ALGORITMOS

DEFINICIÓN: Por [algoritmo](http://es.wikipedia.org/wiki/es%3Aalgoritmo), se entiende a un conjunto *finito* de instrucciones que se deben seguir para resolver un problema. No obstante, desde el punto de vista de la programación de ordenadores, la definición del algoritmo como la especificación de una serie de pasos, es incompleta. Debe observarse que los ordenadores son equipos que tienen limitaciones físicas en cuanto a capacidad de almacenamiento y procesamiento. Por consiguiente debemos refinar un poco más nuestra definición de algoritmo para hacerla aplicable de manera efectiva en el ámbito de la informática.

El algoritmo es un conjunto de pasos, instrucciones o acciones que se deben seguir para resolver un problema. Existen una gran cantidad de algoritmos, hay que coger el más efectivo. Hay dos tipos de algoritmos que son los *cualitativos* y *cuantitativos*, *cualitativos* son todos aquellos pasos o instrucciones descritos por medio de palabras que sirven para llegar a la obtención de una respuesta o solución de un problema, y *cuantitativos* son todos aquellos pasos o instrucciones que involucran cálculos numéricos para llegar a un resultado satisfactorio.

# PSEUDOCÓDIGO:

Lenguaje intermedio entre nuestro lenguaje y el lenguaje de programación.

El principal objetivo del pseudocódigo es el de representar la solución a un algoritmo de la forma más detallada posible, y a su vez lo más parecida posible al lenguaje que posteriormente se utilizara para la codificación del mismo.

Las principales características de este lenguaje son:

* Se puede ejecutar en un ordenador
* Es una forma de representación sencilla de utilizar y de manipular.
* Facilita el paso del programa al lenguaje de programación.
* Es independiente del lenguaje de programación que se vaya a utilizar.
* Es un método que facilita la programación y solución al algoritmo del programa.

Todo documento en pseudocódigo debe permitir la descripción de:

* Instrucciones primitivas
* Instrucciones de proceso
* Instrucciones de control
* Instrucciones compuestas
* Instrucciones de descripción

Estructura a seguir en su realización:
Cabecera:

* Programa:
* Modulo:
* Tipos de datos:
* Constantes:
* Variables:

Cuerpo:

* Inicio
* Instrucciones
* Fin

Para comentar en pseudocódigo se le antepone al comentario dos asteriscos (\*)
Ejemplos

\* Programa que calcula el área de un cuadrado a partir de un lado dado por teclado.

Programa: area\_cuadrado
Modulo: main \*\*( también se puede llamar principal)
Variables:
   lado: natural
   area: natural
Inicio
   Visualizar "Introduce el lado del cuadrado"
   Leer lado
   Area<- lado \* lado
   Visualizar "El área del cuadrado es", area
Fin

\* Programa que visualice la tabla de multiplicar del numero introducido por teclado

Programa: Tabla multiplicar
Modulo: main
Variables:
t: entero
num : entero
Inicio
   Visualizar "Introduce un número"
   Leer num
   Desde t=1 hasta t=10 repetir
      Visualizar num, " X", t, "=", num\*t
   Fin desde
Fin

Una vez que tenemos preparado un diagrama de flujos (ordinograma u organigrama) y un pseudocódigo ya podemos comenzar con la codificación del programa en nuestro ordenador. A partir de aquí todo varía dependiendo del lenguaje de programación que utilicemos, pero en todos los programas tendremos que definir los tipos de datos que utilizaremos. De todo esto hablaré en el siguiente artículo.

CLASES DE ALGORITMOS

Cualitativos:

Son aquellos en los que se describen los pasos utilizando palabras.

Cuantitativos:

Son aquellos en los que se utilizan cálculos numéricos para definir los pasos del proceso.

 Técnicas de diseño de algoritmos

* Algoritmos voraces: Seleccionan los elementos más prometedores del conjunto de candidatos hasta encontrar una solución. En la mayoría de los casos la solución no es óptima.
* Algoritmos paralelos: permiten la división de un problema en sus problemas de forma que se puedan ejecutar de forma simultánea en varios procesadores.
* Algoritmos probabilísticos: algunos de los pasos de este tipo de algoritmos están en función de valores pseudoaleatorios.
* Algoritmos determinísticos: El comportamiento del algoritmo es lineal: cada paso del algoritmo tiene únicamente un paso sucesor y otro antecesor.
* Algoritmos no determinísticos: El comportamiento del algoritmo tiene forma de árbol y a cada paso del algoritmo puede bifurcarse a cualquier número de pasos inmediatamente posteriores, además todas las ramas se ejecutan simultáneamente.
* Divide y vencerás: Dividen el problema en subconjuntos disjuntos obteniendo una solución década uno de ellos para después unirlas, logrando así la solución al problema completo.
* Meta heurísticas: Encuentran soluciones aproximadas (no óptimas) a problemas basándose en un conocimiento anterior (a veces llamado experiencia) de los mismos.
* Programación dinámica: Intenta resolver problemas disminuyendo su coste computacional aumentando el coste espacial.
* Ramificación y acotación: se basa en la construcción de las soluciones al problema mediante un árbol implícito que se recorre de forma controlada encontrando las mejores soluciones.
* Vuelta atrás: se construye el espacio de soluciones del problema en un árbol que se examina completamente, almacenando las soluciones menos costosas

EJEMPLO DE CADA UNA

Cualitativos

1. Algoritmo para insertar tarjeta SIM en un celular.
Datos: conjunto de piezas iniciales (celular, tarjeta SIM)
INICIO
Paso 1. Abra la tapa de la ranura de la tarjeta SIM.
Paso 2. Inserte la tarjeta SIM en la ranura.
Paso 3. Asegúrese que el área de contacto de la tarjeta esté orientada hacia arriba y que la esquina biselada esté orientada hacia el dispositivo.
Paso 4. Presione la tarjeta.
Paso 5. Cierre la tapa de la ranura de la tarjeta.
FIN
2. Algoritmo para cambiar la rueda pinchada de un automóvil
INICIO
Paso 1. Aflojar los tornillos dela rueda pinchada con la llave inglesa.
Paso 2. Ubicar el gato mecánico en su sitio.
Paso 3. Levantar el gato hasta que la rueda pinchada pueda girar libremente.
Paso 4. Quitar los tornillos y la rueda pinchada.
Paso 5. Poner rueda de repuesto y los tornillos.
Paso 6. Bajar el gato hasta que se pueda liberar.
Paso 7. Sacar el gato de su sitio.
Paso 8. Apretar los tornillos con la llave inglesa.
FIN

3. Realizar un puré de papas.
INICIO
1.- Buscar utensilios.
2.- Lavar las papas.
3.- Llenar la olla con agua.
4.- Colocar las papas dentro de la olla.
5.-Encender el fogón.
6.- Colocar la olla al fuego.
7.- Esperar a que hierva.
¿Están listas las papas?
SI: ir al paso 8.
No: ir al paso 7.
8.- Retirar las papas.
9.-Quitar la cascara a las papas.10.- Triturar las papas.
11.- Agregar queso, mantequilla y leche.
12.- Mezclar.
13.- Agregar sal al gusto.
14.- Servir.
FIN
ALGORITMOS CUANTITATIVOS
1. Algoritmo para calcular el área de un triángulo.
• Área del triángulo “x”.
• Base del triángulo “b”.
• Altura del triángulo “h”.
• A ← b\*h/2.

Algoritmo
1.- INICIO.
2.- Suministrar cifra de altura “h”.
3.- Suministrar cifra de base“b”.
4.- Multiplicar las cifras de base y altura.
5.- Dividir el resultado de la multiplicación entre 2.
6.- Generar resultado.
7.- FIN.

QUÉ SON VARIABLES

En [programación](http://es.wikipedia.org/wiki/Programaci%C3%B3n), una variable está formada por un espacio en el sistema de almacenaje ([memoria principal](http://es.wikipedia.org/wiki/Memoria_principal) de un ordenador) y un nombre simbólico (un [identificador](http://es.wikipedia.org/wiki/Identificador)) que está asociado a dicho espacio. Ese espacio contiene una cantidad o información conocida o desconocida, es decir un [valor](http://es.wikipedia.org/wiki/Valor_%28inform%C3%A1tica%29). El nombre de la variable es la forma usual de referirse al valor almacenado: esta separación entre nombre y contenido permite que el nombre sea usado independientemente de la información exacta que representa

QUÉ SON LAS CONSTANTES

Una constante es un valor que no puede ser alterado/modificado durante la ejecución de un programa, únicamente puede ser leído.

Una constante corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos.

EJEMPLO

En [C](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C) las constantes se declaran con la directiva #define, esto significa que esa constante tendrá el mismo valor a lo largo de todo el programa.

El identificador de una constante así definida será una cadena de caracteres que deberá cumplir los mismos requisitos que el de una variable (sin espacios en blanco, no empezar por un dígito numérico, etc.).

Ejemplo:

 #include <stdio.h>

 #define PI 3.1415926

 int main()

 {

 printf("Pi vale %f", PI);

 return 0;

 }

Lo cual mostrará por pantalla:

 Pi vale 3.1415926

Es decir, PI es una constante a la que le hemos asignado el valor 3.1415926 mediante la directiva #define.

La directiva #define también se puede utilizar para definir expresiones más elaboradas con operadores (suma, resta, multiplicación etc) y otras constantes que hayan sido definidas previamente, por ejemplo:

 #define X 2.4

 #define Y 9.2

 #define Z X + Y

REALIZAR LA SIGUIENTE CONSULTA LA CUAL DEBE QUEDAR EN LA PÁGINA WEB DENTRO DE LA CARPETA DE INFORMÁTICA Y SUBCARPETA ALGORITMOS

**CONECTIVOS LÓGICOS Y PROPOSICIONES COMPUESTAS**

**EXPRESIONES LÓGICAS Y MATEMÁTICAS**

Al evaluar una expresión lógica siempre obtendremos un valor de tipo lógico, es decir verdadero o falso. En estas expresiones se pueden utilizar dos tipos de operadores:

* **Operadores Relacionales:** se utilizan para comparar valores de dos expresiones que pueden ser del mismo tipo: aritmética, lógicas, de carácter o de cadena.

|  |
| --- |
| **OPERADORES REALCIONALES EN PSEUDOCÓDIGOS** |
| **<** | Menor que |  |  |  |
| **<=** | Menor o igual que |  |  |
| **>** | Mayor que |  |  |  |
| **>=** | Mayor o igual que |  |  |
| **=** | Igual que |  |  |  |
| **<>** | Diferente que |  |  |

**Ejemplo:**

22 > 13 (aritméticas) **V**

22.5 ‹ 3.44 (aritméticas) **F**

C > f (de carácter) **F**

Verdadero = falso (lógica) **F**

**VALORES DE VERDAD:**

**V (Verdadero)** y en las tablas de verdad **(1)**

**F (falso)** y en las tablas de verdad **(0)**

* **Operadores Lógicos**

Un **operador lógico** actúa exclusivamente, sobre valores de expresiones lógicas. Los operadores lógicos son:

|  |
| --- |
| **OPERADORES LÓGICOS EN PSEUDOCÓDIGOS** |
| **Y** | Conjunción |  |  |
| **0** | Disyunción |  |  |
| **NO** | Negación |  |  |
|  |  |  |  |

**CONJUNCIÓN (AND) (y):** se utiliza para conectar dos proposiciones que se deben cumplir para obtener un resultado verdadero.

**Ejemplo**:

El coche enciende cuando tiene gasolina en el tanque (y) tiene corriente la batería

P: el coche enciende

Q: tiene gasolina en el tanque

R: tiene corriente la batería

**DISYUNCIÓN (OR) (o):** se deduce que si al menos una de las dos expresiones es verdadera, el resultado será verdadero

**Ejemplo:**

Una persona puede entrar al cine si compra su boleto **(u)** obtiene un pase

P: entra al cine

Q: compra su boleto

**REALIZAR UN ALGORITMO QUE MUESTRE TODO EL**

**PROCESO QUE DEBO TENER EN CUENTA PARA**

**REALIZAR LA MATRÍCULA DEL SEMESTRE II**

# Algoritmo Cualitativo

## Ejemplo1: Freír Tajadas

1.- Buscar utensilios.

2.- Quitar la cáscara al plátano.

3.- Sacar las tajadas del plátano.

4.- Ingresar a la cocina a prender la estufa.

5.- Colocarle aceite al sartén.

6.- Esperar que el aceite este caliente.

¿Estará caliente el aceite?Pregunta de verificación.

**SI:** ir al paso 7.

**NO:** ir al paso 6.

7.- Colocar las tajadas en el sartén.

8.- Esperar a que doren las tajadas por un lado.

¿Estarán listas las tajadas por este lado?Pregunta de verificación.

**SI:** ir al paso 9.

**NO:** ir al paso 8 (Revisión).

9.- Voltear las tajadas.

10.- Esperar a que se doren las tajadas.

¿Estarán doradas las tajadas?

**SI:** ir al paso 11.

**NO:** ir al paso 10.

11.- Retirar las tajadas.

12.- Colocar las tajadas en una vasija.

13.- Apagar la estufa.

14.- Colocar adicionales a la tajada

 - Queso, guayaba**.**

**SI:** ir al paso 14 y seleccionar

**NO:** ir al paso 16.

15.- Servir.

16.- Salir de la cocina