ALGORITMOS Y PROGRAMAS

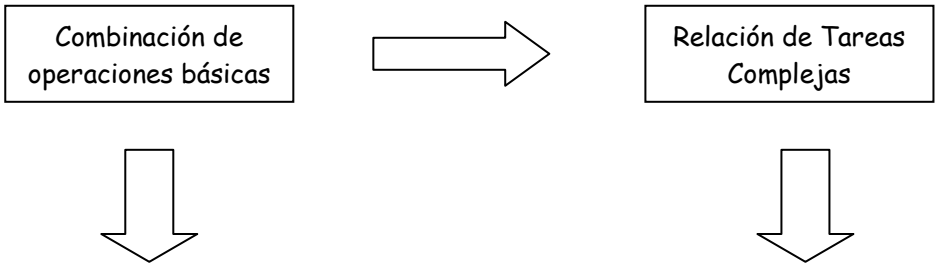
1.1.- El Ordenador y el Sistema Operativo

1.1.1.- El Ordenador

Al aparecer la Informática en los años 50 se abre una nueva era en el tratamiento de la información, que conlleva un cambio radical en la sociedad, al crear una nueva cultura que, no sólo se basa en la Informática como ciencia aplicada, sino que depende íntimamente de ella permitiendo avances tecnológicos impensables unos años antes.

A pesar de los conocimientos adquiridos y los adelantos conseguidos en los últimos años en el campo de la Informática, lo cierto es que el ordenador sigue siendo una máquina incapaz de actuar o realizar tareas por sí misma sin la ayuda del razonamiento humano.

Un ordenador sólo es capaz de ejecutar órdenes y de realizar operaciones básicas como sumas, restar, multiplicar, dividir valores numéricos, comparar valores numéricos o alfanuméricos y almacenar o recuperar información. Por ello, su potencia de cálculo dependerá básicamente de la eficacia, rapidez y precisión, así como de la memoria disponible. Con la combinación de estas operaciones básicas un ordenador puede llegar a realizar acciones o cálculos verdaderamente complejos, pero siempre existirá una estrecha dependencia de la máquina con el hombre, sin la cual el ordenador es una herramienta carente de utilidad.



En definitiva, ***un ordenador es una simple máquina capaz de aceptar datos, procesarlos y facilitar datos o resultados de salida***

Los datos son introducidos o recuperados del ordenador mediante los **periféricos** o elementos destinados a auxiliar a la **Unidad Central de Proceso (UCP),** en su relación con el mundo exterior, mediante los interfaces adecuados.

Según la función que desempeñen, los periféricos, respecto del ordenador al que están conectados, pueden dividirse en:

- ***Dispositivos de entrada.-*** Encargados de introducir datos a un sistema desde el mundo exterior: Teclado, ratón, lector de códigos de barras, etc.

- ***Dispositivos de salida.-*** Encargados de sacar al exterior los resultados obtenidos en un proceso u operación realizada por la Unidad Central: Pantalla, Impresora, Plotter, etc.

- ***Dispositivos de entrada y salida.-*** Capaces tanto de introducir como de extraer información de la Unidad Central de Procesos: Dispositivos de memoria masiva (Discos Duros, Disquetes, cintas magnéticas, etc.), etc.

1.1.2.- El Sistema Operativo

Sin software, un ordenador es inútil y no vale para nada. Es el software el que hace que el ordenador pueda almacenar, procesar, y recuperar información. Sin software, el hardware de un ordenador sería un conjunto de circuitos electromecánicos mas o menos complejos sin ninguna utilidad.

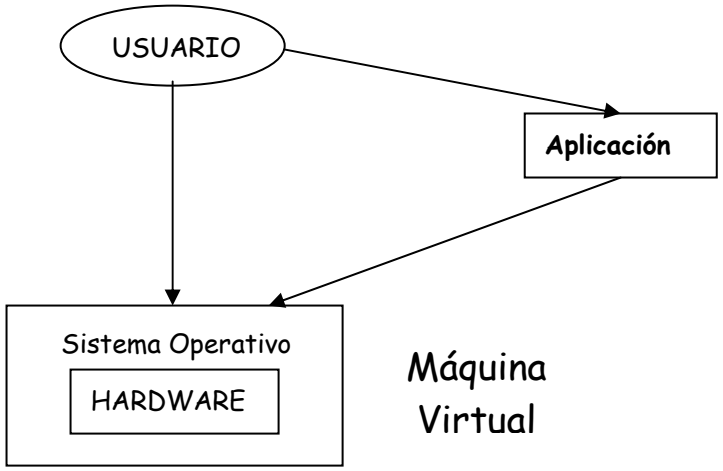
Para poder utilizar el hardware, el usuario precisa de un interfaz o capa de software que se encargue de gestionar todos los elementos del sistema y haga fácil el entendimiento del hardware. Esta capa de software tan ligada al hardware recibe el nombre genérico de ***Sistema Operativo (SO)***.

El SO es pues el software básico y fundamental de un sistema informático. Se hecho, algunas empresas de software desarrollan sus productos únicamente para ciertos tipos de sistemas operativos.

En una primera aproximación el Sistema Operativo es un tipo de software que controla el funcionamiento del elemento físico (hardware), ocultando sus detalles al usuario y permitiéndole así trabajar con el ordenador de una manera fácil y segura. Para ello, el SO enmascara los recursos físicos permitiendo un manejo sencillo y flexible de los mismos y proporciona determinados servicios de alto nivel al usuario que no están directamente presentes en el hardware subyacente.

En este sentido el SO hace posible la ejecución y uso de aplicaciones por parte del usuario. De esta manera, el SO, junto con el hardware (la máquina física) definen una *máquina virtual* que puede ser utilizada sin conocer las características concretas de los dispositivos hardware.

Desde este punto de vista, ***el SO es el interlocutor del usuario con el hardware, es decir, el grupo de programas que hacen accesible y utilizable el hardware***.

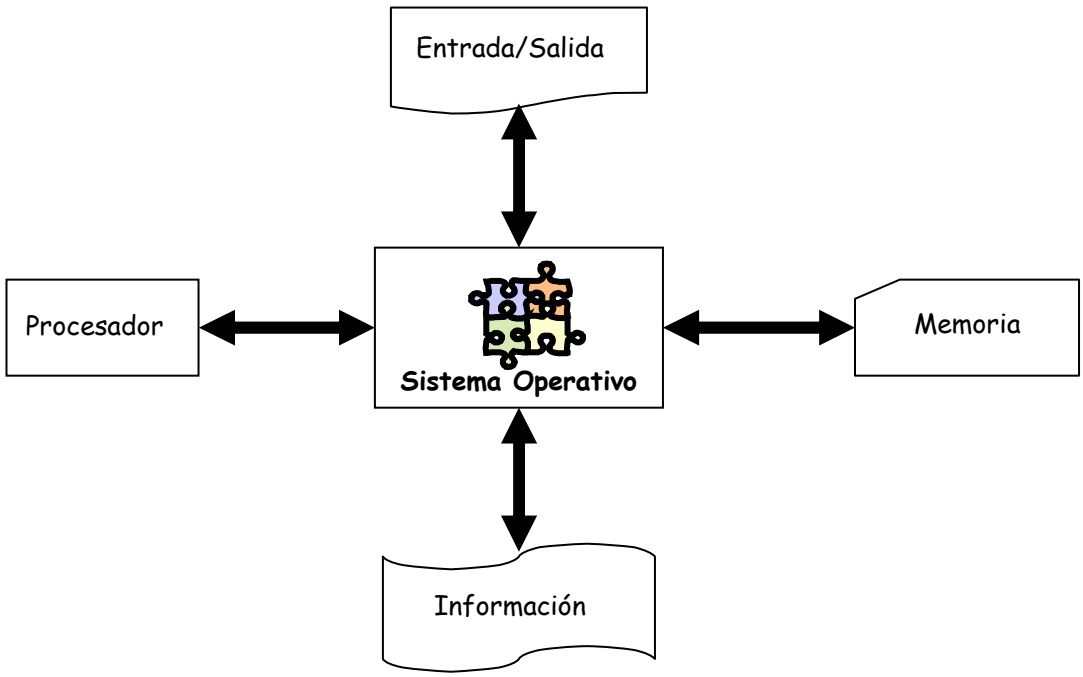


Por otra parte, el ordenador integra un conjunto de elementos necesarios para llevar a cabo su trabajo. Estos elementos, denominados ***recursos,*** deben ser racionalmente distribuidos y utilizados para obtener de ellos el mejor rendimiento. Los principales recursos del ordenador son:

- ***El procesador:*** dado que es el lugar donde se ejecutan las instrucciones de los programas de aplicación, se debe controlar la forma en que se ejecutan dichos programas
- ***La memoria principal:*** puesto que es el dispositivo donde residirán los datos a procesar y los programas a ejecutar, es preciso regular su uso y ocupación
- ***Los dispositivos periféricos:*** que permiten la comunicación de los programas con el exterior y debe, por tanto, asegurarse su adecuado direccionamiento y control
- ***La información:*** Los datos, sus tipos, tamaños y métodos de representación tienen que estar perfectamente controlados para evitar operaciones erróneas o falsas interpretaciones

Teniendo en cuenta la necesidad de controlar los recursos, ***el SO es el administrador de los recursos ofrecidos por el hardware de cara a conseguir un uso eficiente de los mismos***

El Sistema Operativo es, pues, un conjunto de programas de control que persiguen obtener un buen aprovechamiento de los recursos hardware y facilitar el uso del ordenador a los usuarios



Sin un sistema operativo un ordenador nunca podrá empezar a funcionar, por tanto, lo primero que ha de ocurrir cuando se pone en marcha un ordenador es la carga, en la memoria principal, del sistema operativo. De hecho, cuando se enciende un ordenador se ejecuta un programa de *autodiagnóstico de encendido* que identifica todos los dispositivos

hardware conectados a la UCP. Posteriormente se ejecuta el *cargador inicial* (Power On Self Test: POST) que, a su vez, carga un programa de autoarranque mas eficiente, que busca el Sistema Operativo y carga parte del mismo (*parte residente*) en la memoria principal.

Una vez que el ordenador ha puesto en marcha su Sistema Operativo, mantiene parte de él en su memoria en todo momento.

Las tareas que desempeña el Sistema Operativo dependen, en cierta medida, del tipo de funcionamiento del ordenador (multitarea, monotarea, etc.). No obstante existen ciertos puntos comunes que pueden considerarse en todos los Sistemas Operativos. De hecho, mientras un ordenador permanezca encendido, el Sistema Operativo ha de desempeñar las siguientes funciones:

- ***Facilitar la tarea del usuario.*** Para ello hace uso de un intérprete de órdenes (***Shell***) que proporciona una interfaz con el usuario, para que éste último pueda comunicarse con el ordenador.
- ***Administrar los dispositivos hardware del ordenador***
- ***Administrar y mantener el sistema de ficheros*** en dispositivos de memoria masiva
- ***Apoyar a otros programas***
- ***Proteger los datos y los programas***
- ***Contabilizar el uso de los recursos*** realizado por los distintos usuarios.

1.1.2.1.- Sistemas operativos multitarea

Los primeros sistemas operativos se denominaban ***monotarea o serie;*** en ellos, hasta que no finalizaba la ejecución de un programa no empezaba a ejecutarse otro. Como es natural, el rendimiento alcanzado con los sistemas operativos monotarea era bastante bajo, debido a que existían tiempos muertos del procesador durante los cuales no realizaba ningún trabajo útil (por ejemplo, cuando se producían operaciones de E/S).

Por contra, los sistemas operativos ***multitarea*** (o ***multiprogramados***) son capaces de ejecutar más de un programa al mismo tiempo. Estos últimos se basan fundamentalmente en las técnicas de multiprogramación y son los más extendidos en la actualidad.

Existen dos tipos de sistemas operativos multitarea, dependiendo de cómo lleven a cabo la gestión del procesador:

- ***Multitarea cooperativa:*** En este esquema existe una cooperación entre el sistema operativo y los programas de aplicación. Los procesos realizan periódicamente una inspección de cara a detectar si otro proceso necesita el procesador. Si ocurre esto, el proceso en ejecución deja el control del procesador al siguiente programa. Con este esquema, el sistema operativo no toma el control del procesador de forma autónoma a la hora de decidir qué programa se ejecutará, sino que depende de lo que determine la aplicación que se esté ejecutando. Este método es el utilizado por el sistema operativo DOS ejecutando Windows de Microsoft.

- ***Multitarea con asignación de prioridades:*** Con este esquema, el sistema operativo mantiene una lista de los procesos que intervienen en cada momento. Cuando se inserta un proceso en la lista, se le asigna una prioridad. En cualquier momento, el sistema operativo puede intervenir y modificar la prioridad de un proceso según las circunstancias. En este caso, el sistema operativo es el responsable de dar el control del procesador a un proceso o a otro según un esquema de prioridades.

1.1.2.2.- Sistemas operativos multiusuario

Los sistemas monousuario, son sistemas muy simples que sólo permiten la conexión de un único usuario en un instante dado por lo cual no necesitan realizar la gestión y control de varios usuarios. Estos sistemas pueden ser monotarea o multitarea.

Frente a éstos, los sistemas operativos ***multiusuario*** permiten que más de un usuario acceda a la computadora al mismo tiempo. Naturalmente para llevar a cabo esto, el sistema operativo debe ser también multitarea y debe establecer mecanismos de identificación, autentificación y control de los distintos usuarios.

Los sistemas multiusuario permiten, además, que cada usuario pueda ejecutar varios programas simultáneamente (ya que son multitarea).De esta forma, se pueden aprovechar al máximo los recursos del sistema informático, obteniéndose un alto rendimiento. UNIX es un ejemplo claro de sistema multitarea y multiusuario desde su concepción.

1.1.2.3.- Sistemas operativos multiproceso

La evolución de los sistemas informáticos ha propiciado que las exigencias de los usuarios crezcan de forma vertiginosa. Actualmente, se piden grandes capacidades de procesamiento a los ordenadores, Lo cual ha obligado a la aparición de sistemas informáticos que contienen dos o más procesadores interconectados trabajando simultáneamente formando un único ordenador. Estos sistemas son denominados **sistemas multiprocesador**.

Es necesario un tipo especial de sistema operativo para gestionar un sistema multiprocesador. En este caso, el sistema operativo debería ser capaz de administrar el reparto de trabajo entre lo distintos procesadores para sacar provecho del paralelismo existente. A este tipo de sistemas operativos se les denomina ***multiproceso o de multiprocesamiento.***

Los primeros sistemas operativos dentro de esta categoría utilizaban lo que se denomina ***multiproceso asimétrico***, que se caracteriza porque un procesador principal controla el comportamiento global de todos los demás, utilizándolos como si fueran otros dispositivos conectados al bus del ordenador. Este tipo de multiproceso presenta un serio problema, el procesador principal puede sobrecargarse en las tareas de administración (puede convertirse en el cuello de botella). Además, con esto esquema, la adición de procesadores no lleva consigo un aumento lineal de las prestaciones.

El ***multiproceso simétrico***, por otro lado, permite que la capacidad del sistema aumente linealmente por cada procesador adicionado En ese caso no existe un procesador controlador único. El problema de este esquema es que su implementación impone la

actualización y rediseño de los sistemas operativos y de los compiladores para trabajar en un ambiente multiproceso.

Existen versiones de UNIX para procesamiento asimétrico, y se están desarrollando versiones para procesamiento simétrico. Windows NT admite procesamiento simétrico.

1.1.2.4.- Sistemas operativos de tiempo real

Existen aplicaciones que son especialmente sensibles al factor tiempo, ya que dependen del tiempo exacto. Estas aplicaciones se denominan ***de tiempo real***, para este tipo de aplicaciones es crucial no perder ni un segundo, ya que en caso contrario la información se puede perder o distorsionar. El concepto de tiempo real hace referencia a que el ordenador debe dar la respuesta dentro de un límite preestablecido. El control de este tipo de procesamiento es llevado a cabo por sistemas operativos denominados de tiempo real.

Los sistemas operativos de tiempo real son ampliamente usados en aplicaciones capaces de controlar y regular el medio en que operan, como pueden ser control industrial, control de vuelo, aplicaciones militares. etc. Estos sistemas deben ser capaces de responder a determinados eventos (***interrupciones***) que se pueden producir asíncronamente y con gran frecuencia en unos plazos de tiempo previamente especificados (existen fuertes restricciones en el tiempo de respuesta). Usualmente, son sistemas multitarea que utilizan algoritmos de planificación de derecho preferencial, con los cuales siempre se ejecuta la tarea más prioritaria sin interrupción, a no ser que se genere otra tarea más prioritaria.

1.2.- Sistemas de Procesamiento de Información

Normalmente, la información se le suministra a un ordenador en la forma mas usual para los humanos, es decir, con ayuda de ***caracteres***. Estos caracteres puedes ser ***alfabéticos*** (tanto mayúsculas A, B, C, … , Z como minúsculas a, b, c, … , z), ***numéricos*** (0, 1, 2, 3, … , 9), ***especiales*** (;, :, !, ", $, %, &, /, (,), =, etc.) de ***control*** (órdenes de control, como carácter indicador de fin de línea, carácter indicador de fin de fichero, etc.) y ***caracteres gráficos*** (que permiten representar figuras elementales: ‖, ╠, ╦, ▣, etc.).

Nos centraremos en los tres primeros tipos de caracteres (***caracteres-texto***). La agrupación de los dos primeros tipos se denomina caracteres ***alfanuméricos***.

Toda comunicación con el ordenador se ha de realizar según los caracteres que admitan sus dispositivos de E/S (todo dato o instrucción se representará con el alfabeto definido para el sistema informático). El conjunto de caracteres codificable en un ordenador se denomina ***juego de caracteres*** de dicho ordenador.

El diseño y construcción de un ordenador se simplifica, aumentando su fiabilidad, si se utilizan sólo dos valores posibles para las variables físicas que representan los datos en el interior de la computadora, Estos valores se representan por 0 o 1, y corresponden a dos niveles de tensión, de corriente, etc. En definitiva, la información se mantiene utilizando dos valores de una magnitud física (bit) representable mediante ceros y unos.

Al tener que traducir toda la información suministrada por la computadora a ceros y unos, es necesario establecer una ***correspondencia*** entre el conjunto de todos los caracteres y el conjunto binario $\{0,1\}^n$ (donde n es el número de bits disponibles para representar los caracteres). En definitiva, es necesario llevar a cabo una codificación entre

los elementos del primer conjunto mediante los del segundo. Estos códigos de transformación se denominan ***códigos de E/S*** y pueden definirse de forma arbitraria, aunque existen códigos de E/S normalizados.

Por ejemplo, si se quiere representar las cinco primeras letras mayúsculas del alfabeto, puede establecerse la siguiente correspondencia:

Carácter	**Código**
A	100
B	011
C	110
D	111
E	101

Como se ve en este ejemplo, se ha establecido una correspondencia entre el conjunto {A, B. C, D, E} y el conjunto $\{0,1\}^3$. Aquí se ha utilizado el número mínimo de bits necesarios para representar estas 5 letras, es decir, tres (en este caso, n debe ser al menos 3).

Para realizar las operaciones aritméticas sobre datos numéricos, la propia computadora efectúa una transformación de la representación en códigos de E/S a una representación basada en el sistema de numeración en base 2 que resulta, como se verá, más adecuada para este objetivo (una representación numérica posicional es muy apta para realizar operaciones aritméticas).

1.2.1.- Los Sistemas de Numeración

Los ordenadores suelen efectuar las operaciones aritméticas utilizando una representación para los datos numéricos basada en el ***sistema de numeración en base dos*** (*binario natural*). También se utilizan como ***códigos intermedios*** los sistemas octal y hexadecimal, ya que, como veremos, el paso de un número de binario a estos sistemas es trivial, con lo cual se obtiene una representación que está mis cerca del sistema decimal (se utilizan como paso intermedio en las conversiones binario-decimal y decimal-binario).

1.2.1.1.- Representación posicional de los números

Un ***sistema de numeración en base b*** implica la existencia de un alfabeto A compuesto por b símbolos. El valor de un número (en base b) depende de la secuencia concreta de cifras del alfabeto, contribuyendo cada cifra con un valor dependiente de ella misma y de su posición dentro del número.

Ejemplo: En el sistema decimal (b = 1O, A = {O, 1,…, 9}) ocurre que:

$$458.23)_{10} = 4\cdot 10^2 + 5\cdot 10^1 + 8\cdot 10^0 + 2.10^{-1} + 3\cdot 10^{-2}$$

Como vemos, cada posición del número tiene un peso y un nombre específico (decenas, centenas, unidades, etc.). Generalizando, si la expresión de un número N en base b es:

$$N)_b = \ldots n_3 n_2 n_1 n_0 n_{-1} n_{-2} n_{-3} \ldots$$

Entonces su valor decimal viene dado por:

$$N)_{10} = \ldots + n_3 \cdot b^3 + n_2 \cdot b^2 + n_1 \cdot b^1 + n_0 \cdot b^0 + n_{-1} \cdot b^{-1} + n_{-2} \cdot b^{-2} + n_{-3} \cdot b^{-3} + \ldots$$

Este resultado, conocido como el ***teorema fundamental de la numeración***, relaciona una cantidad expresada en cualquier sistema de numeración con la misma cantidad expresada en el sistema decimal.

Ejemplo: El código octal está basado en un alfabeto de 8 símbolos (A = {O, l, 2, 3, 4, 5, 6, 7} y b = 8). El número octal $l65,4)_8$ tiene una representación decimal que viene dada por:

$$165,4)_8 = 1 \cdot 8^2 + 6 \cdot 8^1 + 5 \cdot 8^0 + 4 \cdot 8^{-1} = 117,5)_{10}$$

1.2.1.2.- Sistema binario

Las operaciones aritméticas dentro de un ordenador se suelen llevar a cabo utilizando una representación para los datos basada en el binario natural. Aunque el cambio de código de E/S a la representación en binario natural la realiza automáticamente el ordenador, veremos una serie de cuestiones relativas al sistema binario y a las transformaciones entre éste y el sistema decimal.

El sistema de numeración binario se basa en el uso de un alfabeto A de sólo dos símbolos (b = 2, A = {0,1}). Los elementos de un alfabeto binario A se llaman bits o cifras binarias.

Las transformaciones entre el sistema decimal y el sistema binario se realizan de la forma siguiente:

- **Transformaciones de binario a decimal**

Para transformar un número representado en el sistema binado a su representación en decimal, únicamente hay que aplicar el teorema fundamental de la numeración, visto en el apartado anterior.

Por ejemplo, para pasar el número binario $110010,101)_2$ a decimal, simplemente hemos de aplicar directamente el teorema visto, utilizando b = 2:

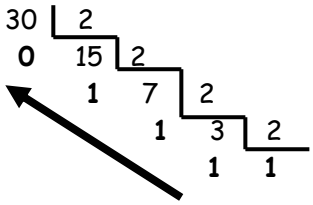
$$110010,101)_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3})_{10} = 50,625)_{10}$$

- **Transformaciones de decimal a binario**

Para transformar un número de decimal a binario, se transforman de forma independiente la parte entera y la parte fraccionaria del número, y mas tarde se concatenan los resultados obtenidos.

Para la ***conversión de la parte entera*** basta con dividir por 2 la parte entera del número decimal, y después realizar divisiones por 2 de los cocientes sucesivos hasta llegar a un cociente menor que 2. El número binario obtenido como resultado tendrá como bit más a la izquierda el último cociente obtenido en el proceso de divisiones sucesivas. El resto de los bits del número binario estaría formado por los restos de las divisiones, comenzando por Los últimos restos obtenidos (por tanto, el bit más a la derecha sería el primer resto obtenido).

Así, la representación en binario del número decimal $30)_{10}$ es $11110)_2$, ya que:



La ***conversión de la parte fraccionaria*** se realiza multiplicando por 2 la parte fraccionaria de número decimal y después realizar la misma operación en cadena con las partes fraccionarias de los resultados obtenidos en los productos sucesivos. El proceso finaliza (suponiendo que el número binario tiene una parte fraccionaria finita) cuando se llega a un resultado con parte fraccionaria nula. El número binario se forma con las partes enteras de los productos obtenidos, leídos en el mismo orden en que se obtienen.

El número de cifras fraccionarias del número binario puede ser muy grande o infinito, por lo cual, internamente, se suele truncar el resultado de la conversión, dependiendo de la capacidad de almacenamiento del ordenador (existe, por tanto lo que se llama *error de truncamiento*).

Así, la representación en binario del número decimal $0{,}325)_{10}$ es $0{,}01010\ldots)_2$ ya que:

0,325	0,650	0,3	0,6	0,2	
X 2	X 2	X 2	X 2	X 2	
0,650	**1**,3	**0**,6	**1**,2	**0**,4	… … …

En la tabla adjunta se muestra la representación en binarios de cada una de las 10 cifras que integran el alfabeto del sistema de numeración tradicional:

Decimal	***Binario***
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

9	1001

Por último, las operaciones aritméticas básicas que pueden realizarse con números binarios son la suma, resta multiplicación y división. Para efectuar este tipo de operaciones, se emplea el mismo algoritmo que el utilizado en el sistema decimal. Lo único que se debe tener en cuenta es que se utilizan otras tablas para estas operaciones:

		SUMA	***RESTA***	***MULTIPLICACIÓN***	***DIVISION***
A	**B**	**A + B**	**A - B**	**A * B**	**A / B**
0	0	0	0	0	Indeterminado
1	0	1	1	0	∞
0	1	1	1 y debo 1	0	0
1	1	0 y me llevo 1	0	1	1

1.2.1.3.- Sistema octal

Es el sistema de numeración cuya base es 8 por lo que cada cantidad está representada por los guarismos arábigos comprendidos entre el 0 y el 7. Análogo a los anteriores, es un sistema secuencial, por lo que, por aplicación del teorema fundamental de la numeración, cualquier cantidad representada en el sistema octal estará formada por N guarismos de representación X_i (X = 0,1,2,…,7 con i = 0,1,2,…,N) según:

$$X_1*8^{N-1} + X_2*8^{N-2} + \ldots + X_N*8^{N-N}$$

Dado que 8 es potencia de 2 puede establecerse la tabla de correspondencia biunívoca representada en la tabla adjunta

El paso de una cantidad representada en sistema binario a sistema octal se realiza agrupando el binario de derecha a izquierda en grupos de tres dígitos y estableciendo la correspondencia mediante la tabla.

1.2.1.4.- Sistema hexadecimal

Es el sistema de numeración cuya base es 16. Es también un sistema secuencial en el que los dígitos están representados por los guarismos arábigos desde el 0 hasta el 9 y por las letras del abecedario mayúsculas desde la A hasta la F.

Análogamente a los casos anteriores, por aplicación del teorema fundamental de la numeración cualquier cantidad representada en el sistema hexadecimal estará formada por N guarismos de representación X_i (X = 0,1,2,…,8,9,A,B,…F con i = 0,1,2,…,N) según:

$$X_1*16^{N-1} + X_2*16^{N-2} + \ldots + X_N*16^{N-N}$$

Dado que 16 es potencia de 2 puede establecerse la tabla de correspondencia biunívoca representada en la tabla:

Decimal ***Octal*** ***Hexadecimal*** ***Binario***

0	0	0	00000
1	1	1	00001
2	2	2	00010
3	3	3	00011
4	4	4	00100
5	5	5	00101
6	6	6	00110
7	7	7	00111
8	10	8	01000
9	11	9	01001
10	12	A	01010
11	13	B	01011
12	14	C	01100
13	15	D	01101
14	16	E	01110
15	17	F	01111
16	20	10	10000

El paso de una cantidad representada en sistema binario a sistema hexadecimal se realiza agrupando el binario de derecha a izquierda en grupos de cuatro dígitos y estableciendo la correspondencia mediante la tabla anterior.

1.2.1.5.- Conversión de una cantidad de un sistema de base "N" a un sistema de base "M"

La conversión se realiza con el paso intermedio de pasar la cantidad de la base "N" inicial a la base decimal. El número decimal resultante se convierte posteriormente al sistema final de base "M". Por tanto:

Conversión de base "N" a decimal: Se realiza por aplicación directa del teorema fundamental de la numeración

Conversión de decimal a base "M": El método consiste en dividir sucesivamente el número decimal utilizando como divisor la base final M. El cociente entero obtenido en esta primera división se utiliza como nuevo dividendo, siempre y cuando sea mayor que el divisor (M). Se realizan divisiones sucesivas hasta que el dividendo sea menor que M. El número en la base M se formará tomando como primer guarismo el resultado del último cociente y los siguientes guarismos serán los distintos restos colocados en orden inverso a su obtención.

1.2.2.- La información y su representación

La representación binaria es la única forma que tiene un ordenador para recibir, tratar o almacenar información. Tanto si ésta es numérica como si es literal.

Por lo que a cantidades se refiere, el ordenador considera un número limitado de bits para almacenar cada dato numérico. Este número, denominado ***Palabra***, es una característica de la computadora. Además, según sea la cantidad, entera o real, el ordenador utiliza diferentes formas de representación.

Se denomina ***Rango de Representación*** de un método de representación numérica al conjunto de números que pueden representarse mediante ese método

1.2.2.1.- Los Números Enteros

Para los números enteros, tanto positivos como negativos, el ordenador utiliza los siguientes métodos de representación:

Módulo y Signo

En función del número de bits para la palabra de la computadora (n), este método utiliza el bit situado mas a la izquierda para el signo (0 para el positivo y 1 para el negativo) utilizando el resto de los n-1 bits para el módulo del número. Al ser un número entero, no tiene parte fraccionaria, por lo que la coma decimal se supone implícitamente situada a la derecha de la palabra.

El Rango de representación de este método para un ordenador que utiliza palabras de n bits es

$$-2^{n-1} + 1 <= X <= 2^{n+1} - 1$$

El mayor inconveniente de este sistema de representación estriba en poseer dos representaciones para el número 0 que, en el caso de n = 8 se representa mediante 00000000 (+0) y 10000000 (-0).

Complemento a 1

El método consiste en representar los números positivos por el método de módulo y signo, y los números negativos complementando todos los dígitos del numero positivo correspondiente (esto es, cambiando todos los 0 por 1 y viceversa) incluido el bit del signo.

Su rango de representación es el mismo que el del método de módulo y signo.

Complemento a 2

El método consiste en representar los números positivos por el método de módulo y signo y los números negativos mediante los siguientes pasos:

- ★ Realizando el Complemento a 1 del número.
- ★ Al resultado obtenido, sumando, en binario, el valor 1 y despreciando el acarreo producido si es que existe.

Exceso a 2n-1

Este método no considera el signo de la cantidad ya que desplaza, mediante el exceso, el valor binario de la cantidad a la zona de los números positivos. Consiste en sumar a la cantidad representada en el sistema decimal un exceso que, para ordenadores con palabras de n bits viene representado por 2n-1 ,y el número decimal así obtenido, representarlo en binario, utilizando tantos dígitos cuantos bits tenga la palabra.

Su rango de representación es el mismo que para módulo y signo.

1.2.2.2.- Los Números Reales

Las cantidades formadas por una parte entera y una parte fraccionaria, se representan mediante dos métodos:

Representación en Coma fija

- *Binario puro*

Un número se representa utilizando el conjunto de bits equivalente a 1 palabra (simple precisión) o 2 palabras (doble precisión), representando en forma binaria el número, utilizando el bit mas a la izquierda para representar el signo y los n-1 restantes para representar el valor positivo del número mediante el método de complemento a 2 .

La coma decimal puede suponerse en cualquier lugar, fijo, distinto de la derecha de la palabra.

- *Decimal Desempaquetado*

Se basa en el sistema de representación BCD (Decimal Codificado en Binario), sistema que representa cada cifra del número decimal en sistema binario mediante un conjunto de 4 bits.

El método consiste en asignar, para cada cifra del número decimal un octeto o byte. El cuarteto de la derecha es la representación BCD de la cifra y el cuarteto de la izquierda (bits de ZONA) está formado por 1111.

El signo se escribe en los bits de zona del octeto mas a la derecha, siendo 1100 el equivalente al signo + y 1101 el equivalente al - .

Se denomina compactación a la representación simplificada, de un código binario mediante los sistemas octal o hexadecimal, esto es, representando cada 3 o 4 bits respectivamente, por su correspondiente carácter octal o hexadecimal.

- *Decimal Empaquetado*

La cantidad, en sistema decimal se representa mediante representación BCD utilizando un cuarteto para cada cifra (se eliminan los bits de zona), salvo para el último dígito que se representa por un octeto, cuyo primer cuarteto es el signo, en las mismas condiciones que en el método decimal desempaquetado.

Representación en coma flotante

Se utiliza para la representación de números reales. Tiene un rango de representación mayor que la representación en coma fija permitiendo a la computadora la utilización de números muy grandes o muy pequeños. Como contraprestación ofrece una disminución en la precisión de los números representados.

Se fundamenta este método en la "*Notación Científica Normalizada*" dividiendo la palabra, o doble palabra del ordenador en dos partes: una ***mantisa*** y un **exponente**, utilizando como ***base de exponenciación*** la determinada por el fabricante del ordenador (normalmente 2 o una potencia de 2). El primer bit de la palabra se reserva para el signo, los siguientes para el exponente y los últimos para la mantisa, dependiendo el número de bits asociados a cada concepto de las prescripciones del fabricante.

El rango de representación mediante este método está formado por el intervalo comprendido entre el "***Desbordamiento Negativo***" y el "***Subdesbordamiento Negativo***", para números negativos y el comprendido entre el "***Subdesbordamiento Positivo***" y el "***Desbordamiento Positivo***", para números positivos. Definiéndose:

mNN (mínimo Número Negativo) = - máxima mantisa * $base^{máximo\ exponente}$

MNN (Máximo Número Negativo) = - mínima mantisa * $base^{-máximo\ exponente}$

mNP (mínimo Número Positivo) = mínima mantisa * $base^{-máximo\ exponente}$

MNP (máximo Número positivo) = máxima mantisa * $base^{máximo\ exponente}$

Por tanto, será:

Subdesbordamiento Positivo: 0 < X < mNP

Subdesbordamiento Negativo: MNN < X < 0

Desbordamiento Positivo: X > MNP

Desbordamiento Negativo: X < mNN

1.2.2.3.- Codificación alfanumérica

Como se ha dicho anteriormente, el sistema binario es la única forma en la que un ordenador puede recibir, tratar o almacenar información. Esto implica que tanto letras como números y caracteres especiales han de ser transformados a binario, asignando una secuencia de 0 y 1 a cada carácter que se introduzca. Esta asignación de secuencias el ordenador lo realiza utilizando un Código (BCD, EBCDIC, ASCII, FIELDATA, etc.) de forma que cada carácter tenga asociada una representación diferenciada en uno de estos códigos.

Los caracteres a representar serán Alfabéticos (en mayúsculas y en minúsculas), Numéricos (del 0 al 9), Caracteres Especiales (. , ; : etc.) y Caracteres de Control (Salto de línea, ACK, etc.).

Los códigos más utilizados son:

- **FIELDATA**, que utiliza 6 bits para la representación de cada código, permitiendo por tanto 2^6 = 64 caracteres distintos.
- **ASCII**, que utiliza 7 bits para cada carácter (2^7 = 128 caracteres distintos).
- **ASCII EXTENDIDO** que utiliza 8 bits para cada carácter (2^8 = 256 caracteres)
- **EBCDIC** que utiliza 8 bits para cada carácter (2^8 = 256 caracteres)

1.3.- Ciclo de vida de una aplicación informática

Se define por **ciclo de vida del software** al conjunto de fases por la que pasa a lo largo del tiempo, desde la fase de estudio y concepción hasta la de realización, explotación y mantenimiento.

Diversas organizaciones profesionales (IEEE o ISO/IEC) han publicado normas relativas al ciclo de vida del software. Concretamente:

La norma IEE 1074 entiende por **ciclo de vida del software** "una aproximación lógica ala adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software".

La norma ISO 12207-1 entiende por **modelo de ciclo de vida** "un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso".

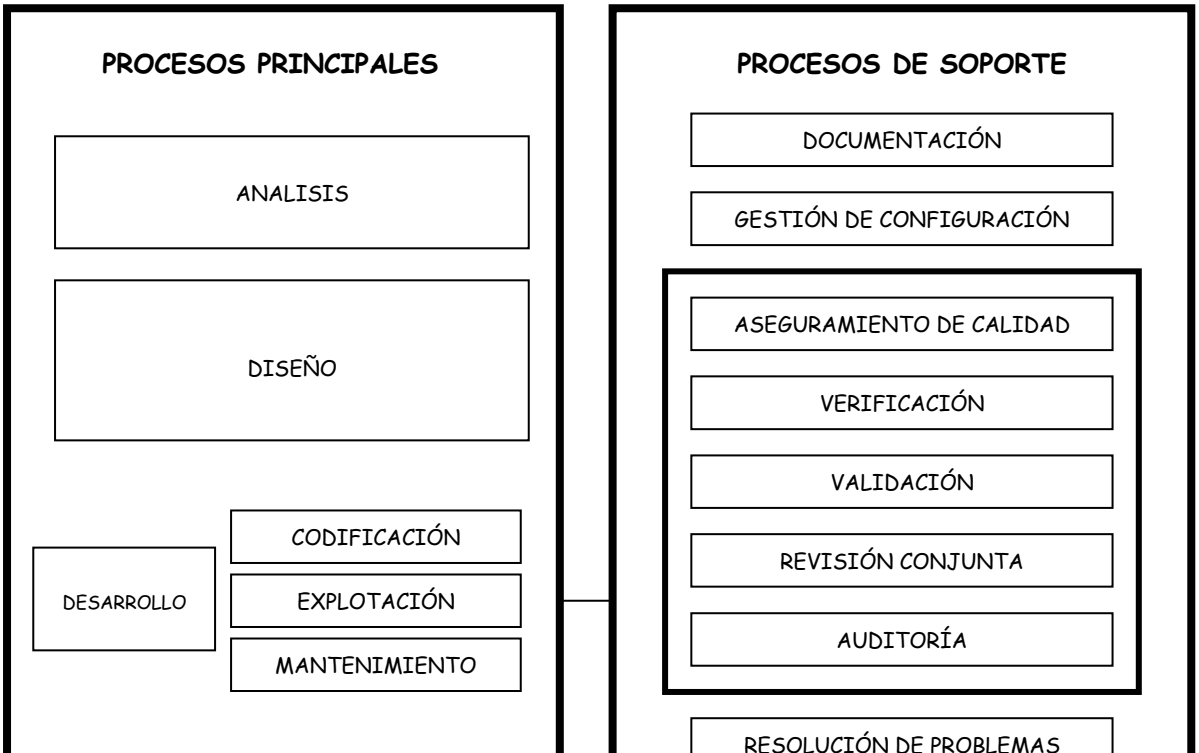
Ambas normas consideran una actividad como un conjunto de tareas u una tarea como una acción que transforma entradas en salidas.

El ciclo de vida abarca por tanto toda la vida del sistema, desde su concepción hasta que deja de utilizarse, por eso, a veces se habla de **ciclo de desarrollo** como el subconjunto del anterior que empieza en el análisis y finaliza con la entrega del sistema al usuario.

1.3.1.- Procesos del ciclo de vida de una aplicación informática

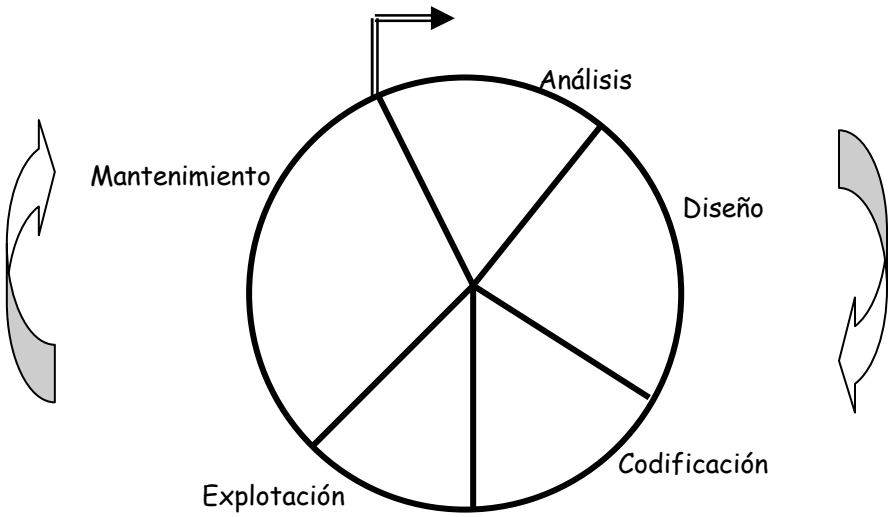
La norma ISO 12207 - 1 establece las actividades a realizar durante el ciclo de vida software, agrupadas según en procesos según la figura, indicando que la norma no fomenta ni especifica ningún modelo concreto de ciclo de vida, gestión de software o método de ingeniería ni establece cómo realizar ninguna de las actividades.

Dichas actividades se agrupan en cinco procesos principales, ocho procesos de soporte y cuatro procesos de la organización.



Los procesos principales se corresponden con el ***ciclo de desarrollo del programa informático*** y se establece como una serie de etapas o fases que deben seguirse secuencialmente y de forma ordenada cuando se desea desarrollar un determinado producto de software.

Estas fases son:



Análisis.- Es la fase en la que se establece cual es el producto a desarrollar, en la que se especifican los procesos y las estructuras de datos que se van a utilizar para satisfacer las necesidades del usuario y se desarrollan prototipos que sirvan para definir con precisión los requerimientos.

En el análisis estructurado, se pueden emplear diversas técnicas, de las cuales las mas importantes son:

- **Diagramas de Flujo de Datos:** Sirven para conocer el comportamiento del sistema mediante representaciones gráficas del flujo de información con diferentes niveles de abstracción y división del problema.

- **Modelos de Datos:** Sirven para conocer las estructuras de datos y sus características. Pueden emplearse *modelos entidad – relación* y formas normales.

- **Diccionario de Datos:** Sirven para describir todos los objetos utilizados en los gráficos así como en las estructuras de datos.

- **Definición de interfaces de usuario:** Sirven para determinar la información que hay que manejar en las entradas y las salidas de datos

Al final de esta fase se debe obtener un documento que contenga laa especificaciones de la aplicación

Diseño.- Es la fase en la que se alcanza la solución óptima, detallada y con mayor precisión posible para el desarrollo de la aplicación, teniendo en cuenta los recursos del sistema, tanto físicos como lógicos.

En el diseño estructurado se pueden diferenciar las siguientes etapas:

- **Diseño externo:** Especifica los formatos de la información en los soportes de entrada y salida. Se diseñan especialmente formatos de pantalla y listados.

- **Diseño de datos**: Establece las estructuras de datos de acuerdo con su soporte físico y lógico específico (estructuras en memoria, ficheros y bases de datos).

- **Diseño modular:** Es una técnica de representación en la que se refleja de forma descendente la división de la aplicación en módulos, así como la interconexión de los mismos con variables de enlace o parámetros. Está basado en los diagramas de flujo de datos obtenidos en el análisis y se deberá tener en cuenta la cohesión de cada módulo y el acoplamiento e interdependencia entre ellos.

- **Diseño procedimental:** Establece las especificaciones para cada módulo describiendo el algoritmo necesario que permita posteriormente una correcta y rápida codificación. Se emplean las técnicas de programación estructurada Para la representación de los algoritmos diseñados suelen emplearse organigramas, ordinogramas, pseudo código, tablas de decisión, etc.

La documentación obtenida conforma el ***cuaderno de carga*** de la aplicación.

Codificación.- Consiste en la traducción de la solución obtenida a un determinado lenguaje de programación. La selección del lenguaje se realiza en función de las especificaciones de diseño expresadas en el cuaderno de carga. Así mismo deben realizarse pruebas para depurar errores (pruebas de caja blanca y caja negra) y verificar la calidad de los programas. En particular, son pruebas a utilizar:

- **Pruebas unitarias:** Para comprobar que cada módulo realiza correctamente su tarea.

- **Pruebas de interconexión:** Para comprobar que en cada programa es correcto el funcionamiento conjunto de todos sus módulos

- **Pruebas de integración:** Para comprobar el funcionamiento correcto del conjunto de programas que constituyen la aplicación, es decir, el funcionamiento de todo el sistema.

Explotación.- En esta fase se realiza la implantación de los programas (aplicación) en el entorno operativo o sistema físico donde va a explotarse habitualmente, y su puesta en marcha para obtener un funcionamiento normal de todo el sistema.

Se suelen realizar las siguientes actividades:

- Instalación de programas
- Pruebas globales de aceptación del sistema
- Conversión de la información del sistema anterior al nuevo sistema
- Entrada en funcionamiento del nuevo sistema
- Eliminación del sistema anterior.

Mantenimiento.- Es la fase que completa el ciclo de desarrollo y en ella se realizan las correcciones necesarias para subsanar errores y deficiencias del producto desarrollado, existiendo la posibilidad de que ciertas acciones de esta fase reinicien el ciclo de desarrollo.

Normalmente se distinguen los siguientes tipos de mantenimiento:

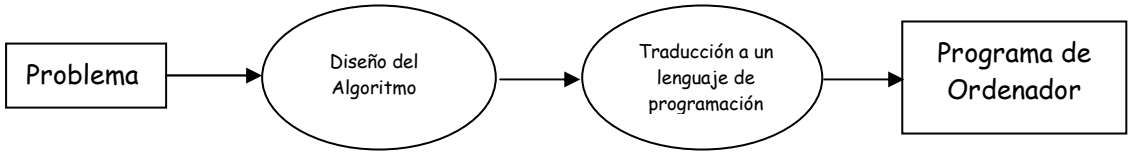
- **Mantenimiento correctivo:** Consiste en corregir errores no detectados en pruebas anteriores y que aparezcan con el uso normal de los programas.
- **Mantenimiento adaptativo:** Consiste en modificar los programas existentesa causa de un cambio de entorno físico y lógico en el que están implicados, por ejemplo, nuevas generaciones de ordenadores, nuevas versiones del sistema operativo, cambio de equipos periféricos, etc.
- **Mantenimiento perfectivo:** Consiste en una mejora de la aplicacióni al recibir, por parte del usuario, propuestas sobre nuevas posibilidades y modificaciones sobre funciones existentes.

1.4.- Algoritmos

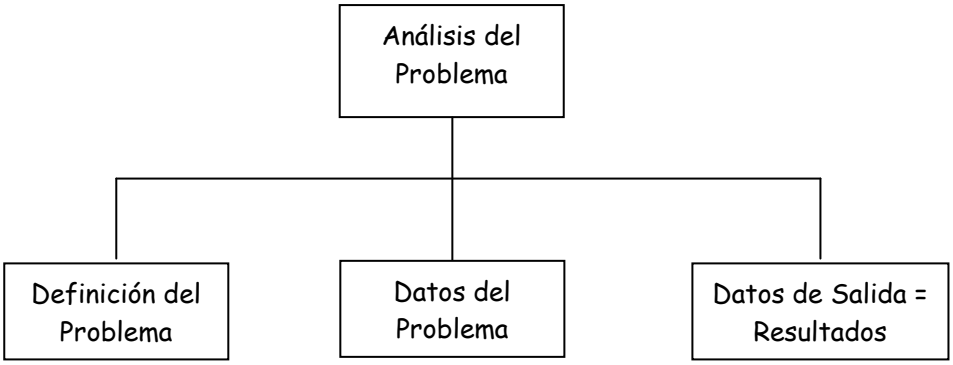
Se define **algoritmo** como la descripción abstracta de todas las acciones u operaciones que debe realizar un ordenador de forma clara y detallada, así como el orden en el que éstas deberán ejecutarse, junto con la descripción de todos aquellos datos que deberán ser manipulados por dichas acciones y que conducen a la solución del problema planteado, facilitando así su posterior traducción al lenguaje de programación correspondiente.

El diseño de todo algoritmo debe reflejar las tres partes de un programa: ***entrada***, ***proceso*** y ***salida***

Es importante tener en cuenta que todo algoritmo debe ser totalmente independiente del lenguaje de programación que será utilizado, esto es, el algoritmo diseñado deberá permitir su traducción a cualquier lenguaje de programación, con independencia del ordenador en que se vaya a ejecutar dicho programa habitualmente



El primer paso para encontrar solución al problema planteado es el **análisis** del mismo. Se debe examinar cuidadosamente el problema a fin de obtener una idea clara sobre lo que se solicita y determinar los datos necesarios para conseguirlo



El algoritmo puede ser definido por una secuencia ordenada de pasos, sin ambigüedades, que conduzcan a la solución de un problema dado y expresado en lenguaje natural, por ejemplo castellano, de forma que todo algoritmo:

Debe ser ***conciso y detallado***, esto es, debe reflejar con el máximo detalle el orden de ejecución de cada acción u operación que vaya a realizar el ordenador.

Debe ser ***flexible,*** y nunca rígido en su diseño debiendo mantener esta cualidad o característica en sus representaciones gráficas, facilitando de esta manera futuras modificaciones o actualizaciones del diseño realizado.

Debe ser ***finito y limitado*** esto es, el algoritmo se caracteriza por tener un comienzo y un final.

Debe ser ***exacto y preciso***, es decir, al aplicar el algoritmo n veces con los mismos datos de entrada, se deben obtener siempre los mismos resultados o datos de salida.

Debe ser ***amigable y entendible***, facilitando su entendimiento así como su traducción a los diferentes lenguajes. Esto es, debe ser claro y sencillo.

En lo que se refiere al **diseño** del algoritmo, éste debe comenzar por identificar las tareas mas importantes para resolver el problema y disponerlas en el orden en que han de ser ejecutadas.

Los pasos en esta primera descripción de actividades deberán ser refinados posteriormente, añadiendo mas detalles a los mismos e, incluso, algunos de ellos, podrán requerir un refinamiento adicional, hasta que pueda obtenerse un algoritmo claro, conciso y completo. Este método de diseño de los algoritmos en etapas, yendo de los conceptos generales a los de detalle mediante refinamientos sucesivos se conoce como método ***descendente, holístico,*** o ***top-down***.

En todo algoritmo deben considerarse tres partes

- **Entrada.-** Información dada al algoritmo

- **Proceso.-** Operaciones o cálculos necesarios para encontrar la solución del problema

- **Salida.-** Respuestas dadas por el algoritmo o resultados finales de los cálculos

Por último, una vez se ha terminado de escribir el algoritmo es necesario comprobar que realiza las tareas para las que se ha diseñado y produce el resultado correcto y esperado. El método mas normal para la ***verificación del algoritmo*** es mediante su ejecución manual, usando datos significativos que abarquen todo el posible rango de valores y anotando en una hoja de papel las modificaciones que se producen en las diferentes fases hasta la obtención de los resultados. Este proceso se conoce como ***Prueba del algoritmo***

Una vez que el algoritmo está diseñado, representado gráficamente mediante alguna herramienta de programación (Diagrama de flujo, organigrama o pseudo código) y verificado, se debe pasar a la fase de ***codificación***, traducción del algoritmo a un determinado lenguaje de programación, que deberá ser completada con la ***ejecución*** y ***comprobación*** del programa en el ordenador. Todo ello representa la ***fase de implementación.***

1.5.- Lenguajes de Programación

Un **lenguaje de programación** es un conjunto de símbolos junto aun conjunto reglas para combinar dichos símbolos que se usan para expresar programas. Los lenguajes de programación, como cualquier tipo de lenguaje, se componen de un **léxico** (conjunto de símbolos permitidos o vocabulario), una **sintaxis** (reglas que indican cómo realizar las construcciones del lenguaje) y una **semántica** (reglas que permiten determinar el significado de cualquier construcción del lenguaje).

Para que una computadora pueda procesar un programa escrito en un determinado lenguaje de programación, es necesario realizar previamente una traducción del programa al lenguaje que entienda dicha computadora según una serie de fases.

1.5.1.- Lenguaje máquina

El lenguaje máquina es el único lenguaje que entiende directamente un ordenador. Por esta razón, su estructura está totalmente adaptada a los circuitos de la máquina y muy alejada de la forma de expresión y análisis de los problemas propia de los humanos.

Esto hace que la programación en este lenguaje resulte tediosa y complicada, requiriéndose un conocimiento profundo de la arquitectura física del ordenador. Frente a esto, el código máquina hace posible que el programador utilice la totalidad de los recursos que ofrece el ordenador, obteniéndose programas muy eficientes (es decir, que aprovechan al máximo los recursos existentes) en tiempo de ejecución y en ocupación de memoria,

Las principales características del lenguaje máquina son las siguientes:

- Las instrucciones se expresan en el *alfabeto binario* (están codificadas en binario como cadenas de ceros y unos), pudiéndose utilizar códigos intermedios (octal y hexadecimal). Esta característica hace que un programa en lenguaje máquina sea difícil de entender y, como consecuencia, difícil de modificar

- Los datos se referencian por medio de las direcciones de memoria donde se encuentran (no aparecen nombres de variables o de constantes).

- Las instrucciones realizan operaciones muy simples, El programador debe ingeniárselas para expresar cada una de las operaciones que desea realizar en términos de las instrucciones elementales de las que dispone.

- Existe muy poca versatilidad para la redacción de las instrucciones, ya que tienen un formato rígido en cuanto a la posición de los distintos campos (código de operación seguido de los campos dedicados a los operandos).

- El lenguaje máquina depende y está íntimamente ligado a la CPU del ordenador. Esto hace que los programas en lenguaje máquina no sean transferibles de un modelo de ordenador a otro (un programa en lenguaje máquina sólo se puede ejecutar en el procesador para el cual está destinado), es decir, existe *baja portabilidad*

Estas limitaciones son resueltas, en parte por el lenguaje ensamblador y prácticamente en su totalidad por los lenguajes simbólicos de alto nivel.

1.5.2.- Lenguaje Ensamblador

El lenguaje ensamblador constituye el primer intento de sustitución del lenguaje máquina por uno más cercano al usado por los humanos. Este acercamiento a las personas se plasma en las siguientes aportaciones:

- Uso de una *notación simbólica* o *nemotécnica* para representar los códigos de operación, de esta forma, se evitan los códigos numéricos, tan difíciles de manejar. Normalmente, los códigos nemotécnicos están constituidos por abreviaturas de las operaciones en inglés (dado el origen anglosajón de los fabricantes), así, por ejemplo, la suma se representa en la mayoría de los ensambladores por ADD.

- *Direccionamiento simbólico*. En lugar de utilizar direcciones binarias absolutas, los datos pueden ser identificados con nombres como COSTE, TOTAL, X,Y,Z, etc.

- Se permite el *uso de comentarios* entre las líneas de instrucciones, haciendo posible la redacción de programas más legibles.

Aparte de esto, el lenguaje ensamblador presenta la mayoría de los inconvenientes del lenguaje máquina, como son su repertorio muy reducido de instrucciones, el rígido formato de las instrucciones, la baja portabilidad y la fuerte dependencia del hardware. Por otro lado, mantiene la ventaja del uso óptimo de los recursos hardware, permitiendo la obtención de un código muy eficiente.

Este tipo de lenguajes hacen corresponder a cada instrucción en ensamblador una instrucción en código máquina. Esta traducción es llevada a cabo por un programa traductor denominado **ensamblador**.

Para solventar en cierta medida la limitación que supone poseer un repertorio de instrucciones tan reducido, se han desarrollado unos ensambladores especiales denominados macroensambladores. Los lenguajes que traducen los macroensambladores disponen de macroinstrucciones cuya traducción da lugar a varias instrucciones máquina y no a una sola. Por ejemplo, existen macroinstrucciones para multiplicar, dividir, transferir bloques de memoria principal a disco, etc.

Dado que el lenguaje ensamblador está fuertemente condicionado por la arquitectura del ordenador que soporta, los programadores no suelen escribir programas de tamaño considerable en ensamblador. Más bien usan este lenguaje para afinar partes importantes de programas escritos en lenguajes de más alto nivel. El lenguaje ensamblador sigue siendo importante, ya que da al programador el control total de la máquina, y como resultado genera un código compacto, rápido y eficiente.

1.5.3.- Lenguajes de alto nivel

Los esfuerzos encaminados a hacer la labor de programación independiente de la máquina dieron como resultado la aparición de lenguajes de programación de alto nivel. Estos lenguajes, mas evolucionados, usan palabras y frases relativamente fáciles de entender y proporcionan también facilidades para expresar alteraciones del flujo de control de una forma bastante sencilla e intuitiva.

Las características fundamentales de los lenguajes de alto nivel son las siguientes:

- Son ***independientes de la arquitectura física de la computadora***. Esto permite utilizar los mismos programas en computadoras de arquitecturas diferentes (portabilidad) y, además, no es necesario conocer el hardware específico de la máquina.

- Al igual que ocurre con el lenguaje ensamblador, la ejecución de un programa en lenguaje de alto nivel requiere de una traducción del mismo al lenguaje máquina de la computadora donde va a ser ejecutado.

- Por lo general***, una sentencia en un lenguaje de alto nivel da lugar, al ser traducida, a varias instrucciones en lenguaje máquina***. Además, determinados componentes del programa pueden ser referenciados con un nombre simbólico para facilitar la comprensión de las personas.

- ***Utilizan notaciones cercanas a las usadas por las personas en un determinado ámbito.*** Se pretende la mayor aproximación posible al lenguaje natural o al lenguaje algebraico. Para ello, las instrucciones vienen expresadas mediante texto, es posible introducir comentarios en las líneas de las instrucciones y la escritura de programas está usualmente basada en reglas parecidas a las humanas. Todas estas características dan lugar a programas más legibles y de más fácil modificación y puesta a punto.

- Se suelen incluir ***instrucciones potentes de uso frecuente*** que son ofrecidas por el lenguaje de programación. Por ejemplo, se suelen ofrecer funciones matemáticas de uso común (seno, coseno, conversión de entero a real, etc.), operadores específicos de entrada/salida, operadores de tratamiento de cadenas de caracteres, etc.

- Existe gran cantidad de lenguajes de alto nivel actualmente en uso, y se han desarrollado diferentes versiones de algunos de ellos. Esta heterogeneidad constituye el principal problema que presentan estos lenguajes.

- Los lenguajes de alto nivel, a diferencia de los lenguajes máquina y ensamblador, no permiten aprovechar completamente los recursos internos de la máquina.

Todas estas características ponen de manifiesto un acercamiento a las personas y un alejamiento de la máquina. Por esta razón, los programas escritos en lenguajes de alto nivel no pueden ser directamente interpretados por el ordenador, siendo necesario realizar previamente su traducción a lenguaje máquina.

Para ello se utilizan programas traductores (previamente desarrollados para cada computadora) que se encarguen de realizar dicho proceso de traducción. Hay dos tipos de traductores de lenguajes de alto nivel: los ***compiladores y los intérpretes***.

Como la computadora puede interpretar y ejecutar únicamente el código máquina, existen programas especiales, denominados traductores, que traducen programas escritos en un lenguaje de programación al lenguaje máquina de la computadora. Un **traductor** es un metaprograma que toma como entrada un programa (o parte de un programa) escrito en lenguaje simbólico alejado de la máquina denominado ***programa fuente*** y proporcionan como salida otro programa semánticamente equivalente, escrito en un lenguaje comprensible por el hardware de la computadora denominado ***programa objeto***.

Un **compilador** traduce completamente un programa fuente, generando un programa objeto (semánticamente equivalente) escrito en lenguaje máquina. Como parte importante de este proceso de traducción, el compilador informa al usuario de la presencia de errores en el programa fuente, pasándose a la creación del programa objeto sólo en el caso de que no hayan sido detectados errores en el programa fuente (por lo general, suele cancelarse la compilación cuando se detectan errores). El programa fuente suele estar contenido en un fichero, y el programa objeto puede almacenarse como otro fichero en memoria masiva para ser ejecutado posteriormente, sin necesidad de volver a realizar la traducción. Una vez traducido un programa, su ejecución es independiente de su compilación.

Un **intérprete** permite que un programa fuente escrito en un determinado lenguaje vaya traduciéndose y ejecutándose directamente sentencia a sentencia por la computadora. El intérprete capta una sentencia fuente, la analiza y la interpreta, dando lugar a su ejecución inmediata. Por consiguiente, en este caso no se crea ningún fichero o programa objeto almacenable en memoria masiva para posibles ejecuciones futuras.

Si utilizamos un intérprete para traducir un programa cada vez que necesitamos ejecutar el programa tenemos que volver a analizarlo, ya que no se genera un fichero objeto. En cambio, con un compilador, aunque sea más lenta, la traducción sólo debe realizarse una vez. Además, los traductores no permiten realizar optimizaciones del código (que eliminan órdenes innecesarias compactando el código) más allá del contexto de cada sentencia del programa.

La principal ventaja de los intérpretes frente a los compiladores es que resulta mas fácil localizar y corregir errores de los programas (depuración de programas), ya que la ejecución de un programa bajo un intérprete puede interrumpirse en cualquier momento

para conocer los valores de las distintas variables y las instrucción fuente que acaba de ejecutarse.

1.6.- Tipos de Lenguajes

Los lenguajes de programación se pueden clasificar de acuerdo a diversos criterios. Como criterios mas extendidos se consideran los que se expresan a continuación.

El criterio más simple que se puede considerar hace referencia a la **proximidad** del lenguaje con la máquina o con el lenguaje natural. De acuerdo a este criterio, existen tres niveles de lenguajes de programación:

- ***Lenguajes de bajo nivel***: Lenguajes máquina.
- ***Lenguajes de nivel medio***: Lenguajes ensambladores y macroensambladores.
- ***Lenguajes de alto nivel***: A los que ya hemos hecho referencia.

Dado que los lenguajes de programación, en cierto modo, han sufrido un desarrollo paralelo a la evolución de las computadoras, pueden clasificarse los lenguajes de programación atendiendo a su **desarrollo histórico** desde el comienzo de las computadoras. Esta clasificación distingue cinco generaciones de lenguajes:

- ***Primera generación:*** Lenguajes máquina.
- ***Segunda generación:*** Ayudas a la programación como son los ensambladores.
- ***Tercera generación:*** Lenguajes de alto nivel imperativos que siguen vigentes en la actualidad, como PASCAL, MODULA-2, FORTRAN, COBOL, C y ADA.
- ***Cuarta generación (4G):*** Lenguajes o entornos de programación orientados básicamente a aplicaciones de gestión y bases de datos, como SQL, NATURAL, etc.
- ***Quinta generación***: Lenguajes orientados a aplicaciones en inteligencia artificial, como LISP y PROLOG.

1.6.1.- Clasificaciones de los lenguajes de alto nivel

Dado que los lenguajes de alto nivel son los más ampliamente usados, suelen considerarse clasificaciones centradas concretamente en estos lenguajes. De forma muy general, los lenguajes de alto nivel se pueden dividir en ***lenguajes de propósito general***, que pueden ser empleados en cualquier tipo de aplicaciones (C y ADA) y ***lenguajes de propósito especial***. No obstante, esta clasificación es demasiado general, por lo que se considera la clasificación que hace Tucker desde el punto de vista del campo de aplicación al que pertenece el lenguaje:

- **Aplicaciones Científicas:** En este tipo de aplicaciones predominan las operaciones numéricas o matriciales propias de algoritmos matemáticos. Lenguajes adecuados para este tipo de aplicaciones son FORTRAN y PASCAL.

- **Aplicaciones de procesamiento de datos:** En estas aplicaciones son frecuentes las operaciones de creación, mantenimiento y consulta sobre ficheros y bases de datos. Dentro de este campo, estarían aplicaciones de gestión empresarial, como programas de nóminas, contabilidad, facturación, control de inventario, etc. Lenguajes aptos para este tipo de aplicaciones son COBOL y SQL.

- **Aplicaciones de tratamiento de textos:** Estas aplicaciones están asociadas al manejo de textos en lenguaje natural. Un lenguaje muy adecuado para este tipo de aplicaciones es el C.

- **Aplicaciones en inteligencia artificial:** Dentro de este campo, destacan las aplicaciones en sistemas expertos, juegos, visión artificial y robótica, Los lenguajes más populares dentro del campo de la inteligencia artificial son LISP y PROLOG.

- **Aplicaciones de programación de sistemas:** En este campo se incluyen la programación de software de interfaz entre el usuario y el hardware, como son los módulos de un sistema operativo y los traductores. Tradicionalmente, para estas aplicaciones se utilizaba el lenguaje ensamblador, no obstante, en la actualidad se muestran muy adecuados los lenguajes ADA. MODULA-2 y C.

Otra forma de clasificar los lenguajes de alto nivel tiene en cuenta el estilo de programación que fomentan, es decir, la filosofía de construcción de programas:

- **Lenguajes imperativos o procedurales**: Estos lenguajes establecen cómo debe ejecutarse una tarea, dividiéndola en partes que especifican cómo realizar cada una de las subtareas asociadas. Estos lenguajes se fundamentan en el uso de variables para almacenar valores y el uso de instrucciones que indican las operaciones a realizar sobre los datos almacenados. La mayoría de los lenguajes de alto nivel son de este tipo: FORTRAN, BASIC, PASCAL, ADA, MODULA2, C, etc.

- **Lenguajes declarativos:** En este caso, el proceso por el cual se ejecuta el programa no aparece de forma explícita en el programa, el programador no tiene que indicar el proceso detallado de cómo realizar la tarea. De hecho en estos lenguajes los programas se construyen mediante descripciones de funciones (lenguajes funcionales, como LISP) o expresiones lógicas que indican las relaciones entre determinadas estructuras de datos (lenguajes de programación lógica. como PROLOG).

- **Lenguajes orientados a objetos:** El diseño de los programas se centra más en los datos y su estructura. Los programas consisten en descripciones de unidades denominadas objetos que encapsulan los datos (almacenados en variables) y las operaciones que actúan sobre ellos (que indican el comportamiento del objeto). El lenguaje más usado dentro de este tipo es el C++ y el Java.

- **Lenguajes orientados al problema:** Este tipo de lenguajes están diseñados para problemas específicos, principalmente de gestión. En estos lenguajes, los programas están formados por sentencias que ordenan qué se quiere hacer. Generalmente, estos lenguajes suelen ser generadores de aplicaciones que permiten automatizar en la medida de lo posible la tarea de desarrollo de

software de aplicaciones de gestión.

1.7.- Objetos de un programa

Los objetos de un programa son todos aquellos entes sobre los que actúa un programa, pudiendo clasificarse en Datos, Identificadores, Operadores y Expresiones.

1.7.1.- Datos

Se denomina **dato** a toda aquella información característica de una entidad y que es susceptible de tratamiento en un programa informático. Aunque internamente, para el ordenador, toda la información es numérica (series de unos y ceros), los datos deben estar tipificados o clasificados, de tal forma que los programas sean capaces de realizar los tratamientos necesarios de una forma determinada.

La forma de clasificar los datos constituye la **estructura de datos** dentro de un lenguaje. En el diseño de un programa es importante establecer correctamente las estructuras de datos, ya que los tratamientos que se han de realizar sobre éstos están relacionados con sus estructuras. Los datos se caracterizan principalmente por un ***identificador***, *un* ***tipo*** y un ***valor***.

Identificador es el nombre que usa el programador para referenciar los datos y otros elementos del programa, permitiendo así su definición en una posición de la memoria del ordenador.

Como normas generales para el empleo de identificadores suelen enumerarse las siguientes:

- Pueden estar constituidos por letras y dígitos y, en algunos casos, por el carácter de subrayado (_).
- Deben empezar por una letra
- No deben contener espacios
- El número máximo de caracteres y nombres reservados que se pueden emplear dependen del compilador utilizado
- El nombre asignado debe tener relación con la información que contiene, pudiéndose emplear abreviaturas que sean significativas.

Por **Tipo** de dato se entiende el *rango de valores que puede tomar el dato, en función de una clasificación y que determina el espacio de memoria que hay que reservar.*

Por último, el **Valor** de un dato representa *el elemento determinado perteneciente al rango de valores según su tipo y que estará contenido en el espacio de memoria reservado.*

1.7.2.- Clasificación de los datos

Pueden establecerse distintas clasificaciones de los datos:

a.- La clasificación mas elemental, desde el punto de vista del **tamaño de la información en la memoria del ordenador**, es la siguiente:

Bit	Unidad mínima para la representación de la información, que sólo puede tomar los valores de 0 y 1
Byte u Octeto	Conjunto de ocho bits, que sirve de referencia en la compartición de la memoria del ordenador y se utiliza para representar un carácter
Palabra	Conjunto de bits que pueden ser manipulados por el ordenador en una sola operación. Pudiendo ser de 8, 16, 32 o 64 bits, dependiendo del procesador utilizado

b.- La clasificación mas interesante es la que contempla las distintas formas de **representación de la información para su almacenamiento y tratamiento**, y es la siguiente:

<table>
<tr><td rowspan="2">Datos Básicos</td><td>Numéricos</td><td colspan="3">Entero
Real</td></tr>
<tr><td colspan="4">Carácter
Lógico</td></tr>
<tr><td>Datos Derivados</td><td colspan="4">Punteros</td></tr>
<tr><td rowspan="4">Datos Estructurados</td><td rowspan="3">Internos</td><td colspan="2">Estáticos Lineales</td><td>Tabla</td></tr>
<tr><td rowspan="2">Dinámicos</td><td>Lineales</td><td>Lista
Pila
Cola</td></tr>
<tr><td>No Lineales</td><td>Árbol
Grafo</td></tr>
<tr><td>Externos</td><td colspan="3">Fichero
Base de datos</td></tr>
<tr><td>Compuesto</td><td colspan="3">Estructura de datos o registro</td></tr>
</table>

c.- Otra clasificación de los datos, en función **de la permanencia de su contenido** o valor en memoria durante la ejecución del programa, es:

Constantes

Variables

A continuación se desglosan las definiciones de los datos antecitados.

1.7.2.1.- Datos básicos

Numéricos

Sirven para contener magnitudes

a.- Numérico entero

Se utiliza para representar números enteros, pudiendo llevar o no el signo correspondiente, se expresa mediante una serie de dígitos (de 0 a 9), pudiendo estar precedido por el signo + o el signo -. Su rango está determinado por el formato de representación en memoria.

b.- Numérico real

Se utiliza para representar los números con parte decimal o los números muy grandes o muy pequeños que no pueden ser contenidos en un número entero. Puede ser expresado de dos formas:

- **Punto decimal:** Utiliza los dígitos del 0 al 9 con su signo correspondiente y un punto para separar la parte entera de la fraccionaria.
- **Notación científica o exponencial:** Utiliza el formato"mantisaEcaracteristica" tal que:

  Mantisa = Número real

  E = Representación de la base decimal

  Característica = Exponente correspondiente a un número entero con su signo.

Carácter

Se utiliza para representar un carácter dentro de un conjunto definido por el fabricante del ordenador, de tal forma que cada carácter se corresponde con un número entero sin signo según un determinado código. Las constantes se expresan encerrando el carácter entre comillas simples.

La representación interna depende del código empleado; si éste es el código ASCII extendido se utilizan 8 bits para la representación de cada carácter:

Carácter	Código ASCII	Representación Interna

A	65	0 1 0 0 0 0 0 1
9	57	0 1 1 1 1 0 0 1
-	45	0 0 1 0 1 1 0 1

Algunos lenguajes (COBOL o BASIC por ejemplo) permiten como tipo de dato básico el **tipo alfanumérico** (string), que es una cadena de caracteres formada por un número determinado de caracteres. Otros lenguajes, como C, no considera la existencia de este tipo de datos definiendo las cadenas de caracteres como ***vectores de caracteres.*** Una cadena de caracteres se expresa delimitando los caracteres que la forman entre dobles comillas.

Lógico

Se utiliza para representar dos valores distintos. Una constante de tipo lógico se puede representar como VERDADERO o FALSO, TRUE o FALSE, 1 o 0. En el texto se representan las constantes como V o F. Internamente se considera el valor 1 como verdadero y el 0 como falso.

1.7.3.1.- Datos derivados

Punteros

Se utilizan para contener la dirección de memoria de otra variable y debe ser definida con el mismo tipo de la variable que va a referenciar o apuntar. Este tipo de variable, empleada en determinados lenguajes, es muy útil para realizar operaciones con estructuras dinámicas y para el paso de parámetros por dirección en una llamada a un módulo del programa.

Los datos de tipo puntero tienen ciertas características especiales que se deben tener en cuenta:

- Inicialmente se deben definir las variables de tipo puntero con el mismo tipo de dato que las variables a las que puede apuntar, aunque todavía no contenga ninguna dirección apuntada

- A continuación se debe asignar un contenido a la variable puntero, que será la dirección de memoria donde se encuentra ubicada la variable que se va a apuntar, la cual deberá estar previamente definida en memoria.

- Puede hacerse una referencia indirecta (***indirección***) al valor de una variable a través de un puntero.

- Se puede volver a utilizar el puntero para apuntar a otra variable, con la condición de que sea del mismo tipo que el definido por el puntero.

1.7.4.1.- Datos estructurados

Las características que los diferencian dentro de su clasificación son las siguientes:

Internos y Externos

Datos internos son los que residen an la memoria principal del ordenador. Por ejemplo, una tabla unidimensional de números enteros

Datos externos son los que residen en un soporte externo a la memoria principal, es decir, memoria auxiliar. Por ejemplo un fichero en un disco magnético

Estáticos y Dinámicos

Datos estáticos son aquellos cuyo tamaño queda definido en la compilación del programa y no se puede modificar durante la ejecución del mismo. Por ejemplo un vector

Datos dinámicos son aquellos cuyo tamaño puede ser modificado durante la ejecución del programa. Por ejemplo una lista encadenada.

Lineales y No Lineales

Datos lineales son los que pueden estar enlazados con un solo elemento anterior y un solo elemento posterior. Por ejemplo, una cola.

Datos no lineales son los que pueden enlazarse con mas de un elemento anterior y mas de un elemento posterior, por ejemplo un árbol.

Compuestos

Datos compuestos son los formados por el programador utilizando los tipos de datos básicos y derivados, pudiendo ser *internos* (estructuras de datos de un elemento) o *externos* (registro de un fichero con diversos campos).

1.7.3.- Constantes y variables

Constantes

Son aquellas cuya información es fija durante la ejecución del programa. Se pueden expresar de dos formas:

a.- De forma explícita mediante su valor

b.- Utilizando un identificador para definir la constante en memoria, asignándole un valor.

El compilador asignará la memoria con el tamaño correspondiente al tipo de dato expresado por su valor. Esta forma tiene la ventaja de permitir el cambio de su valor en la definición de la constante sin tener que modificar todos los lugares del programa donde se emplea.

Variables

Son datos cuya información puede ser variable durante la ejecución del programa. Estos datos deben ser definidos con un identificador y un tipo de dato. El identificador permite referenciar la variable para su uso en el programa, pudiéndose modificar su valor, mientras que el tipo de dato permite determinar el tamaño de la variable en la memoria.

Debe tenerse en cuenta que antes de utilizar una variable en el programa, ésta debe contener un valor que puede ser asignado inicialmente (en la definición) o bien durante la ejecución del programa.

1.7.2.- Operadores

En función de las operaciones que se vayan a realizar, los operadores se clasifican en:

OPERADOR	SÍMBOLO	SIGNIFICADO
Paréntesis	( )	Paréntesis
Aritméticos	**,^	Potencia
	*	Producto
	/	División
	div,\	División entera
	%,mod	Módulo (resto de la división entera)
	+	Signo positivo o suma
	-	Signo negativo o resta
Alfanuméricos	+	Concatenación
	-	Concatenación eliminando espacios
Relacionales	==,=	Igual a, asignación
	!=,<>	Distinto a
	<	Menor que
	<=	Menor o igual que
	>	Mayor que
	>=	Mayor o igual que
Lógicos	!, NOT, no	Negación

	&&, AND, y ||, OR, o	Conjunción Disyunción

1.7.3.- Tablas de verdad

En las operaciones lógicas, el resultado se determina por medio de tablas de verdad; suponiendo que A y B son expresiones lógicas y que V es Verdadero y F es Falso se especifican las siguientes tablas de verdad:

Para el operador AND

A	**B**	**A** AND **B**
V	**V**	**V**
V	**F**	**F**
F	**V**	**F**
F	**F**	**F**

La conjunción es Verdadera cuando los dos operandos son Verdaderos y Falso en el resto de los casos.

Para el operador OR

A	**B**	**A** OR **B**
V	**V**	**V**
V	**F**	**V**
F	**V**	**V**
F	**F**	**F**

La disyunción es Falsa cuando los dos operandos son Falsos y Verdadera en todos los demás casos.

Para el operador NOT

A	NOT **A**
V	**F**
F	**V**

La negación refleja lo contrario de lo expresado por el operando.

1.7.4.- Orden de prioridad de los operadores

Dentro de las expresiones hay que tener en cuenta un orden de prioridadde los operadores que depende del lenguaje que se vaya a utilizar, pero que, de forma general, se puede establecer de mayor a menor prioridad de la forma siguiente:

1º.- Paréntesis (comenzando por los mas externos)

2º.- Signo

3º.- Negación

4º.- Potencia

5º.- Producto, división y módulo

6º.- Suma y resta

7º.- Concatenación

8º.- Relacionales

9º.- Conjunción

10º.- Disyunción

Los operadores con la misma prioridad se evalúan de izquierda a derecha.

1.7.5.- Expresiones

Las expresiones son un conjunto de datos (operandos) y operadores con unas reglas específicas de construcción. Los operandos pueden ser también valores retornados por funciones. En la obtención del resultado se debe tener en cuenta el orden de prioridad de los operadores.

En función del resultado que se obtiene, las expresiones se pueden clasificar en:

- ***Numéricas:*** Su resultado es numérico y utilizan operandos y operadores numéricos. Deben ser escritas en formato algorítmico para que puedan ser interpretadas por el ordenador

- ***Alfanuméricas:*** Su resultado es una cadena de caracteres y utilizan operadores alfanuméricos

- ***Lógicas o booleanas:*** Su resultado es Verdadero (V) o Falso (F) y utilizan operadores relacionales y lógicos.