Tema 1

Introducción a la computación

El ”ordenador” se refiere a una máquina que organiza o procesa datos de acuerdo con un conjunto de instrucciones predefinidas. En inglés, estas máquinas se describen usando el término ”computer”, que se deriva del verbo latino ”computare”, calcular o contar. En español, sin embargo, el término que utilizamos para estos dispositivos es la palabra ”ordenador” que proviene del verbo ”ordenar”, en el contexto de organizar o disponer cosas de manera adecuada. En este tema vamos a intentar explicar someramente como están formados los ordenadores y como funcionan.

1.1. Componentes de un ordenador

Un ordenador está constituido por dos partes esenciales, el hardware y el software.

- El **hardware** se refiere a todos los componentes físicos y tangibles de una computadora o sistema informático. Estos componentes incluyen todas las partes físicas que podemos ver y tocar, como la unidad central de procesamiento, la memoria RAM, el disco duro, la placa base, el monitor, el teclado o el ratón, entre otros dispositivos periféricos.
- El **software** se refiere a los programas informáticos, conjuntos de instrucciones o datos que permiten al ordenador realizar diversas funciones o tareas. Este conjunto de instrucciones puede variar desde tareas aparentemente simples como el control del hardware, el procesamiento de texto hasta operaciones complejas como el control de sistemas automatizados o la simulación de fenómenos físicos.

John Von Neumann[1], propuso en 1945 un diseño conceptual en el que se basan la mayoría de los ordenadores modernos. Este modelo describe la arquitectura básica de un ordenador digital y consta de los siguientes componentes principales:

1. La Unidad Central de Procesamiento (”Central Processing Unit” en inglés, CPU).
2. La Unidad de Almacenamiento, o ”Memoria”.
3. Las Unidades de Entrada y Salida.

Estas unidades están interconectadas, tal y como se representa en la Figura **??**.

[1] John von Neumann, uno de los matemáticos más importantes del Siglo XX y uno de los padres de los ordenadores tal y como los conocemos hoy en dia https://es.wikipedia.org/wiki/John_von_Neumann

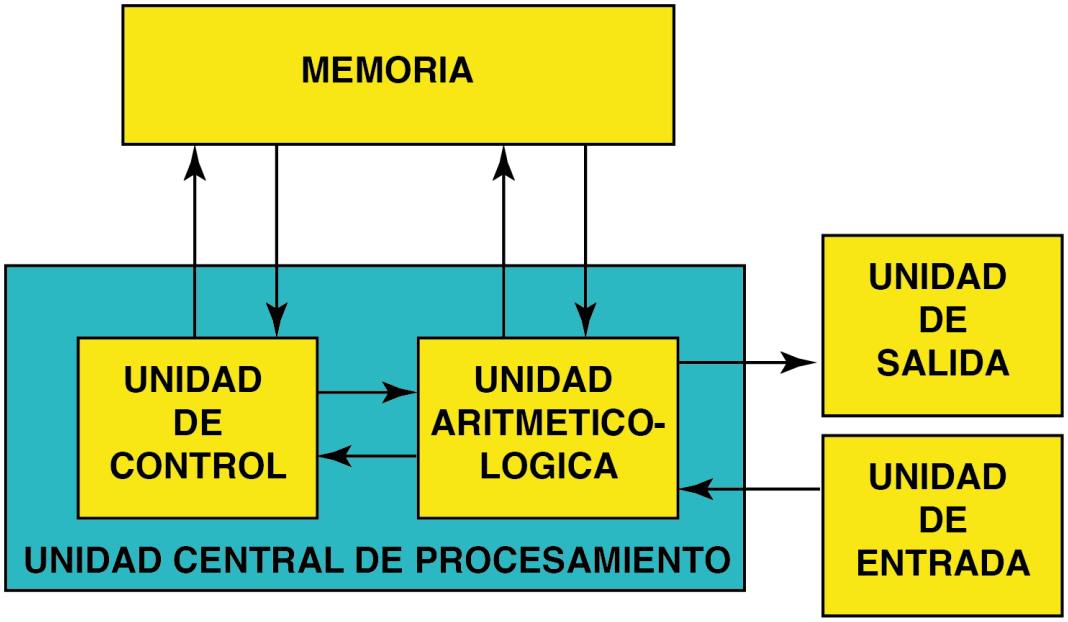

Figura 1.1: Arquitectura de Von Neumman.

La **Unidad Central de Procesamiento o CPU** es el ”cerebro” del ordenador, responsable de interpretar los programas y realizar operaciones aritméticas y lógicas. La CPU ejecuta instrucciones almacenadas en la memoria, procesando datos según las operaciones especificadas. La CPU cuenta con dos elementos:

- La unidad de control: responsable de coordinar y controlar todas las operaciones dentro del ordenador. Esta unidad, coordina la ejecución de las instrucciones, dirigiendo el flujo de datos y controlando el funcionamiento de otras unidades del sistema.
- La unidad aritmético-lógica: responsable de realizar las operaciones aritméticas y lógicas en los datos. Realiza las operaciones básicas como suma, resta, multiplicación, división y también las operaciones lógicas como AND, OR, NOT, etc...

La **memoria**, o unidad de almacenamiento comprende los dispositivos que se utilizan para almacenar datos e instrucciones. Estos dispositivos pueden incluir memoria RAM, memoria ROM, discos duros o unidades de estado sólido (SSD), entre otros.

La **unidad de entrada** se encarga de recibir datos y convertirlos en una forma que la computadora pueda entender y procesar. Dispositivos típicos de entrada pueden ser teclados, ratones, escáneres, micrófonos, etc... Finalmente, la **unidad de salida** toma los resultados de los cálculos realizados por la computadora y los presenta de una forma comprensible para el usuario. Los dispositivos de salida comunes incluyen monitores, impresoras, altavoces, etc...

1.1.1. Unidad Central de Procesamiento

Como se ha explicado anteriormente, la CPU se encarga de ejecutar las instrucciones contenidas en los algoritmos (”códigos”) y de controlar el funcionamiento de los distintos componentes del ordenador. Suele estar integrada en un chip denominado microprocesador, que engloba las unidades de control y la unidad aritmético-lógica en un mismo elemento. Algunas de las características y funciones clave de la CPU incluyen:

- La ejecución de instrucciones: La CPU interpreta y ejecuta instrucciones almacenadas en la memoria del sistema, realizando operaciones como sumas, restas, multiplicaciones, divisiones, comparaciones, y traslación de de datos.
- Arquitectura: Existen diferentes arquitecturas de CPU, como x86, ADM, PowerPC, entre otras, que varían en su diseño interno y conjunto de instrucciones. Estas arquitecturas determinan cómo se procesan las instrucciones y se manejan los datos.
- Velocidad de reloj, medida en hercios (Hz), gigahercios (GHz) o incluso terahercios (THz): Determina la cantidad de instrucciones que la CPU puede ejecutar por segundo. Una velocidad de reloj más alta generalmente se traduce en un mejor rendimiento del sistema. Por ejemplo una CPU con una velocidad de 1 Megahercio es capaz de realizar un millón (10^6) operaciones en cada segundo.

Las CPUs modernas constan normalmente de una memoria caché integrada, que es una memoria de acceso rápido utilizada para almacenar datos e instrucciones que se utilizan con frecuencia. La caché ayuda a mejorar el rendimiento del sistema al reducir los tiempos de acceso a la unidad de memoria general.

1.1.2. Memoria

La memoria en un ordenador es un componente crucial que se utiliza para almacenar temporal o permanentemente datos e instrucciones que la CPU (Unidad Central de Procesamiento) necesita para realizar operaciones. En función de si es de ”solo lectura” o si es de ”lectura y escritura” la memoria se denomina ROM o RAM.

- La ROM (Read Only Memory) es la memoria de ”solo lectura” y almacena instrucciones y datos de forma permanente. El término ”solo lectura” significa que los datos almacenados en ella no pueden ser modificados o borrados una vez que han sido programados en la memoria durante la fabricación. Una cualidad de la memoria ROM es que dichos datos no desaparecen ni se pierden en el caso de que se vaya la luz o se agote la batería (por ejemplo), es un tipo de memoria ”no volátil”. La memoria ROM se utiliza a menudo para almacenar el código de inicio o de arranque del sistema operativo de un ordenador, que es fundamental para iniciar el proceso de arranque del dispositivo y cargar el sistema operativo en la memoria RAM.
- La memoria RAM (Random Access Memory) es un tipo de memoria volátil que se utiliza para almacenar temporalmente datos y programas que se van a usar o están en uso actualmente. Cuando se lee un fichero en el ordenador, los datos necesarios se transfieren desde el almacenamiento permanente (como un disco duro o un SSD) a la memoria RAM para que la CPU pueda acceder a ellos rápidamente. La RAM es mucho más rápida que los dispositivos de almacenamiento permanentes, lo que permite a la CPU acceder y manipular los datos de manera más eficiente. La memoria RAM permite un acceso aleatorio a cualquier ubicación de memoria, lo que significa que la CPU puede acceder a cualquier dato almacenado en la memoria RAM en cualquier momento, sin tener que recorrer secuencialmente la memoria. La RAM es mucho más rápida que el almacenamiento de datos no volátil, lo que permite un acceso rápido a los datos utilizados con frecuencia por el procesador. Esto contribuye a un rendimiento más ágil del sistema en general. Sin embargo, la RAM es volátil, lo que significa que su contenido se borra cuando se apaga el ordenador o se pierde la alimentación eléctrica. Esto contrasta con

el almacenamiento no volátil, como la ROM vista anteriormente, los discos duros o las unidades de estado sólido (SSD), que retienen los datos incluso cuando el dispositivo está apagado.

Una de las características principales de la memoria (junto con la velocidad de acceso a los datos que contiene) es su capacidad, que indica la cantidad de datos que puede almacenar. La capacidad de almacenamiento se mide en **bits** (bit: binary unit). Los bits pueden tener solamente dos valores, 0 y 1, que, al combinarse múltiples veces, representan la información deseada. Los bits suelen agruparse en grupos de ocho, 1 byte = 8 bits, y sus múltiplos. Por ejemplo, 1KiloByte son 1024 bytes o 2^10 bits. En solamente un byte se pueden representar por lo tanto $2^8 = 256$ posibles valores, por ejemplo un conjunto de números enteros desde 1 a 256. La codificación de los números depende del sistema de representación que se explica más abajo.

1.2. Software y Lenguajes de Programación

El software es un conjunto de programas y datos diseñados para realizar tareas específicas en un ordenador e incluye aplicaciones (o programas), sistemas operativos, controladores de dispositivos y utilidades entre otros. Es la parte no tangible de un sistema informático que permite a los usuarios interactuar con el hardware y realizar diversas funciones.

Los lenguajes de programación son la base sobre la cual se desarrolla el software. Estos lenguajes proporcionan la sintaxis y las estructuras necesarias para implementar los algoritmos y la lógica de los programas. En general, un lenguaje de programación es un conjunto de reglas y símbolos que permiten escribir instrucciones que un ordenador puede entender y ejecutar. Estas instrucciones se conocen como código fuente y se escriben en un formato legible para los humanos. Luego, el código fuente se traduce a lenguaje máquina, que es el conjunto de instrucciones binarias que la computadora puede entender y ejecutar.

Existen muchos tipos de lenguajes de programación, cada uno con sus propias características y usos específicos. Algunos de los tipos más comunes de lenguajes de programación incluyen:

1. Lenguajes de programación de bajo nivel. Estos lenguajes están diseñados para interactuar de manera eficiente con el hardware y suelen ofrecer un control más detallado sobre los recursos del sistema. Ejemplos son:

   - Lenguajes máquina: Son códigos binarios que representan instrucciones directamente ejecutables por la CPU.
   - Lenguaje ensamblador: El lenguaje ensamblador es un tipo de lenguaje de programación que representa instrucciones de máquina mediante abreviaciones o símbolos fácilmente entendibles por los humanos, pero que están directamente relacionadas con las operaciones y estructuras de la arquitectura del procesador.

2. Lenguajes de programación de alto nivel. Estos lenguajes son más fáciles de entender y de escribir para los humanos, ya que están más alejados del lenguaje máquina. Suelen ofrecer abstracciones más potentes y facilidades para el desarrollo de software. Ejemplos incluyen MATLAB, Python, Java, C++, o JavaScript, Ruby, PHP, entre otros.

Los lenguajes de programación pueden ser interpretados o compilados. Los **lenguajes interpretados**, como MATLAB, Python o JavaScript, se ejecutan directamente por un intérprete sin necesidad de compilar el código previamente, mientras que los lenguajes compilados, como C++ o Java, se traducen a lenguaje máquina antes de ser ejecutados, lo que puede resultar en un rendimiento más rápido. Esto se debe a que, en un lenguaje compilado, el código fuente se traduce completamente a código de máquina antes de la ejecución, lo que permite que el programa se ejecute de manera más eficiente ya que las instrucciones están directamente optimizadas para la arquitectura del procesador.

Para traducir un lenguaje de alto nivel al lenguaje-máquina ser usa generalmente un compilador o un intérprete. El **compilador** es un programa informático que traduce el código fuente al lenguaje máquina ejecutable y funciona en varias etapas, incluyendo análisis léxico, sintáctico y semántico, generando un archivo ejecutable que puede ser utilizado independientemente del código fuente original. El **interprete** en cambio, ejecuta instrucciones escritas en un lenguaje de programación de alto nivel directamente sin necesidad de compilarlas previamente, traduciéndolas y ejecutándolas línea por línea en tiempo real. Esto permite una interacción más dinámica y flexible con el código, pero se traduce en ejecuciones más lenta en comparación con los programas compilados.

En esta asignatura vamos a comenzar estudiando un lenguaje considerado de nivel medio, **el lenguaje C**, para posteriormente estudiar un lenguaje de nivel alto interpretado, **usando MATLAB**. El lenguaje C es un lenguaje apropiado para iniciarse en la Física Computacional dado que, aunque proporciona un buen nivel de abstracción respecto al hardware subyacente, también permite un control más directo sobre la memoria y el hardware en comparación con los lenguajes de alto nivel, lo que lo coloca en una posición intermedia entre los lenguajes de alto y bajo nivel y apropiada para aprender los fundamentos de programación que se persiguen en esta asignatura.

1.3. Tipos de Datos

Los ordenadores trabajan con varios tipos de datos para almacenar y manipular información. Algunos de los tipos de datos comunes incluyen:

- Enteros (Integer): Representan números enteros sin decimales, como -3, 0, 42.
- Flotantes (Floating-point): Representan números con decimales, como 3.14, -0.001, 2.5.
- Caracteres (Character): Representan caracteres individuales, como ’a’, ’B’, ’5’.
- Cadenas de caracteres (String): Son secuencias de caracteres, como "Hola mundo", "12345".
- Booleanos (Boolean): Representan valores de verdadero (true) o falso (false).
- Arreglos (Array): Colecciones ordenadas de elementos del mismo tipo de datos.
- Estructuras (Structures): Agrupaciones de datos de diferentes tipos bajo un único nombre.
- Punteros (Pointers): Almacenan direcciones de memoria, permitiendo el acceso indirecto a los datos.

- Enumeraciones (Enum): Conjuntos de constantes con nombres simbólicos.

Dependiendo del lenguaje de programación y del entorno de desarrollo, pueden haber más tipos de datos específicos disponibles.
Las estructuras de datos son formas específicas de organizar y almacenar datos para que puedan ser utilizados eficientemente por el ordenador. Ejemplos de estructuras de datos incluyen listas, pilas, colas, árboles o grafos, cada uno diseñado para diferentes tipos de operaciones y situaciones. Estas estructuras pueden ser simples o complejas, dependiendo de la naturaleza de los datos y las operaciones que se deseen realizar. El concepto de datos estructurados se refiere a la combinación de varios tipos de datos en una sola estructura, lo que permite representar de manera más eficiente información compleja y relaciones entre datos.

1.4. Representación de la información

Ahora bien, **¿como se convierten todos estos tipos de datos a bits y a bites?**.

Existen diferentes sistemas de numeración utilizados para representar la información. Entre ellos, el más común y familiar para las personas el sistema **Decimal o en base 10**. Este sistema se compone de diez símbolos diferentes: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9. Así cadaa posición en un número decimal representa una potencia de diez.
Por ejemplo, cualquier número entero se puede representar por la fórmula $N_{(10)} = a_k 10^k + a_{k-1}10^{k-1} + ... + a_0 10^0$. Por ejemplo el número 42567 se escribe en base 10 como:$42567_{(10)} = 4\cdot 10^4 + 2\cdot 10^3 + 5\cdot 10^2 + 6\cdot 10^1 + 7\cdot 10^0$. Para representar números decimales lo único necesario es dejar a k tomar valores negativos.

El sístema **binario o en base 2** es el que trabaja directamente con los valores que puede tener un bit, es decir, que se compone únicamente de dos símbolos: 0 y 1. Cada posición en un número binario representa una potencia de dos: $N_{(2)} = b_k \cdot 2^k + b_{k-1}\cdot 2^{k-1} + ... + b_0 \cdot 2^0$, donde b_i solo puede tomar como valores $0, 1$. Así el numero 1 en binario sería $1_{(2)} = 1$, el 2, $2_{(2)} = 10$, el 3, $3_{(2)} = 11$ y así sucesivamente.
En general, para convertir un número decimal a binario, puedes seguir estos pasos:

1. División sucesiva por 2: Se divide el número decimal entre 2 y se anota el cociente y el residuo. El residuo será el bit menos significativo del número binario, mientras que el cociente se utilizará para la siguiente división. Se repite este proceso hasta que el cociente sea 0.

2. Reorganización de los residuos: se leen los residuos de abajo hacia arriba. Estos residuos forman el número binario, donde el primer residuo es el bit menos significativo y el último es el bit más significativo.

Veamos un ejemplo con el número decimal 13:

- Paso 1:

$$13 \div 2 = 6, \text{ residuo 1.}$$
$$6 \div 2 = 3, \text{ residuo 0.}$$
$$3 \div 2 = 1, \text{ residuo 1.}$$
$$1 \div 2 = 0, \text{ residuo 1.}$$

- Paso 2:

Los residuos, leídos de abajo arriba son 1, 1, 0, 1.

Por lo tanto, **el número 13 en base 2 se representa como 1101**. Es importante recordar que el bit menos significativo se encuentra a la derecha y el más significativo a la izquierda.

Otro sśitema de representación ampliamente utilizado en la programación es el sśitema **Hexadecimal (o en base 16)**. Se compone de dieciséis símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F (donde A representa el valor 10, B el valor 11, y así sucesivamente hasta F que representa el valor 15). Cada posición en un número hexadecimal representa una potencia de dieciséis. La conversión entre estos sistemas numéricos es importante en informática. Para convertir un número hexadecimal a decimal, se asigna primero un valor decimal a cada dígito hexadecimal. Después se multiplica cada dígito hexadecimal por 16 elevado a la potencia correspondiente según su posición (empezando desde la derecha y contando desde 0) y finalmente se suman los productos obtenidos en el paso anterior. Por ejemplo, para convertir el número hexadecimal 2A a decimal:

- Paso 1: Asignamos valores decimales a los dígitos hexadecimales: 2 en hexadecimal es igual a 2 en decimal y A en hexadecimal es igual a 10 en decimal.
- Paso 2: Multiplicamos cada dígito por 16 elevado a la potencia correspondiente: $2 \cdot 16^1 = 2 \cdot 16 = 32$ y $A \cdot 16^0 = 10 \cdot 1 = 1$.
- Paso 3: Sumamos los productos: $32 + 10 = 42$.

Por lo tanto, el número hexadecimal 2A es igual a 42 en decimal.

Otro sístema, utilizado con menos frecuencia en la informática moderna es el Octal (Base 8), que se compone de ocho símbolos: 0, 1, 2, 3, 4, 5, 6 y 7, donde cada posición en un número octal representa una potencia de ocho.

El almacenamiento de los diferentes tipos de datos en los bits se realiza utilizando distintos métodos y formatos de representación, dependiendo del tipo de dato y del contexto en el que se utilice. Por ejemplo, los números enteros se convierten a bits utilizando la representación binaria de su valor. Evidentemente, se utiliza un cierto número de bits para representar el número, dependiendo del rango de valores que se espera manejar. Para otro tipo de números hay que aplicar procedimientos más complejos.
Por ejemplo, un número en coma flotante consta de varias partes que representan diferentes aspectos del número: el signo, la mantisa, el exponente y la base. El signo indica si el número es positivo o negativo. Por lo general, se representa con un bit, donde 0 indica positivo y 1 indica negativo. La mantisa (también conocida como fracción o significando) representa la precisión del número. Es una fracción binaria normalizada que va desde 1 hasta (1 - ϵ), donde ϵ es el error de redondeo. El exponente determina la magnitud del número y es un valor entero que ajusta la posición del punto decimal de la mantisa. Finalmente, la base hace referencia al sístema en la que esta representado el número. Estas partes trabajan juntas para representar números reales con una cantidad fija de bits. La combinación de la mantisa y el exponente permite representar números muy grandes o muy pequeños con precisión, mientras que el bit de signo indica si el número es positivo o negativo.

Los números en coma flotante se representan utilizando el estándar IEEE 754. Este formato incluye un número de bits para la mantisa y el exponente, junto con un bit de signo para

indicar el signo del número. Esta norma establece dos formatos básicos para representar los números reales: simple precisión y doble precisión, asignando un número específico de bits al exponente y a la mantisa. Otros datos, como por ejemplo ”los caracteres” se representan en binario utilizando otras reglas acordadas, como ASCII (American Standard Code for Information Interchange) o Unicode. En este caso, cada carácter se asigna a un valor numérico específico, que luego se convierte a su representación binaria. Por ejemplo, el carácter ”A” se representa en binario como 01000001 en ASCII.

1.5. Operaciones entre datos

Para combinar la información presente en los datos, el ordenador puede realizar principalmente operaciones aritméticas y lógicas. Ambos tipos de operaciones son fundamentales en la programación y se utilizan en una variedad de contextos para manipular datos y controlar el flujo de un programa.
Las operaciones Aritméticas son bien conocidas, y engloban por ejemplo:

- Suma (+): Combina dos valores numéricos para obtener un resultado total. Por ejemplo, $2 + 3 = 5$.
- Resta (-): Substrae un valor numérico de otro para obtener un resultado. Por ejemplo, $5 - 3 = 2$.
- Multiplicación (*): Realiza la repetición de sumas sucesivas de un número por otro. Por ejemplo, $2 * 3 = 6$.
- División (/): Divide un valor numérico por otro para obtener un resultado. Por ejemplo, $6/2 = 3$.
- etc...

Otro tipo son las operaciones lógicas, aquellas que se utilizan para evaluar condiciones y tomar decisiones en función del valor de expresiones booleanas, es decir, valores que pueden ser verdaderos o falsos. Estas operaciones se basan en el álgebra booleana y son fundamentales en la programación para controlar el flujo de un programa y realizar decisiones lógicas.:

- **AND,** $\&$: Devuelve verdadero (true) si ambos operandos son verdaderos. Por ejemplo, A AND B es verdadero solo si A es verdadero y B es verdadero. En binario: $1\&1 = 1$, $1\&0 = 0$, $0\&1 = 0$ y $0\&0 = 0$.
- **OR,** $|$: Devuelve verdadero si al menos uno de los operandos es verdadero. Por ejemplo, A OR B es verdadero si A es verdadero o B es verdadero o ambos son verdaderos. En binario: $1|1 = 1$, $1|0 = 1$, $0|1 = 1$ y $0|0 = 0$.
- **NOT**: Devuelve el inverso lógico del operando. Por ejemplo, NOT A es verdadero si A es falso, y es falso si A es verdadero. En binario: $NOT1 = 0$, $NOT0 = 1$.
- **XOR (OR exclusivo)**: Devuelve verdadero si solo uno de los operandos es verdadero, pero no ambos. Por ejemplo, A XOR B es verdadero si A es verdadero y B es falso, o si A es falso y B es verdadero, pero es falso si ambos son verdaderos o ambos son falsos. En binario: $1XOR1 = 0$, $0XOR0 = 0$ y $0XOR1 = 1XOR0 = 1$

Estas operaciones son fundamentales en la programación y se utilizan para realizar cálculos y tomar decisiones basadas en condiciones lógicas. Son parte integral de los lenguajes de programación y se utilizan en una variedad de contextos, desde la manipulación de datos hasta el control del flujo de un programa.

1.6. Ficheros y Carpetas

Los ficheros (o archivos) y carpetas son elementos fundamentales en el almacenamiento de información de los ordenadores. Un archivo es una unidad de información que se almacena en un dispositivo de almacenamiento, como un disco duro o una unidad flash. Los archivos pueden contener datos de cualquier tipo, como texto, imágenes, audio, video o código de programa y deben de estar identificados por un nombre único y una extensión que indica su tipo de contenido (por ejemplo, .txt para archivos de texto, .jpg para imágenes, .mp3 para archivos de audio, etc.).
Una carpeta es una estructura de almacenamiento utilizada para organizar y agrupar archivos relacionados. Las carpetas pueden contener archivos y otras carpetas, creando una jerarquía de directorios que ayuda a mantener la información organizada y accesible. El ordenador realiza las siguientes operaciones sobre los archivos y carpetas:

- Creación: Es el proceso de generar un nuevo archivo en el sistema de archivos. Se le asigna un nombre único y se especifica su ubicación en la estructura de directorios.
- Apertura: Implica acceder a un archivo existente para su lectura o escritura. Durante esta operación, el sistema operativo asigna recursos para que el programa pueda trabajar con el archivo.
- Cierre: Es el proceso de liberar los recursos asociados con un archivo que se ha abierto. Esto es importante para evitar pérdidas de datos y asegurar la integridad del archivo.
- Lectura: Consiste en obtener datos almacenados en un archivo y cargarlos en la memoria del programa para su procesamiento.
- Escritura: Involucra la creación o actualización de datos en un archivo, donde los datos en la memoria del programa se escriben en el archivo en el disco.

Durante este curso se estudiarán las herramientas, tanto en C como en MATLAB, para llevar a cabo todas estas operaciones.