

# Diseño de algoritmos

Jose Jesus García Rueda. Adaptado de "El algoritmo, una iniciación a la programación" (<http://www.desarrolloweb.com/manuales/67/>) y de "Diseño estructurado de algoritmos" (<http://www.itver.edu.mx/comunidad/material/algoritmos/>)

## Introducción

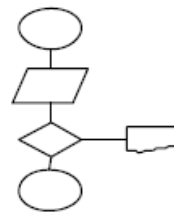
- La principal razón para aprender a programar es utilizar la computadora para resolver problemas.
- "Algoritmo": se deriva de la traducción al latín de la palabra árabe "alkhowarizmi", nombre del matemático árabe que enunció reglas paso a paso para sumar, restar, multiplicar y dividir números decimales.
- Un algoritmo es una serie de pasos organizados que describe el proceso a seguir para solucionar un problema específico.
- Dos tipos de algoritmos:
  - Cualitativos: Emplean palabras. Ej: Receta de cocina, cambiar una rueda, usar la guía telefónica.
  - Cuantitativos: Utilizan cálculos numéricos. Ej: Resolver una ecuación de 2º grado.

## Un algoritmo cotidiano

- Algoritmo para leer las páginas de un libro:
  1. Inicio.
  2. Abrir el libro en la 1ª página.
  3. Leer la página.
  4. ¿Es la última que deseo leer?  
Sí: Ve al paso 7.  
No: Ve al paso 5
  5. Pasar a la siguiente página.
  6. Ve al paso 3.
  7. Cerrar el libro.
  8. Fin.

## Lenguajes algorítmicos

- Un lenguaje algorítmico es un conjunto de símbolos y reglas que permiten describir de manera explícita un proceso.
- Es independiente de cualquier lenguaje de programación.
- Debe permitir una traducción clara del algoritmo al programa.
- Dos tipos de lenguajes algorítmicos:
  - **Gráficos:** Por ejemplo, los diagramas de flujo.
  - **No gráficos:** Por ejemplo, el pseudocódigo.



```
INICIO
  Edad: Entero
  ESCRIBE "¿cuál es tu edad?"
  Lee Edad
  SI Edad >= 18 entonces
    ESCRIBE "Eres mayor de edad"
  FINSI
  Escribe "fin del algoritmo"
FIN
```

## Creación de algoritmos

- Proceso de programación típico:
  - Dado un determinado problema...
  - ...el programador idea una solución...
  - ...y la expresa mediante un algoritmo.
  - Codificación del algoritmo.
  - Ejecución del programa.
- Metodología para la solución de problemas por medio de un ordenador:
  1. Definición del problema: clara y precisa. Es casi la mitad del trabajo...
  2. Análisis del problema: Colocarse en el lugar del ordenador y analizar qué requeriríamos para realizar la tarea.
    - Datos de entrada.
    - Información a producir (salida)
    - Métodos y fórmulas para procesar los datos
  3. Diseño del algoritmo.

## Características de un buen algoritmo

- Debe tener un punto particular de inicio.
- No debe ser ambiguo.
- Debe ser general.
- Debe ser finito en tamaño y en tiempo de ejecución.








Prueba de escritorio: se toman datos específicos como entrada y se sigue el algoritmo hasta obtener un resultado.

## Técnicas de diseño

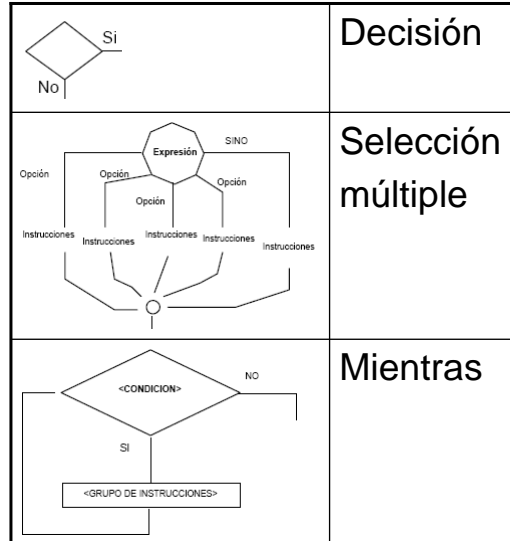
- Top Down: Se descompone sucesivamente el problema inicial en subproblemas.
  - Con cada descomposición, se simplifican los subproblemas.
  - Las diferentes partes del problema pueden ser programadas de forma independiente.
  - El programa final queda estructurado en forma de bloques o módulos.
- Bottom Up: Se programa cada proceso según vaya apareciendo.
  - Difícil llegar a una integración tal que el desempeño global sea fluido.
  - Proclive a la duplicación de esfuerzos.
  - Pueden no satisfacerse los requisitos globales de la aplicación.
- La creación de algoritmos se basa en la técnica descendente.

## Diagramas de flujo

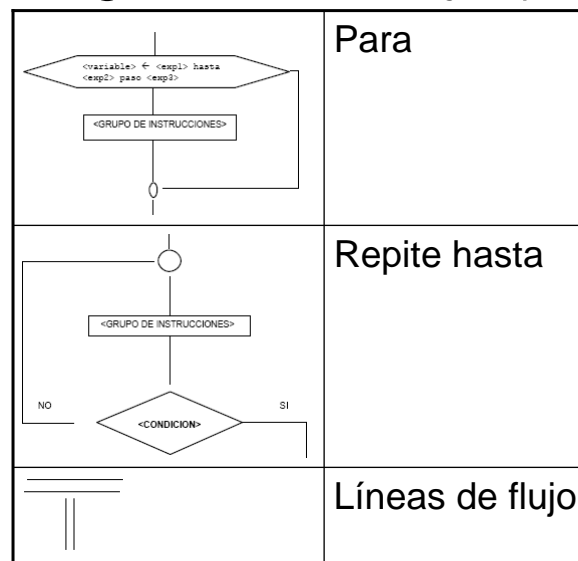
- Es una forma de representar gráficamente un algoritmo.
- Cada paso se escribe dentro de un símbolo.
- Los pasos se conectan unos con otros mediante **líneas de flujo**.
- Son fáciles de diseñar, pero difíciles de actualizar.
- Los símbolos que utiliza están normalizados:

|   |                        |
|---|------------------------|
|  | Inicio/Final           |
|  | Entrada/ Salida        |
|  | Proceso                |
|  | Salida por impresora   |
|  | Conector dentro página |
|  | Conector fuera página  |
|  | Salida por pantalla    |

## Diagramas de flujo (II)

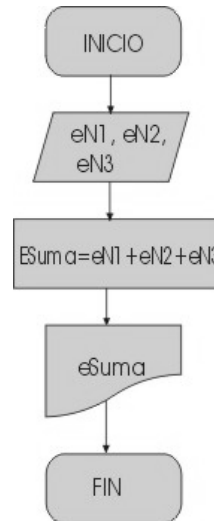


## Diagramas de flujo (III)



## Ejemplo de diagrama de flujo

- Diseñe un algoritmo que lea 3 números, los sume e imprima el resultado por impresora.



## Recomendaciones para los diagramas de flujo

- Emplear solamente líneas de flujo horizontales y/o verticales.
- Evitar el cruce de líneas (usando los conectores)
- Usar los conectores sólo cuando sea necesario.
- No dejar líneas de flujo sin conectar.
- Se deberá poder leer de arriba abajo y de izquierda a derecha.
- Ser escuetos y claros con lo que se escriba dentro de los símbolos.

## Pseudocódigo

- Mezcla de lenguaje de programación y de lenguaje natural.
- Representación narrativa de los pasos que debe seguir un algoritmo.
- Utiliza palabras, no gráficos.
- Ventajas:
  - Ocupa menos espacio.
  - Permite representar fácilmente operaciones repetitivas complejas.
  - Es muy fácil pasar del pseudocódigo al lenguaje de programación.
  - Quedan claros los niveles que tiene cada operación.

## Principales características de los pseudocódigos

- Utilizan operadores aritméticos y lógicos.
- Se pueden incluir comentarios.
- Se debe respetar una indentación en los bloques de instrucciones.
- Usan ciertas palabras clave: PSEUDOCÓDIGO, VARIABLES, INICIO, FIN, LEE, ESCRIBE, IMPRIME, IF\_THEN\_ELSE, CASE OF, FOR DO, WHILE DO, REPEAT UNTIL, ARRAY...
- Comienzan con el nombre del pseudocódigo, seguido de la declaración de variables, y luego el cuerpo del pseudocódigo.

## Ejemplo de pseudocódigo

- Algoritmo que lee 3 números, los suma e imprime su resultado.

PSEUDOCÓDIGO sumatorio

VARIABLES

eN1, eN2, eN3, eSuma: Entero

INICIO

ESCRIBE "Dame tres números:"

LEE eN1, eN2, eN3

$eSuma = eN1 + eN2 + eN3$

ESCRIBE "El resultado de la suma es: ", eSuma

FIN

## Diagramas estructurados (Nassi-Schneiderman)

- Como un diagrama de flujo en el que se omiten las flechas de unión, y las cajas son contiguas.
- Son fáciles de diseñar y difíciles de actualizar.
- Las acciones sucesivas se escriben en cajas sucesivas.

|                                      |
|--------------------------------------|
| Inicio                               |
| Leer<br>Nombre,Hrs,Precio            |
| Calcular<br>$Salario = Hrs * Precio$ |
| Calcular<br>$Imp = Salario * 0.15$   |
| Calcular<br>$Neto = Salario + Imp$   |
| Escribir<br>Nombre, Imp, SNeto       |
| Fin                                  |

## EJEMPLO

- Diseñar un algoritmo que, dada una operación aritmética expresada en el formato habitual (y String), vaya leyéndola carácter a carácter, para así calcular su resultado.
- Respetar las reglas típicas de precedencia entre operaciones aritméticas (+ y - < \* y /)

## Paso 1: Análisis

- Datos de entrada: una cadena de caracteres representando una operación aritmética.
- Datos de salida: Un número, resultado de realizar la operación aritmética.
- Forma de operar: Emplearemos dos pilas, una de operadores y otra de operandos.
  - En la primera guardaremos los operadores que vayamos leyendo, siempre que su precedencia sea mayor que los operadores ya en la pila. De no ser así, se extraen los operadores de precedencia mayor y se operan.
  - En la segunda iremos guardando los operandos que vayamos leyendo, así como los resultados de las operaciones parciales.

## Paso 2: Diseño del algoritmo

- Desarrollo del algoritmo en pseudocódigo:
  - ...

## Paso 3: Prueba de escritorio

- ¿Qué tal funciona el algoritmo para “1\*2\*3+4\*5\*6+1\*3\*4”?