UNIDAD DIDÁCTICA 1: ALGORITMOS Y PROGRAMAS

1.0.- Introducción.

La principal razón para que las personas aprendan lenguajes y técnicas de programación es utilizar el ordenador como una herramienta para resolver problemas. La resolución de un problema exige al menos los siguientes pasos:

1.- Definición o análisis del problema.

2.- Diseño del algoritmo.

3.- Transformación del algoritmo en un programa.

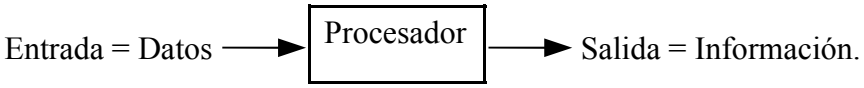
4.- Ejecución y validación del programa.

La palabra "Algoritmo" viene de un matemático persa que vivió en el siglo IX llamado Mohammed Al-Khowârizî.

1.1.- Los sistemas de procesamiento de la información.

En el uso diario, datos e información son esencialmente sinónimos. Sin embargo, los informáticos suelen hacer una diferencia: Datos, se refiere a la representación de algún hecho, concepto o entidad real (palabras escritas, números, dibujos etc). Información implica datos procesados y organizados.

Sistema de procesamiento de información es un sistema que transforma datos brutos en información organizada, significativa y útil.



Ejemplo de Sistema de Información:

Un termostato que controla la temperatura de un edificio es un sistema de procesamiento de la información. La entrada es la temperatura media y la salida es una señal que controla la caldera del aire acondicionado.

El conjunto de instrucciones que especifican la secuencia de operaciones a realizar, en orden, para resolver un sistema específico o clase de problemas, se denomina algoritmo. O sea, un algoritmo es una fórmula para la resolución de un problema.

Ejemplos:

procesadores	Algoritmo
Cocinero	---------> Receta.
Pianista	----------> Partitura.

Cuando el procesador es un ordenador, el algoritmo ha de expresarse de una forma que recibe el nombre de programa.

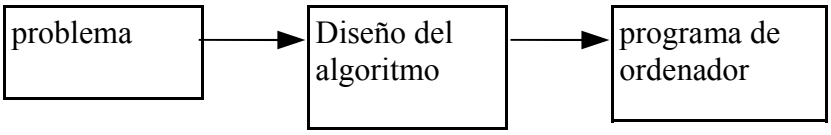
Un programa se escribe en un lenguaje de programación y a la actividad de expresar un algoritmo en forma de programa se le denomina programación.

Cada paso en el algoritmo está expresado por medio de una instrucción en el programa. Por consiguiente, un programa consta de una secuencia de instrucciones, cada una de las cuales especifica las operaciones que debe realizar la computadora.

1.2.- Concepto de Algoritmo.

No olvidemos que el objetivo es el de resolver problemas mediante ordenador.

Un programador de ordenadores es antes de nada una persona que resuelve problemas, por lo que para llegar a ser un programador eficaz se necesita aprender a resolver problemas de un modo riguroso y sistemático. La metodología necesaria para resolver problemas mediante programas se denomina metodología de la programación. La resolución de un problema exige el diseño de un algoritmo que resuelva el problema propuesto.



Los pasos para la resolución de un problema son:

1.- Diseño del algoritmo que describe la secuencia ordenada de pasos que conducen a la solución de un problema dado (Análisis del problema y desarrollo del algoritmo).

2.- Expresar el algoritmo como un programa en un lenguaje de programación adecuado (fase de codificación).

3.- Ejecución y validación del programa por el ordenador.

Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta.

1.2.1.- Características de los algoritmos.

Las características fundamentales que debe cumplir todo algoritmo son:

A) Un algoritmo debe ser preciso e indicar el orden de realización de cada paso.

B) Un algoritmo debe estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.

C) Un algoritmo debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.

La definición de un algoritmo debe describir tres partes: Entrada, proceso y salida. En el algoritmo de receta de cocina por ejemplo se tendrá:

Entrada: Ingredientes y utensilios empleados.

Proceso: Elaboración de la receta en la cocina.

Salida: Terminación del plato.

Ejemplo 1.1:

Un cliente realiza un pedido a una fábrica. La fábrica examina en su banco de datos la ficha del cliente, si el cliente es solvente, entonces la empresa acepta el pedido, en caso contrario rechaza el pedido. Redactar el algoritmo correspondiente.

Solución:

Los pasos del algoritmo son:

1.- Inicio
2.- Leer el pedido.
3.- Examinar ficha del cliente.
4.- Si el cliente es solvente aceptar el pedido, en caso contrario rechazar el pedido.
5.- Fin.

1.3.- Los Lenguajes de Programación.

Para que un procesador realice un proceso, se le debe suministrar en primer lugar un algoritmo adecuado. El procesador debe ser capaz de interpretar el algoritmo, lo que significa:

- Comprender las instrucciones de cada paso.

- Realizar las operaciones correspondientes.

Esas instrucciones estarán dispuestas una a continuación de otras en un determinado lenguaje y formando un programa.

Se pueden clasificar los lenguajes (de forma tradicional) en:

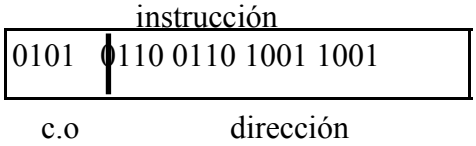
* Lenguaje Máquina.

* Lenguaje de bajo nivel (Ensamblador).

* Lenguaje de alto nivel (Gestión, Científico, Propósito general, específicos etc).

1.3.1.- Lenguajes Máquina.

Son aquellos que están escritos en lenguajes que directamente entiende la máquina (ordenador), ya que sus instrucciones son cadenas binarias (secuencias de ceros y unos) que especifican una operación y las posiciones (dirección) de memoria implicadas en la operación. Se denominan instrucciones de máquina o código máquina (es el conocido código binario).



Las instrucciones en lenguaje máquina dependen del hardware del ordenador y por tanto serán diferentes de un ordenador a otro.

Ventajas:

* Transferir un programa a memoria sin necesidad de traducción posterior, lo que supone una mayor velocidad de ejecución a cualquier otro lenguaje.

Inconvenientes:

* Dificultad y lentitud en la codificación.
* Poca fiabilidad.
* Dificultad grande de verificar y poner a punto los programas.
* Los programas sólo son ejecutables en el mismo procesador.

Para evitar los lenguajes máquina desde el punto de vista del usuario se han creado otros lenguajes que permiten escribir programas con instrucciones parecidas al lenguaje humano (mnemotécnicos en inglés). Estos lenguajes son los de bajo nivel y alto nivel.

1.3.2. Lenguajes de bajo nivel.

Los lenguajes de bajo nivel son más fáciles de utilizar que los lenguajes máquina, pero, al igual que ellos, dependen de la máquina en particular.

El lenguaje de bajo nivel por excelencia es el ensamblador (assembler language). Las instrucciones en lenguaje ensamblador son instrucciones conocidas como mnemónicos (mnemonics). Por ejemplo, mnemotécnicos típicos de operaciones aritméticas son en inglés: ADD, SUB, DIV etc; y en español: SUM, RES, DIV etc.

Una instrucción típica de suma sería:

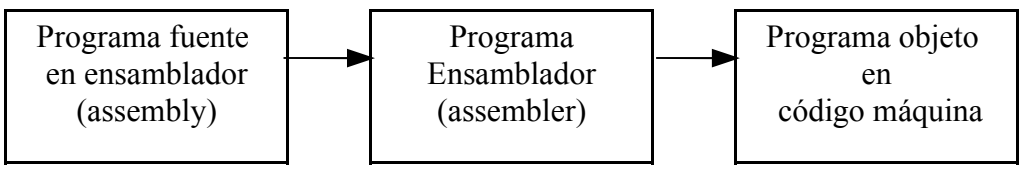
ADD M,N P

Esta instrucción podría significar “Sumar el número contenido en la posición de memoria M al número almacenado en la posición de memoria N y situar el resultado en la posición de memoria P”. Esto es mucho mas sencillo que recordar:

0110 1001 1010 1011

Un programa escrito en lenguaje ensamblador no puede ser ejecutado directamente por la máquina, sino que requiere una fase de traducción al lenguaje máquina.

El programa original escrito en lenguaje ensamblador se denomina programa fuente y el programa traducido en lenguaje máquina se conoce como programa objeto, ya directamente entendible por el ordenador.



Los lenguajes ensambladores presentan la ventaja frente a los lenguajes máquina de su mayor facilidad de codificación.

Los inconvenientes mas significativos de los lenguajes ensambladores son:

* Dependencia total de la máquina, lo que impide la transportabilidad de los programas. El lenguaje ensamblador del PC es distinto que el del Apple Macintosh, por ejemplo.
* La formación de los programadores es mas compleja que la de los programadores de alto nivel puesto que deben conocer bastante el interior de la máquina (registros, memoria, puertos de E/S etc).

1.3.3. Lenguajes de alto nivel.

Los lenguajes de alto nivel son los mas utilizados por los programadores. Están diseñados para que las personas escriban y entiendan los programas de un modo mucho mas fácil que los lenguajes máquina y ensambladores. Otra razón es que un programa escrito en un lenguaje de alto nivel es independiente de la máquina, o sea, las instrucciones del programa del ordenador no dependen del diseño hardware de un ordenador en particular. Por lo tanto los programas escritos en lenguajes de alto nivel son portables o transportables, lo que significa la posibilidad de poder ser ejecutados con poca o ninguna modificación en diferentes tipos de ordenadores.

Los lenguajes de alto nivel presentan las siguientes ventajas:

* El tiempo de formación de los programadores es relativamente corto comparado con otros lenguajes.
* La escritura de programas se basa en reglas sintácticas similares a los lenguajes humanos. Algunos nombres de instrucciones pueden ser: READ, WRITE, PRINT, OPEN etc.
* Las modificaciones y puestas a punto de los programas son más fáciles.
* Reducción del coste de los programas.
* Transportabilidad

Los inconvenientes son:

* Incremento del tiempo de puesta a punto, al necesitarse diferentes traducciones del programa fuente para conseguir el programa definitivo.
* No se aprovechan los recursos internos de la máquina tanto como en los lenguajes máquina y ensamblador.
* Aumento de la ocupación de memoria.
* El tiempo de ejecución de los programas es mucho mayor.

Al igual que sucede con los lenguajes ensambladores, los programas fuente tienen que ser traducidos por programas traductores, llamados en este caso compiladores o intérpretes.

Los lenguajes de alto nivel son muy numerosos, y destacan:

BASIC, COBOL, PASCAL, FORTRAN, C, ADA, MODULA 2, PROLOG, LISP, VISUAL BASIC, VISUAL C, etc.

1.3.4. Traductores de Lenguaje.

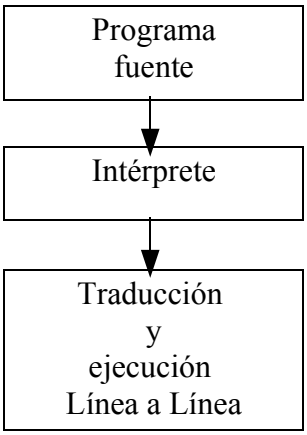
Los traductores de lenguajes son programas que traducen a su vez los programas escritos en lenguajes de alto nivel a código máquina.

Los traductores se dividen en:

* Intérpretes
* Compiladores.

1.3.4.1. Intérpretes.

Un intérprete es un traductor que toma un programa fuente, lo traduce y a continuación lo ejecuta línea a línea.



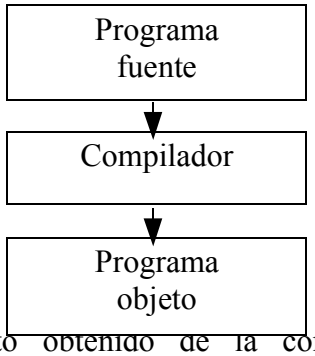
Ejemplo: Basic, Smalltak.

1.3.4.2. Compiladores.

Un compilador es un programa que traduce los programas fuente escritos en lenguajes de alto nivel a lenguaje máquina.

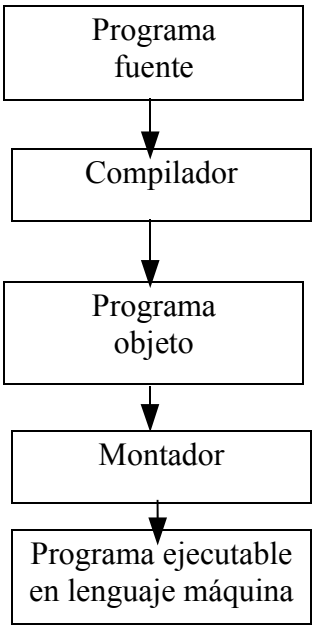
Fases:

La compilación es el proceso de traducción de programas fuente a programas objeto.



El programa objeto obtenido de la compilación no ha sido traducido normalmente a código máquina, sino a ensamblador.

Para conseguir el programa máquina real se debe utilizar un programa llamado montador o enlazador (linker). El proceso de montaje conduce a un programa en lenguaje máquina directamente ejecutable.



El proceso de ejecución de un programa Pascal, por ejemplo, tiene los siguientes pasos:

1.- Escritura del programa fuente con un editor (programa que permite a un ordenador actuar de modo similar a una máquina de escribir) y guardarlo en un dispositivo de almacenamiento (disco).

2.- Introducir el programa fuente en memoria.

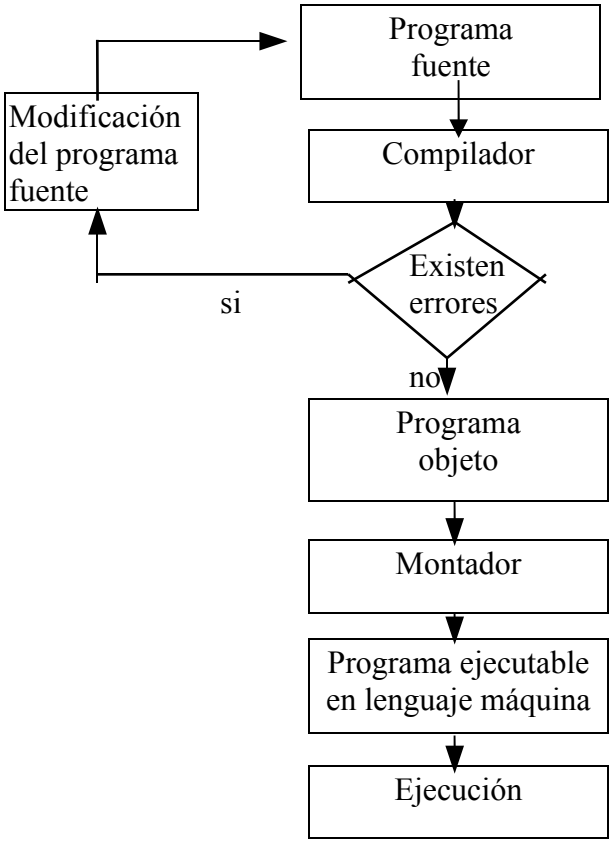
3.- Compilar el programa con el compilador Pascal.

4.- Verificar y corregir errores de compilación (listado de errores).

5.- Obtención del programa objeto.

6.- El montador obtiene el programa ejecutable.

7.- Se ejecuta el programa y si no existen errores, se tendrá la salida del programa.



En la ejecución hay que suministrar datos al programa que se está ejecutando en ese momento en la memoria y se obtendrán unos resultados.

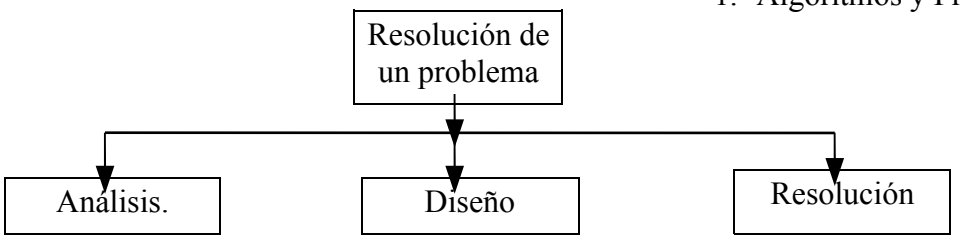
2.1.- La resolución de problemas.

El ordenador es una herramienta para la resolución de problemas. La resolución de problemas la podemos dividir en:

* Análisis del problema.
* Diseño o Desarrollo del algoritmo.
* Resolución del algoritmo en el ordenador.

El primer paso, análisis del problema, requiere un estudio a fondo del problema y de todo lo que hace falta para poder abordarlo; es en sí mismo objeto de estudio, con ciertas metodologías de análisis para poder hacer un mejor estudio inicial y un control de todas las fases del famoso “ciclo de vida” del desarrollo software. Estas metodologías se emplean, por supuesto en grandes proyectos de software.

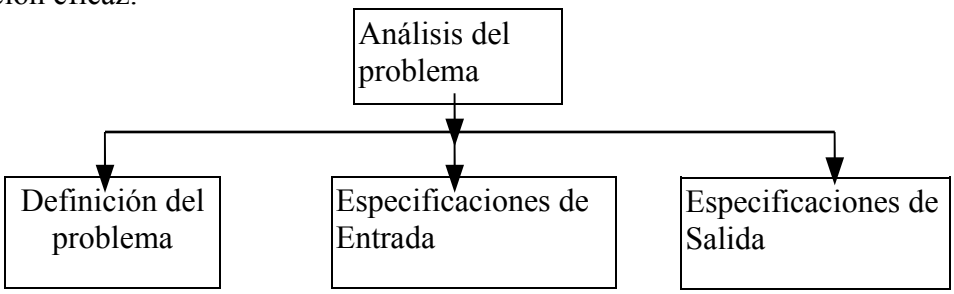
Nosotros nos vamos a centrar mas en los otros dos pasos.



2.2.- Análisis del problema.

El propósito del análisis de un problema es ayudar al programador (Analista) para llegar a una cierta comprensión de la naturaleza del problema.

Una buena definición del problema, junto con una descripción detallada de las especificaciones de entrada/salida, son los requisitos más importantes para llegar a una solución eficaz.



Ejemplo: Leer el radio de un círculo y calcular e imprimir su superficie y su circunferencia.

Análisis:	Entradas:	Radio del circulo (Variable RADIO).
	Salidas:	Superficie del circulo (Variable SUPERFICIE).
		Circunferencia del circulo (Variable LONGITUD)
	Variables:	RADIO, SUPERFICIE, LONGITUD de tipo REAL.

2.3.- Diseño del algoritmo.

Los problemas complejos se pueden resolver mas eficazmente con el ordenador cuando se rompen en subproblemas que sean mas fáciles de solucionar que el original. Este método se suele denominar “divide y vencerás”, y consiste en dividir un problema complejo en otros mas simples. Así el problema de encontrar la superficie y la longitud de un circulo se puede dividir en tres problemas mas simples o subproblemas:

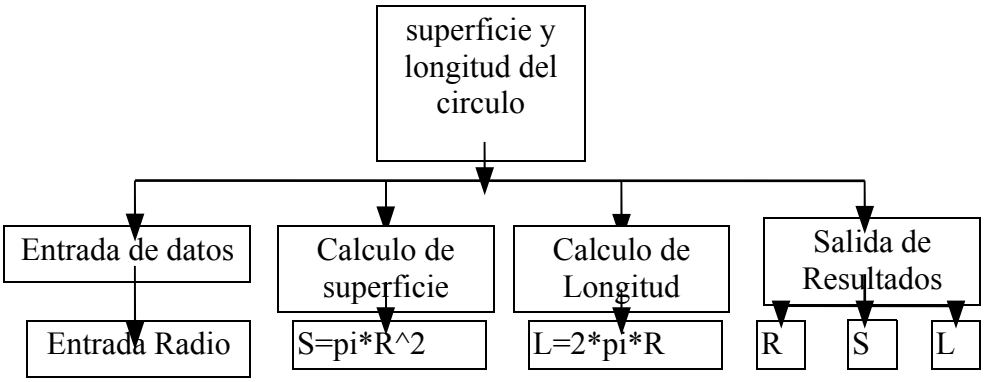
1.- Leer datos de entrada (RADIO)
2.- Calcular Superficie y Longitud.
3.- Escribir los Resultados.

La descomposición del problema original en subproblemas más simples y a continuación dividir estos subproblemas en otros mas simples se denomina diseño descendente (top-down design).

Tras la primera descripción del problema (poco específica), se realiza una siguiente descripción mas detallada con mas pasos concretos. Este proceso se denomina refinamiento del algoritmo.

Subproblema Refinamiento.

Leer Radio	Leer Radio
Calcular Superficie	Superficie=PI*Radio^2.
Calcular Longitud	Longitud=2*PI*Radio
Escribir resultados	Escribir Radio, Longitud, Superficie

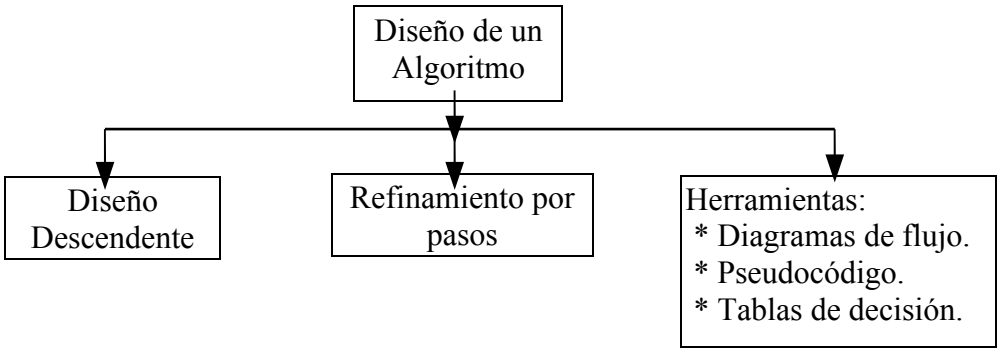


Las ventajas más importantes del diseño ascendente son:

* El problema se comprende más fácilmente al dividirse en partes mas simples denominadas módulos.

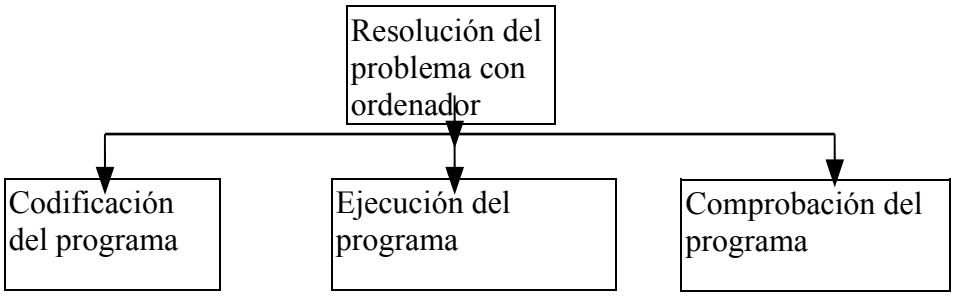
* Las modificaciones en los módulos son más fáciles.

* La comprobación del problema se puede verificar fácilmente.



2.4.- Resolución del problema mediante ordenador.

Una vez que el algoritmo está diseñado y representado gráficamente mediante una herramienta de programación, se debe de pasar a la fase de resolución práctica del problema con el ordenador.



2.5.- Diagramas de flujo.

1.- Algoritmos y Programas

Un diagrama de flujo (flowchart) es una de las técnicas de representación de algoritmos más antiguas y la vez mas utilizada, aunque su empleo ha disminuido considerablemente, sobre todo, desde la aparición de lenguajes de programación estructurados. Como observación es preciso indicar que un diagrama de flujo debidamente realizado permite altos niveles de estructuración, mantener las nuevas corrientes de diseño y modularización y ser seguramente, para los no iniciados la técnica más fácil de aprendizaje en el interesante campo de la diagramación.

En resumen, un diagrama de flujo es una representación gráfica, de las acciones y el orden en que deben realizarse, de una determinada tarea o proceso.

Existen varios tipos de diagramas de flujo, pero todos están comprendidos en dos grandes bloques:

* Organigramas.
* Ordinogramas.

Los organigramas están referidos al sistema, y se realizan mediante bloques, y los ordinogramas lo están al programa siendo más ricos que los anteriores en cuanto a detalles.