separador y la designación del directorio raíz.

En todo momento, durante el uso de cualquier programa, se supone que el usuario está posicionado en un directorio concreto del sistema de archivos, que recibe la denominación de *directorio actual* o también *directorio de trabajo*.

El sistema para acceder a un archivo o recurso a través de Internet utiliza un sistema de rutas similar al descrito. Primero va la dirección web del ordenador al que se quiere conectar, por ejemplo, https://caminos.upm.es y, a continuación, la ruta dentro de ese ordenador al recurso al que se quiere acceder. Casi la totalidad de servidores de páginas a través de Internet utiliza el sistema operativo Linux, y el separador de carpetas en las direcciones web es la barra inclinada /. Un ejemplo de la ruta de acceso a un fichero en Internet podría ser el siguiente:

https://webs.um.es/iverdu/AP07_FicherosA.pdf

Observe que la dirección anterior se refiere al ordenador *webs.um.es*, dentro del cual hay una carpeta llamada *iverdu* que contiene el fichero llamado *AP07_FicherosA.pdf*. Si se teclea esa ruta en el navegador, se descargará el fichero.

Consejo

Aunque los sistemas operativos actuales permiten que los nombres tengan caracteres acentuados, espacios y otros caracteres especiales, se recomienda utilizar solo combinaciones de letras del alfabeto inglés, números y el símbolo de guion bajo para los nombres de archivos o directorios.

De esta forma se evitan problemas de compatibilidad entre distintos sistemas operativos y con algunos programas.

En particular, la utilización de espacios en los nombres de archivos provoca muchos problemas de compatibilidad, por lo que se recomienda usar el guion bajo para separar las distintas palabras de un mismo nombre, o el criterio *Camel Case*, poniendo las palabras juntas pero con la primera letra de cada palabra en mayúsculas. Los siguientes ejemplos podrían ser nombres de archivo sin problemas de compatibilidad:

`datos_anuales.txt` `DatosAnuales.txt`

1.4.3 Rutas absolutas y relativas

Para identificar un archivo concreto, el sistema operativo necesita conocer, además de su nombre completo, el directorio en el que se encuentra, su ruta. La indicación de la ruta se puede hacer de dos maneras: indicando su *ruta absoluta* o indicando su *ruta relativa*:

- **Ruta absoluta:** en este caso se indica la ruta completa hasta el archivo, partiendo del directorio raíz de la unidad de almacenamiento. Un ejemplo podría ser el siguiente:

  `C:\carpeta_1\datos.txt`

  Esta ruta se refiere al fichero *datos.txt* que se encuentra dentro de la carpeta de nombre *carpeta_1* del directorio raíz de la unidad *C:*.
- **Nombre de archivo solamente:** si solo se especifica el nombre de archivo, por ejemplo *datos.txt*, el sistema operativo buscará el archivo dentro del directorio actual, también llamado *directorio de trabajo*.
- **Ruta relativa:** por ejemplo, *carpeta_1\datos.txt*; en este caso se está indicando que se requiere el archivo *datos.txt* que está dentro de la carpeta *carpeta_1* del directorio actual.

En los ejemplos anteriores se han utilizado direcciones de una unidad de almacenamiento de Windows. En los sistemas Linux o MacOS se haría de manera similar, simplemente cambiando el caracter separador y la denominación del directorio raíz.

1.4.4 Tipos de archivos

Hay dos tipos de archivos:

- archivos de *texto plano*.
- archivos *binarios*.

Los *archivos de texto plano* contienen letras, números y algunos caracteres especiales, como el carácter *fin de línea*, por ejemplo. Se pueden abrir en un editor de texto para examinar su contenido. Al programar, el código de los programas se desarrolla en archivos de texto plano. También se pueden utilizar para guardar información o para otros usos específicos. En estos archivos, no hay tipo de letra, o negritas o colores. Solo hay caracteres[4].

Los *archivos binarios* son una secuencia de bytes y suelen ser interpretados como alguna cosa que no sean caracteres de texto. Un ejemplo típico son los programas de ordenador compilados, pero también se utilizan para almacenar imágenes, sonido, versiones comprimidas de otros archivos, etc.

Observación

A lo largo del texto se utilizará de manera indistinta la denominación *archivo* o *fichero*.

1.4.5 Editores y procesadores de texto

Los *editores de texto* son programas que se utilizan para crear, inspeccionar o modificar los archivos de texto plano.

Cuando se abre un archivo de texto plano en el editor de texto, se verá con un tipo de letra concreto y, a veces, también con diferentes colores. Son las ayudas que proporciona el propio editor, para dar legibilidad al texto, pero el archivo de texto, en sí, no contiene información específica de tipo de letra, colores u otras consideraciones de estilo.

El tipo de letra (la fuente) se elige en el editor, y se aplica a todo el archivo. Es habitual utilizar *fuentes monoespaciadas*. La fuentes monoespaciadas son juegos de caracteres en los que todos los caracteres ocupan la misma anchura al visualizarlos. En estos juegos de caracteres, la letra *m* ocupa lo mismo que

[4] En realidad, los archivos de texto plano son archivos binarios que solo tienen caracteres con determinada codificación.

la letra *i*, por ejemplo. Emulan a las antiguas máquinas de escribir, y es habitual utilizarlos para visualizar archivos de texto plano. Un clásico ejemplo de fuente monoespaciada es el tipo de letra *Courier*.

Para programar, se utilizan editores de texto. Hay numerosos editores disponibles. Algunos son comerciales y otros de uso libre. Tres ejemplos de editores de uso libre de alta calidad son *Visual Studio Code*, de *Microsoft*, *Notepad ++* o *eMacs*.

Los editores de texto suelen proporcionar *ayudas a la programación*, por ejemplo, colorear de diferente manera distintas zonas de código, o *autocompletar* las expresiones habituales del lenguaje que se esté utilizando. También es habitual que indiquen los errores de sintaxis en el código.

Los *procesadores de texto* son programas que se utilizan para escribir documentos. Permiten dar formato a los párrafos, seleccionar diferentes tipos de letra en diferentes partes del documento, incluir imágenes, crear índices de contenido, etc. Los archivos que generan incluyen, además del texto propiamente dicho, información del estilo de cada parte del documento. Ejemplos de procesadores de texto son *Microsoft Word*, *Libre Office Writer* o *Google Doc*.

Inicialmente, los procesadores de texto guardaban sus documentos en archivos binarios de formato propietario. Es el caso del formato *.doc* de *Microsoft word*. No obstante, se van imponiendo los formatos abiertos que permiten intercambiar documentos entre distintos procesadores. Para documentos de texto, el estándar de formato abierto es el *ODT, Open Document Text*[5]. Actualmente, *Microsoft* utiliza el formato abierto *.docx*, similar al *odt*. Aún así, sigue siendo difícil utilizar los documentos generados con *Word* en otros procesadores. Hay quien piensa que es la propia empresa Microsoft la que pone dificultades para que nunca haya una total compatibilidad de sus documentos con los de otros procesadores.

Otro enfoque diferente para la creación de documentos es el utilizado por LaTeX. Se trata de un sistema de creación de documentos de alta calidad tipográfica. El objetivo inicial era la creación de documentos con mucha carga de contenido matemático pero, actualmente, permite la creación de documentos con todo tipo de contenidos. LaTeX es más difícil de utilizar que los procesadores mencionados anteriormente pero, una vez que se sabe utilizar, los resultados son de alta calidad y se incrementa la productividad. Es una alternativa muy eficaz para escribir libros, tesis doctorales y otros tipos de documentos complejos. Este texto

[5] El 1 de julio de 2015, ISO e IEC presentaron una nueva revisión del estándar ODT denominada ISO/IEC 26300-1:2015

se ha escrito completamente en LaTeX. Un buen libro para aprender a utilizar LaTeX es la referencia [1] de la bibliografía.

Otra alternativa interesante es *Markdown*. Se trata de un lenguaje muy sencillo para crear documentos utilizando realzados del texto, listas, enlaces, imágenes y otros elementos. Una característica de *Markdown* es que los documentos creados se exportan fácilmente a otros formatos, por ejemplo al lenguaje HTML que utilizan las páginas web o los blogs. También se utiliza en programas de planificación de tareas, como *Todoist*, o en programas de mensajería chat, como *Microsoft Teams* o *Discord*. El lector puede consultar la referencia [2].

Una característica común a LaTeX, *MarkDown* y otros sistemas similares de escritura es que los documentos se crean utilizando un editor de texto plano, lo que facilita poder concentrarse en el contenido, dejando los aspectos relativos al formato para un momento posterior; el texto plano también facilita el trabajo colaborativo de varias personas en un mismo documento. Estos documentos, al estar basados en archivos de texto plano, se pueden crear mediante un programa, utilizando las técnicas de escritura de archivos que proporcionan los lenguajes de programación. Animamos al lector a experimentar con este tipo de sistemas de escritura, que le permitirán salirse del limitado marco de *Microsoft Word*.

¡Atención!

Para programar, escribir documentos con LaTeX o con *Markdown*, se utilizan *editores de texto plano*. Los *procesadores de texto*, como *Microsoft Word* o *Libre Office Writer*, no se utilizan para programar.

1.5 Lenguajes de programación

Un *lenguaje de programación* es un lenguaje formal, con reglas gramaticales bien definidas, que permite escribir una serie de instrucciones, o secuencias de órdenes en forma de algoritmos, con el fin de controlar el comportamiento físico o lógico de un sistema informático. A todo este conjunto de órdenes escritas mediante un lenguaje de programación se le denomina *programa informático*.

El ordenador se maneja mediante códigos binarios en lo que se denomina *código máquina*, que es lo que entiende el procesador. Los lenguajes de programación facilitan la creación de instrucciones en un lenguaje más comprensible pero, al final, el programa desarrollado en cualquier lenguaje hay que traducirlo a código máquina. Al archivo con las instrucciones escritas en un lenguaje de pro-

gramación determinado se le denomina el *código fuente del programa*. Según el instante en el que se traduce a código máquina, los lenguajes se clasifican en *lenguajes compilados* y *lenguajes interpretados*.

En los *lenguajes compilados*, tras codificar el programa en texto plano con un editor, se *compila*, generando el archivo de código máquina que será lo que se ejecute en el ordenador. A este archivo se le denomina el *binario* del programa. Los archivos *.exe* del sistema operativo *Windows* son ejemplos de programas compilados. Un mismo código fuente necesita una versión diferente de código máquina para cada procesador y para cada sistema operativo. Ejemplos de lenguajes compilados son el C y sus variantes, el Rust, el Fortran, etc.

En los *lenguajes interpretados*, hay un componente denominado *intérprete* o también *máquina virtual*, que va cogiendo las instrucciones del programa y traduciéndolas en directo a código máquina. Ejemplos de lenguajes interpretados son el Java, el Python o el Javascript.

En general, los lenguajes compilados son más rápidos y se pueden ejecutar sin necesidad de disponer de ningún intérprete. Se utilizan para cualquier aplicación en la que la velocidad sea crítica, como puede ser el diseño CAD, el tratamiento de imágenes, el modelado 3D, el cálculo de estructuras, los juegos de ordenador y otros.

Los lenguajes interpretados son más lentos que los lenguajes compilados pero, su facilidad de uso, hace que estén muy extendidos en la comunidad científica como lenguajes para análisis de datos, estadística, matemáticas, gráficos y otros campos. Otra característica que hace atractivos estos lenguajes es que, si se dispone del intérprete correspondiente, el mismo programa puede funcionar en diferentes sistemas operativos o distintos procesadores. Por ejemplo, un programador puede desarrollar un programa Java en su ordenador Apple y el código funcionará en un ordenador Windows o Linux, siempre que tengan instalado Java.

Como se verá más adelante, para poder resolver cualquier problema de computación, el lenguaje de programación tiene que cumplir determinadas condiciones. Entonces se dice que es un lenguaje *completo*. Los lenguajes mencionados en los párrafos anteriores son todos lenguajes completos.

Hay lenguajes que, a pesar de ser completos, están pensados para un ámbito muy específico, y no se suelen utilizar para programación general. Por ejemplo, el lenguaje SQL es un lenguaje completo, pero está pensado para interactuar con bases de datos relacionales, y no se suele utilizar fuera de ese ámbito.

Hay otros lenguajes, que sin servir para resolver algoritmos, son muy útiles en campos específicos. Un ejemplo sería el lenguaje *HTML*, que se utiliza para escribir páginas web. También está muy extendido el uso del lenguaje *XML*, que sirve para describir documentos o conjuntos de datos. Hay numerosas implementaciones del *XML*; por ejemplo, el formato *.docx* que se ha mencionado antes, es un formato basado en *XML*, como también lo es el formato *GPX*, en el que proporcionan los *tracks* los dispositivos GPS; y hay muchos más ejemplos de formatos de datos que utilizan *XML*.

Los principios por los que se rige la programación, son independientes del lenguaje elegido. Lo importante es aprender a programar en alguno de los lenguajes existentes. Actualmente hay cientos de lenguajes de programación. El índice TIOBE, que se actualiza cada mes, permite hacerse una idea de los lenguajes más utilizados en cada momento. Puede consultar el último índice TIOBE publicado en el siguiente enlace:

https://www.tiobe.com/tiobe-index/

1.6 Algoritmos

Un *algoritmo* es un conjunto de instrucciones o reglas definidas y no ambiguas, ordenadas y finitas que permite solucionar un problema, realizar un cómputo, procesar datos y llevar a cabo otras tareas o actividades. Dado un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución.

Aunque se pueden utilizar algoritmos para resolver cualquier tipo de problema, aquí se utilizarán solo para resolver problemas de computación.

Por ejemplo, para obtener el valor máximo de un conjunto finito de números enteros se podría proceder como sigue:

1. Ordenar los elementos en forma de lista.
2. Establecer el primer elemento de la lista como valor máximo.
3. Coger el valor del siguiente elemento de la lista y compararlo con el valor máximo guardado. Si el valor del elemento es mayor que el valor máximo guardado, guardarlo como nuevo valor máximo, sustituyendo al que había.
4. Repetir el paso 3 hasta finalizar la lista.
5. Al terminar de recorrer la lista, el valor máximo será el valor que haya guardado.

Como se puede apreciar, el algoritmo está descrito en términos de lenguaje corriente. Los algoritmos, en sí mismos, son independientes del lenguaje de

programación concreto en el que se implementen posteriormente. Se podría implementar utilizando cualquier lenguaje de programación completo.

Existen numerosos algoritmos para resolver problemas de computación. A lo largo del curso se mostrarán algunos casos sencillos de algoritmos que todo programador debe conocer.

1.7 El lenguaje C

La historia de C está íntimamente ligada a la del sistema operativo UNIX, que empezó a desarrollarse en 1969 usando lenguaje ensamblador, que es lo más cercano que existe a programar en código máquina.

C es un lenguaje de propósito general originalmente desarrollado por Dennis Ritchie y Ken Thompson entre 1969 y 1972 en los laboratorios Bell, como evolución del anterior lenguaje B. Más de ciencuanta años después, sigue siendo uno de los lenguajes más utilizados, tanto en entornos profesionales como para el aprendizaje de la programación.

Antes de C ya existían lenguajes de alto nivel, como FORTRAN o COBOL, que tenían cierta portabilidad entre diferentes máquinas, pero eran bastante ineficientes en el uso de memoria y tiempo de ejecución. El lenguaje C se concibió como una solución a este problema, diseñándolo para ser al mismo tiempo portable y eficiente.

Desde las primeras versiones, se incluían el compilador de C y algunas utilidades, lo que favoreció su portabilidad a otras máquinas; en teoría, *solo* había que crear un compilador para el sistema al que lo quisieras portar.

Otra de sus características distintivas fue la implementación nativa del paradigma de programación estructurada, que introducía conceptos como subrutinas, estructuras condicionales (*if* y *while*) y de iteración (bucles *for* y *while*). Estas abstracciones permitieron a los programadores escribir código más legible y más fácil de mantener e impulsó de forma decisiva la adopción de la programación estructurada en toda la industria.

En 1978 Ritchie y su compañero Brian Kernighan, el autor del primer *Hola, Mundo*, publicaron el libro *The C Programming Language*, contibuyendo de manera decisiva a la difusión de C.

En 1983, la American National Standards Institute o ANSI creó un comité para crear una especificación estándar de C. En 1989, se ratificó dicho estándar y en 1990 fue adoptado por la ISO (*International Organization for Standardization*). Desde entonces, las distintas versiones de C se han publicado como un estándar

comúnmente aceptado.

Dennis MacAlistair Ritchie falleció el 12 de octubre de 2011 a los 70 años, solo en su casa de New Jersey. Su muerte no tuvo mucho eco en unos medios de comunicación centrados en el deceso de Steve Jobs, que se había producido apenas una semana antes. Paradójicamente, Ritchie creó algunas tecnologías que fueron clave para el éxito de Jobs. Al fin y al cabo, MacOS es un nieto de UNIX, de la misma forma que Objective-C es un superconjunto de C.

REFERENCIAS

[1] Julio Mulero González and Juan Matías Sepulcre Martínez. *LATEX con palabras clave: 2ª edición*. Universidad D'Alacant, October 2020. ISBN 978-84-9717-704-7.

[2] Basic Syntax | Markdown Guide. URL https://www.markdownguide.org/basic-syntax/.