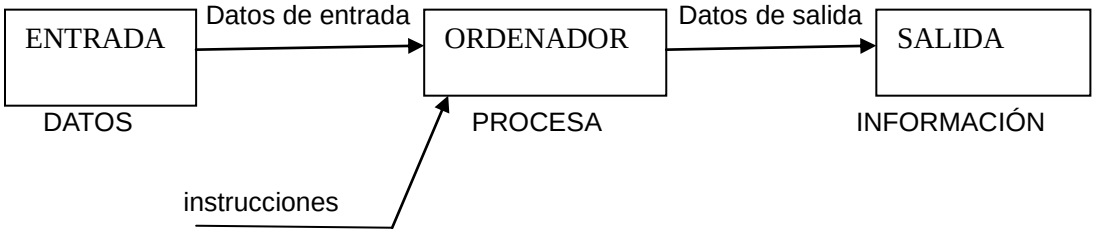
U.T. 1: INTRODUCCIÓN A LA PROGRAMACIÓN

1. CONCEPTOS.

INFORMÁTICA: (Fr: INFORmation autoMATIQUE). Tratamiento automático de la información, o ciencia que estudia el tratamiento racional de la información.

ORDENADOR o COMPUTADOR(A): (Fr: Ordenateur; Ing: Computer) Máquina de origen electrónico capaz de **aceptar unos datos de entrada**, efectuar con ellos unas **operaciones lógicas y aritméticas** y **proporcionar la información resultante**; todo ello sin intervención de un operador humano y bajo el control de un programa de instrucciones previamente almacenado en el propio ordenador.
Es capaz de realizar una gran variedad de trabajos, pero **sólo es capaz de hacer tres clases de operaciones básicas**: ***operaciones aritméticas sencillas*** (sumar, restar, …) ***operaciones lógicas sencillas*** (comparar dos valores, seleccionar o copiar símbolos numéricos o no, …) ***y almacenar o recuperar información.*** Con estas pocas operaciones, utilizadas y combinadas de forma adecuada mediante lo que llamamos **programa**, se pueden llegar a realizar **tareas increíblemente complejas** para resolver problemas de gestión, técnicos o de cualquier otro tipo.
Una computadora puede considerarse como un sistema cuya salida o **resultados** dependen de sus entradas, constituidas por **datos e instrucciones**. Los datos son conjuntos de símbolos utilizados para representar o expresar un valor numérico, un hecho, un objeto o una idea; en la forma adecuada para ser tratado.
Frecuentemente las salidas de un programa se denominan también datos pudiendo utilizarse estos como datos de entrada de un programa posterior. Es decir, **la palabra dato se utiliza como contraposición a instrucción**. La computadora actúa con dos tipos de informaciones **instrucciones**, que indican a la máquina qué es lo que tiene que hacer y **datos**, elementos sobre los que se actúa.



DATOS: son conjuntos de símbolos utilizados para representar o expresar un valor numérico, un hecho, un objeto o una idea.

INSTRUCCIONES: son operaciones bien definidas, específicas de cada máquina y almacenadas de forma permanente en la misma, que establecen las tareas que debe realizar.

PROGRAMA: Conjunto de órdenes o instrucciones que se le dan a un ordenador para realizar un proceso determinado.

APLICACIÓN: Conjunto de programas y documentación que permiten la realización de un determinado tipo de trabajo.

SISTEMA INFORMÁTICO: Conjunto de elementos necesarios para la realización y explotación de aplicaciones informáticas. Está compuesto por:

> **Hardware**: Conjunto de materiales físicos que componen el sistema informático, es decir, la propia computadora, los dispositivos externos a la misma, así como todo material físico relacionado con ellos (conexiones, cables, etc.).

Software: Parte lógica del sistema informático que dota al equipo físico de la capacidad para realizar cualquier tipo de tareas (conjunto de programas ejecutables sobre el hardware junto con los documentos y datos asociados a los mismos).

Personal Informático: Conjunto de personas que desempeñan las distintas funciones relacionadas con la utilización y explotación de las computadoras en una determinada empresa u organización.

2. CODIFICACIÓN DE LA INFORMACIÓN.

La **codificación** es una **transformación** que representa los elementos de un conjunto mediante uno o una combinación de los de otro, de forma tal que a cada elemento del primer conjunto le corresponda uno o una combinación única de elementos del segundo.

Un **código binario** es aquel que sólo utiliza dos símbolos distintos, que representaremos gráficamente como 0 y 1.

Un **BIT (Binary digIT)** es la unidad mínima de información (es decir, si sabemos algo, como mínimo lo que sabemos debe ser un bit. Por ejemplo, ¿esto es azul o no lo es? La unidad más elemental de información es un valor binario, conocido como **bit**. Un bit es una **posición o variable que toma el valor 0 ó 1**. En el interior de los ordenadores la información se almacena y se transfiere según un código binario (representado por 0 y 1), es decir, bits. Un bit **se representa como un determinado valor de voltaje en los circuitos electrónicos para el 1, o como otro valor, o la ausencia de voltaje, para el cero.**

Un **byte (CARÁCTER U OCTETO)** es el número de bits necesarios para almacenar un carácter. Este número depende del código utilizado por la computadora, siendo generalmente ocho. Como el byte es una unidad relativamente pequeña, es usual utilizar múltiplos:

1KB = 1 KiloByte = 2_{10} bytes = 1024
1MB = 1 MegaByte = 2_{20} bytes = 1024*1024
1GB = 1 GigaByte = 2_{30} bytes = 1024*1024*1024
1TB = 1 TeraByte = 2_{40} bytes = 1024*1024*1024*1024

2.1. Sistemas de numeración posicionales.

Un sistema de numeración es el conjunto de símbolos utilizados para representar cantidades, así como las reglas que rigen dicha representación. Una cantidad se representará por un conjunto ordenado de símbolos en cada uno de los sistemas existentes.
Un sistema de numeración posicional es aquel en que el valor que representa un símbolo en un número depende de su valor absoluto y de la posición que ocupa en el número respecto del punto que separa la parte entera de la parte fraccionaria, llamado “punto decimal” genéricamente. Los sistemas de numeración posicionales se rigen mediante el Teorema Fundamental de la Numeración, que dice que el valor decimal de una cantidad expresada en un sistema de numeración posicional cualquiera de base B, viene dado por su polinomio equivalente:

$X_n * B_n + X_{n-1} * B_{n-1} + \ldots + X_1 * B_1 + X_0 * B_0 + X_{-1} * B_{-1} + X_{-2} * B_{-2} + \ldots\ldots$
donde B es la base, utiliza B símbolos, $0 <= X_i < B$

Un sistema de numeración posicional en base *b*, utiliza *b* símbolos. Un número estará formado por un conjunto de dígitos/símbolos cuyo valor (el de cada dígito) depende de:
- La cifra en sí, el dígito.
- La posición que ocupa dentro del número.

Sistema decimal: Utiliza 10 símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) y su base es **10**.

3278,52 3000 + 200 + 70 + 8 + 0,5 + 0,02 = 3278,52
3278,52 $\mathbf{3}_3\mathbf{2}_2\mathbf{7}_1\mathbf{8}_0,\mathbf{5}_{-1}\mathbf{2}_{-2}$ $3*10_3 + 2*10_2 + 7*10_1 + 8*10_0 + 5*10_{-1} + 2*10_{-2} = 3278,52$

Sistema octal: Utiliza 8 símbolos (0, 1, 2, 3, 4, 5, 6, 7) y su base es **8**.

$175,372_8$ $1*8^2 + 7*8^1 + 5*8^0 + 3*8^{-1} + 7*8^{-2} + 2*8^{-3} =$
$= 64 + 56 + 5 + 3/8 + 7/64 + 2/512 = 125,4921875_{10}$

Sistema hexadecimal: Utiliza 16 símbolos (0, 1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,A ,B ,C ,D ,E ,F) y su base es **16**.

$3EA,2B_{16} = 3*16^2 + 14*16^1 + 10*16^0 + 2*16^{-1} + 11*16^{-2} =$
$= 3*256 + 14*16 + 10*1 + 2/16 + 11/256 = 768 + 224 + 10 + 0,125 + 0,04296875 =$
$= 1002,16796875_{10}$

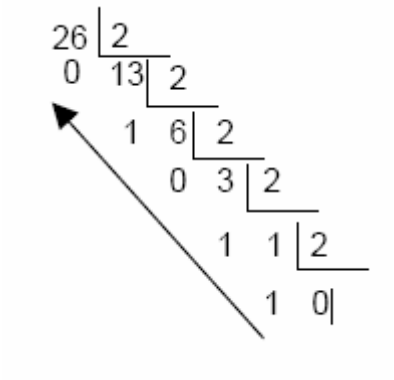
Sistema binario: Está basado en la utilización de dos símbolos, **0** y **1**, y su base es **2**.

$11010011,101_2 = 1*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2} +$
$1*2^{-3} = 128 + 64 + 0 + 16 + 0 + 0 + 2 + 1 + 0,5 + 0 + 0,125 = 211,625_{10}$

2.2. Cambios de sistemas de numeración.

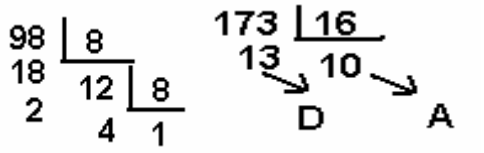
Para pasar de decimal a binario: Debemos distinguir entre la parte entera y la parte fraccionaria.

Parte entera: dividir sucesivamente (sin obtener decimales en el cociente) el número decimal y los cocientes que se van obteniendo por 2 hasta que el cociente en una de las divisiones sea 0. La unión de todos los restos obtenidos escritos en orden inverso nos proporciona el número inicial expresado en el sistema binario. El último resto obtenido será el bit más significativo (MSB: "Most Significative Bit", el bit de más a la izquierda) y el primer resto obtenido será el bit menos significativo (LSB: "Least Significative Bit", el bit de más a la derecha).



Por tanto, $26_{10} = 11010_2$

Para pasar de decimal a octal o a hexadecimal, la regla es la misma, pero cambiando el 2 por un 8 o un 16, respectivamente.



$98_{10} = 142_8$ $173_{10} = AD_{16}$

Parte fraccionaria: Se obtiene multiplicando por 2 sucesivamente la parte fraccionaria del número decimal de partida y las partes fraccionarias que se van obteniendo en los productos sucesivos. El número binario se forma con las partes enteras (que serán ceros o unos) de los productos obtenidos.

0,1875 x 2 = 0,375	01,875 x 8 = 1,5	0,1875 x 16 = 3,0
0,3750 x 2 = 0,75	0,5000 x 8 = 4,0	
0,7500 x 2 = 1,5		
0,5000 x 2 = 1,0		
$0,1875)_{10} = 0,0011)_2$	$0,1875)_{10} = 0,14)_8$	$0,1875)_{10} = 0,3)_{16}$

En octal y hexadecimal (cualquier otra base) la regla es la misma, cambiando la base.

Los cambios entre cualquier otro sistema posicional y decimal consisten en aplicar el teorema fundamental de la numeración tal como se ha visto antes.

2.3. Cambios de base directos entre sistemas de base potencia exacta de 2.

Los sistemas de numeración cuya base sea potencia exacta de 2, como 2, 4, 8, 16,etc, admiten la realización de cambios de base directos de cualquiera de ellos a binario (base 2) y viceversa, o entre ellos, mediante la sustitución de los símbolos por su representación en binario (base 2) y luego desde binario al sistema al que se quiere convertir. La siguiente tabla muestra la equivalencia en binario de los símbolos utilizados en los sistemas octal (8) y hexadecimal (16).

Binario	Octal	Binario	Hexadecimal
000	0	0000	0
001	1	0001	1
010	2	0010	2
011	3	0011	3
100	4	0100	4
101	5	0101	5
110	6	0110	6
111	7	0111	7
		1000	8
		1001	9
		1010	A
		1011	B
		1100	C
		1101	D
		1110	E
		1111	F

El sistema octal utiliza ocho símbolos distintos (0->7), por lo que son necesarias 8 combinaciones de números binarios distintas para representarlos. Puesto que n símbolos binarios permiten 2^n combinaciones distintas, necesitamos 3 dígitos binarios para tener 2^3=8 combinaciones que necesitamos. El razonamiento es análogo para hexadecimal, pero con cuatro cifras binarias, puesto que 2^4=16.

Pasar del sistema mayor (octal o hexadecimal) a binario consiste en sustituir cada cifra por su representación en binaria según la tabla, y eliminar los ceros a la izquierda no significativos:

$126)_8$ => 001 010 110 => $1010110)_2$
$1D3)_{16}$ => 0001 1101 0011 => $111010011)_2$

Pasar de binario a octal o hexadecimal consiste en dividir el número binario en grupos de tres (octal) o cuatro (hexadecimal) cifras, empezando por la derecha, rellenando por la izquierda con ceros si es necesario, y luego sustituyendo cada grupo por su símbolo equivalente según la tabla:

$1101110100010101)_2$ => 001, 101, 110, 100, 010, 101 => $156421)_8$
$1101110100010101)_2$ => 1101, 1101, 0001, 0101 >= $DD15)_{16}$

Pasar de octal a hexadecimal o viceversa: pasamos de octal a binario y luego de binario a hexadecimal o al contrario, utilizando los mecanismos ya vistos:

$DD15)_{16}$ => 1101 1101 0001 0101 => 001 101 110 100 010 101 => $156421)_8$

3. SISTEMA INFORMÁTICO. HARDWARE Y SOFTWARE

3.1. Estructura funcional de un ordenador.

Un ordenador se compone de las siguientes unidades funcionales:

Unidad de Entrada: Es el dispositivo por donde **se introducen en la computadora los datos e instrucciones**. En estas unidades se transforman las informaciones de entrada en señales binarias de naturaleza eléctrica. Una misma computadora puede tener distintas unidades de entrada. Son unidades de entrada el teclado, un digitalizador, una lectora de tarjetas de crédito, etc.
Unidad de Salida: Es un dispositivo por donde **se obtienen los resultados** de los programas ejecutados. La mayor parte de estas unidades transforman las señales eléctricas binarias en caracteres escritos o visualizados. Son dispositivos de salida una pantalla o una impresora.

Memoria: Es la unidad donde se **almacenan tanto los datos como las instrucciones** para después poder acceder a ellos. Existen dos tipos de memoria:

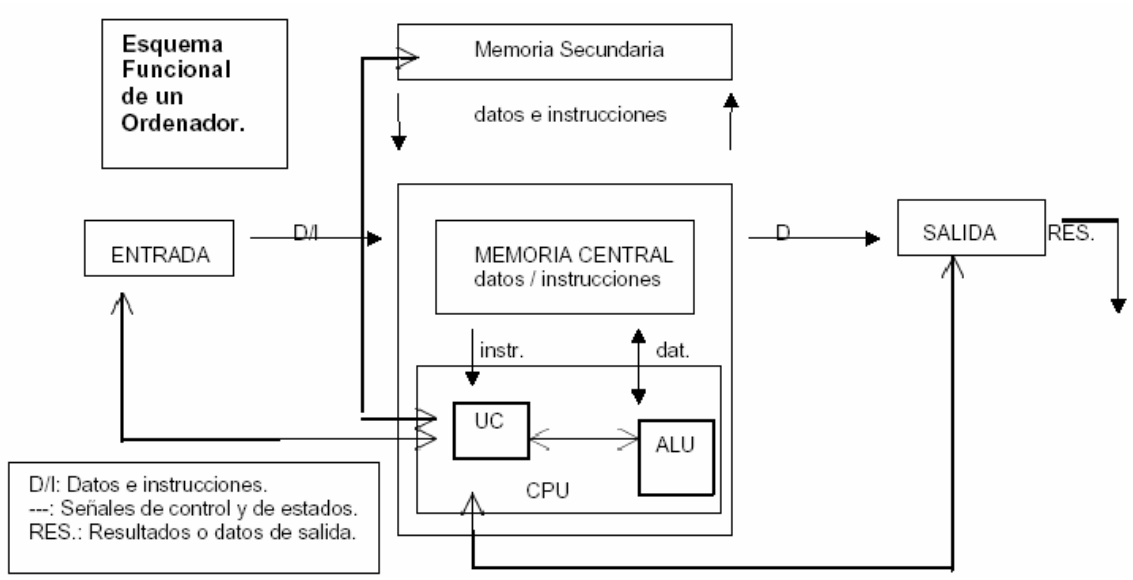
Memoria principal o Central: la memoria que actúa con mayor velocidad y está ligada directamente a las unidades más rápidas de la computadora (Unidad de Control y Unidad Aritmético-Lógica). **Para que un programa (serie de instrucciones) se ejecute debe estar almacenado en la memoria principal. La memoria está dividida en posiciones, denominadas también *palabras* de memoria**, de un determinado número de bits. Normalmente hay un tipo de memoria que sólo se puede leer. Es la llamada memoria ROM (Read Only Memory) y que es permanente, es decir, al desconectar la computadora, su información no se pierde; y otra en la que se puede leer y escribir, RAM (Random Access Memory) que es volátil.

Memoria Secundaria o externa: La memoria principal, aunque es muy rápida no tiene gran capacidad para la información, por lo que **para guardar masivamente información** se utilizan otros tipos de memoria, tales como discos, cintas magnéticas, DAT, streamer, magneto-ópticos, CD-ROM, DVD, que son más lentos pero disponen de mayor capacidad.

Unidad Aritmético-Lógica (ALU)**:** Esta unidad posee los circuitos electrónicos con los que **se hacen las operaciones de tipo aritmético** (sumas, restas, etc.) **y de tipo lógico** (comparaciones, etc.)

Unidad de Control (UC)**:** Detecta señales eléctricas de estado procedentes de las distintas unidades, indicando su situación o condición de funcionamiento. **Lee de la memoria** una a una **las instrucciones** del programa **y genera**, de acuerdo con el código de operación de la instrucción y con las señales de estado, **señales de control dirigidas a todas las unidades**, controlando las operaciones que implican la ejecución de una instrucción.

Se denominan ***periféricos*** de una computadora al conjunto de sus **unidades de entrada y salida y de memoria secundaria**. Al resto de las unidades, es decir, UC, ALU, y Memoria Central se le denomina normalmente **computadora central**. La unidad de procesamiento central o **CPU** (Central Processing Unit) es el conjunto de la ALU y de la UC. Esto es lo que se conoce normalmente como **microprocesador**. La unidad de control contiene un **reloj o generador de pulsos** que sincronizan todas las operaciones elementales del ordenador. La frecuencia del reloj (que viene dada en Mhz) es un parámetro que **determina en parte la velocidad** de funcionamiento del ordenador. Además el **bus de datos** y el **bus de direcciones** permiten el envío de datos y direcciones entre los diferentes componentes.



3.2. Software.

El hardware (Ferretería) engloba todos los elementos físicos (de ahí el nombre) que componen o interactúan con el ordenador. Por contraposición, se denominó "Software" a todo lo que interactúa en el ordenador pero no es tangible. Comprende los datos, el sistema operativo, las metaaplicaciones y el software de aplicación, comercial o de elaboración propia.

Datos: aquellos elementos considerados como unidad de tratamiento dentro de un sistema de proceso de datos. Se distingue entre **datos de entrada**, pendientes de procesar o elaborar, y **datos de salida**, que son los resultados obtenidos una vez elaborados los datos iniciales. Al conjunto de datos se le denomina **información**.

Instrucción: es un conjunto de símbolos que representan una orden de operación para la computadora. Las operaciones suelen realizarse con o sobre datos.

Programa: es un conjunto ordenado de instrucciones que se dan a la computadora indicándole las operaciones o tareas a realizar.

Lenguaje de programación: Las instrucciones se forman con elementos o símbolos tomados de un determinado repertorio y se construyen siguiendo unas reglas precisas. Todo lo relativo a los símbolos y reglas para construir un programa se denomina **lenguaje de programación**. En definitiva, un lenguaje de programación se define según unos **símbolos** y una **sintaxis**.

Sistema operativo: Conjunto de programas básicos encargados de hacer posible el manejo del ordenador y la utilización del resto del software, actuando como intermediario entre el usuario y el hardware. Se encarga de la gestión del procesador (CPU), de la memoria principal y secundaria, y de los dispositivos periféricos. En resumen, controla el flujo de información entre dispositivos, facilita la interactividad usuario-máquina y asigna las tareas y coordina el funcionamiento interno del ordenador de manera eficiente, eficaz y segura.

Software de aplicación: Creado para realizar tareas concretas con la ayuda del ordenador. De ahí su amplia variedad, desde procesadores de texto a editores de video, desde juegos a clientes de mensajería instantánea, etc.

Metaaplicaciones: Engloban todas las aplicaciones destinadas a la creación de programas mediante la utilización de lenguajes de programación. Incluyen **compiladores**, **enlazadores** e **intérpretes**.

4. LENGUAJES DE PROGRAMACIÓN.

Un lenguaje de programación es una notación para escribir programas, es decir, para describir algoritmos dirigidos a un ordenador. Un lenguaje viene dado por una gramática o conjunto de

reglas (sintaxis) que se aplican a un alfabeto.
Según su parecido con el lenguaje natural, se clasifican en:

De bajo nivel:

Lenguaje máquina: Es el único que entiende el ordenador (se compone de unos y ceros) y depende del procesador.

Lenguaje ensamblador: Utiliza símbolos mnemotécnicos para las instrucciones y tiene una correspondencia 1 a 1 con respecto al lenguaje máquina. Depende también del procesador.

ADD B,1
MOV A,B

Al igual que el lenguaje máquina: ocupación mínima de memoria y mínimo tiempo de ejecución.

- De alto nivel:

Pretenden lograr la independencia de la máquina. Son más lentos y necesitan más recursos.

Lenguajes de Alto Nivel de propósito general: se componen de instrucciones formadas por frases parecidas al lenguaje natural (inglés).

Lenguajes de Alto Nivel orientados a objetos: C++, SMALLTALK, EIFFEL, DELPHI, JAVA,

Lenguajes de Alto Nivel especializados: LISP, PROLOG,..

Como el ordenador sólo entiende el lenguaje máquina, necesitamos traducir las instrucciones de los lenguajes de alto nivel y ensambladores. Para ello contamos con *diferentes tipos de programas para traducir las instrucciones*: **Traductores.**

5. APLICACIONES DE LA INFORMÁTICA.

- Investigación científica y humanística. Se utiliza el ordenador como instrumento para la resolución de cálculos matemáticos, recuentos numéricos,etc. conducentes al desarrollo de la investigación. Dentro de este campo se usa el ordenador para operaciones tales como: resolución de ecuaciones y problemas matemáticos en general; análisis de datos; análisis automático de textos; simulación.

- Aplicación técnica. Son aplicaciones en las que se utiliza el ordenador como herramienta para facilitar diseños de ingeniería: trazado de planos, etc.

- Documentación e información (Bases de Datos).Por ejemplo: Documentación científica y técnica, archivos automatizados de bibliotecas, bases de datos con historias clínicas, sistemas de teletexto.

- Gestión administrativa. Contabilidad, facturación, gestión de personal, control de producción, gestión de entidades bancarias...

- Inteligencia artificial. Sistemas expertos, reconocimiento del lenguaje natural, visión artificial.

- Instrumentación y control. Robots industriales, control informático del tráfico, control de contaminación industrial.

6. DISEÑO DE UNA APLICACIÓN INFORMÁTICA.

Una aplicación informática es un conjunto de programas con funcionalidad conjunta. **Uno o varios programas interrelacionados que tienen por objeto la realización de una determinada tarea de forma automática mediante el uso de un sistema informático.** Por ejemplo, una aplicación para llevar la contabilidad de una empresa.

Dependiendo de la metodología que se emplee, los pasos a seguir para la obtención de una aplicación informática pueden variar, pero en general podemos concretarlos en los siguientes:

1.- Análisis del sistema *¿Qué tenemos y qué queremos obtener?* Partimos de un problema a resolver y obtenemos una *especificación* del sistema.
2.- Diseño del sistema *¿Qué vamos a utilizar? ¿Cómo lo vamos a organizar?* Partimos de una *especificación* y obtenemos un algoritmo.
3.- Codificación del algoritmo. Partimos de un algoritmo y obtenemos un programa.
4.- Depuración y pruebas.
5.- Documentación.
6.- Instalación y Mantenimiento de la aplicación.

Otro concepto importante es el de **Ciclo de Vida** de una aplicación informática: es el proceso que se sigue desde el planteamiento de un problema o tarea hasta que se tiene una solución instalada en la computadora y en funcionamiento por los usuarios finales, y mientras ésta sea de utilidad.
Normalmente distinguiremos dos fases dentro del Ciclo de Vida: **Fase de Diseño** (durante la cual no es imprescindible el uso del ordenador) y **Fase de Instalación y Explotación.**

Fase de diseño:
problema **ANALISIS** especificación **DISEÑO** algoritmo
CODIFICACIÓN / PROGRAMACIÓN programa

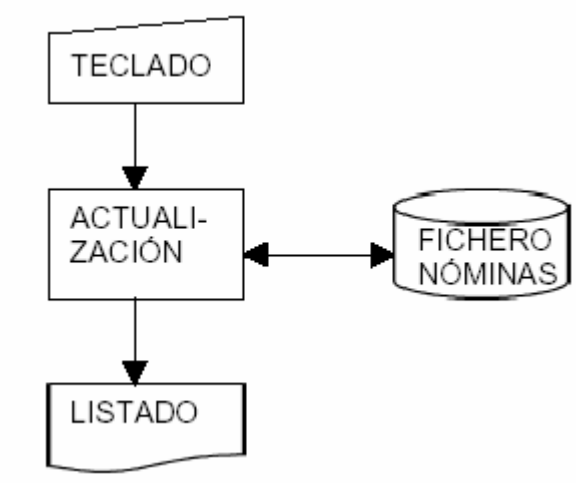
Fase de instalación y explotación:
EDICIÓN **programa fuente** COMPILACIÓN **programa objeto**
MONTAJE, ENLAZADO o LINKING **programa ejecutable** PRUEBA DE EJECUCIÓN
aplicación EXPLOTACIÓN Y MANTENIMIENTO

7. ALGORITMOS Y PROGRAMAS. NOTACIÓN.

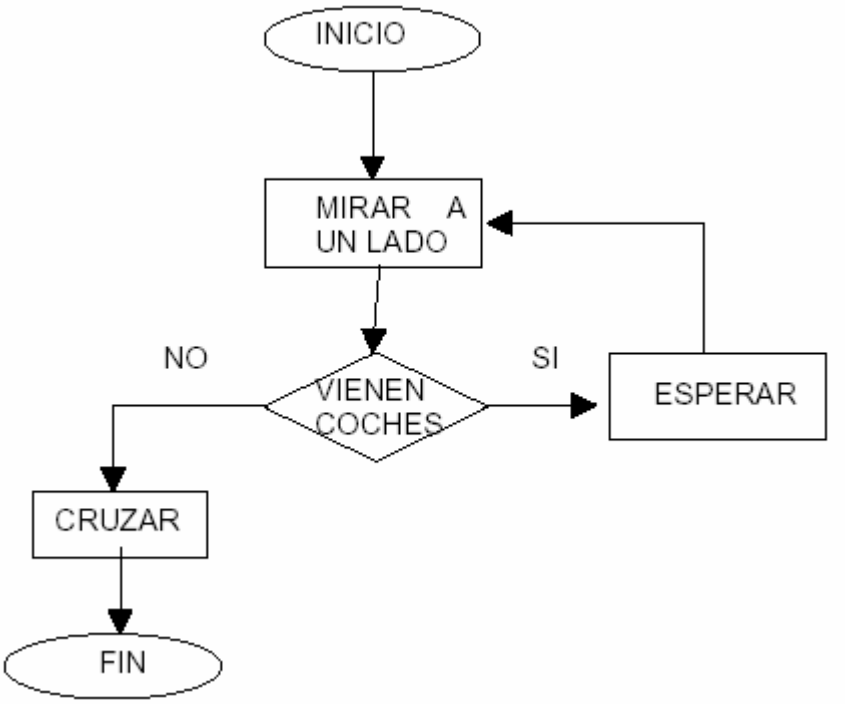
ALGORITMO: La representación de las acciones y operaciones detalladas necesarias a tomar para la resolución de un problema. Debe ser **preciso** (en cuanto al orden de las operaciones), **finito** (en cuanto al número de las operaciones), y **correcto** (debe obtener la solución al problema).
Un mismo problema puede ser resuelto por más de un algoritmo.

Representación de un algoritmo:
GRÁFICA:

- ORGANIGRAMA: Diagrama de flujo del sistema. Flujo de datos einformaciones. Fase de análisis.



- ORDINOGRAMA: Diagrama de flujo del programa: Secuencia lógica de las operaciones. Fase de programación.



LENGUAJE

- PSEUDOCÓDIGO

8. ERRORES.

Si se ha seguido un buen método de diseño, no ha de preocuparnos excesivamente la presencia de errores; lo importante será utilizar un **técnica adecuada de depuración**, que nos permita eliminarlos con facilidad.

Según la fase en la que se detecten, los errores pueden ser:

- Errores de compilación (errores sintácticos): mensajes de error del compilador.

- Errores de ejecución: operaciones no permitidas que se detectan por una parada anormal del programa durante su ejecución. Más difíciles de detectar porque dependen de los datos de entrada. **Lo típico cuando te llama un usuario diciendo que le falla y tu le preguntas qué ha hecho**.

- Errores de lógica: Cuando se obtienen resultados incorrectos. Para detectarlos se emplean ejecuciones de prueba con varios grupos de datos de ensayo. Después se comprueban los resultados con los que se deben obtener.

- Errores de especificación: El peor y el más costoso de corregir, y se debe a la realización de unas especificaciones incorrectas motivadas por una mala comunicación entre el analista y quien plantea el problema. Hay que repetir el trabajo.

9. DOCUMENTACIÓN DE LOS PROGRAMAS.

Facilitan la explotación y el mantenimiento, por lo que conviene que la documentación sea amplia, clara y precisa: **nunca está de más**, tanto para mantener programas hechos por otros o por nosotros. Podemos realizar la siguiente clasificación:

Interna: Dentro del programa fuente

- Comentarios
- Código autodocumentado: identificadores, uso de constantes, ...

Externa: Conjunto de documentos que se acompañan con el programa pero sin formar parte de él.

- Especificaciones del análisis
- Descripción del diseño del programa (ordinogramas, pseudocódigo,...)
- Descripción de las versiones (importante)
- Descripción del programa principal y subprogramas
- Manual de usuario
- Manual de mantenimiento o del programador
- Manual de instalación o de explotación: requisitos mínimos, ...

10. CALIDAD DE LOS PROGRAMAS.

Para un determinado problema se pueden construir diferentes algoritmos de resolución o programas. La elección del más adecuado se debe basar en una serie de requisitos de calidad que adquieren gran importancia a la hora de evaluar el coste de su diseño y mantenimiento.

Las características generales que debe reunir un programa son:

- Legibilidad: ha de ser claro y sencillo para una fácil lectura y comprensión.

- Fiabilidad: ha de ser robusto, es decir, capaz de recuperarse frente a errores o usos inadecuados.

- Portabilidad: su diseño debe permitir la codificación en diferentes lenguajes de programación, así como su instalación en diferentes sistemas.

- Modificabilidad: ha de facilitar su mantenimiento, esto es, las modificaciones y actualizaciones necesarias para adaptarlo a una nueva situación. Relacionado con la legibilidad.

- Eficiencia: se deben aprovechar al máximo los recursos de la computadora minimizando la memoria utilizada y el tiempo de proceso de ejecución. Esto hoy en día no se cumple: Windows, Office, …

Metodología de la programación es el conjunto de métodos y técnicas disciplinadas que ayudan al desarrollo de unos programas que cumplan los requisitos anteriormente expuestos.

Algunos métodos o técnicas:

- PROGRAMACIÓN CONVENCIONAL: Poca claridad y falta de fiabilidad

- PROGRAMACIÓN MODULAR: También llamado diseño modular, descendente o mediante refinamientos sucesivos (top-down, stepwise refinement), se basa en la realización de una serie de descomposiciones sucesivas del problema inicial.

PROGRAMA Una serie de módulos

- PROGRAMACIÓN ESTRUCTURADA (DIJKSTRA): Se basa, entre otras reglas, en el uso exclusivo de las estructuras **secuencia**, **alternativa**, e **iteración** para el control del flujo de ejecución de las instrucciones (nada de GOTO's de los que se utilizan con frecuencia en BASIC, por ejemplo). Produce programas fáciles de verificar, depurar y mantener, que es lo que se pretende.

- PROGRAMACIÓN ORIENTADA A OBJETOS: La Programación Orientada a Objetos (POO) es una forma de programación que utiliza objetos, ligados mediante mensajes, para la solución de problemas. La POO se puede ver como una extensión de la programación estructurada, que intenta potenciar los conceptos de modularidad y reutilización del código.