

Introducción a la programación

Diagramas de flujo,
pseudocódigo y
paradigmas

Problema

Situación que se nos presenta y que mediante la aplicación de un algoritmo, pretendemos resolver.



Análisis y Solución de un Problema

Hay que identificar estos aspectos:

Conjunto de Entrada: está compuesto por todos aquellos datos que se ingresan.

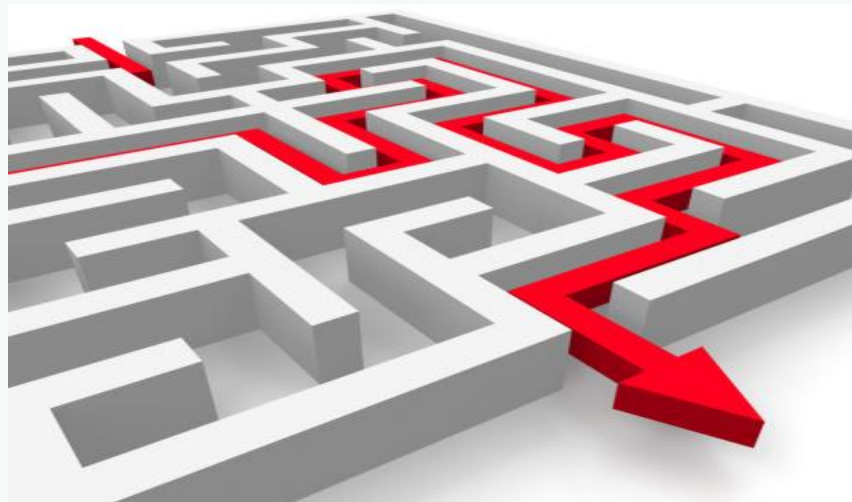
Conjunto de Salidas: está compuesto por todos los datos que el sistema regresará o como resultado del proceso.

estableciendo un algoritmo que convierta el conjunto de entrada en el de salidas



¿Qué es un algoritmo?

Un algoritmo es un conjunto de pasos, procedimientos o acciones que permiten alcanzar un resultado o resolver un problema.



Características

Preciso: Debe indicar el orden de realización de paso y no puede tener ambigüedad

Definido: Si se sigue dos veces o más se obtiene el mismo resultado.

Finito: Tiene fin, es decir tiene un número determinado de pasos.

Correcto: Cumplir con el objetivo.

Eficiente: Realizarlo en el menor tiempo posible

Eficaz: Que produzca el efecto esperado



Algoritmo y programa

- Los algoritmos son independientes de los lenguajes de programación.
- El algoritmo es la infraestructura de cualquier solución, escrita luego en cualquier lenguaje de programación.



Técnicas de representación

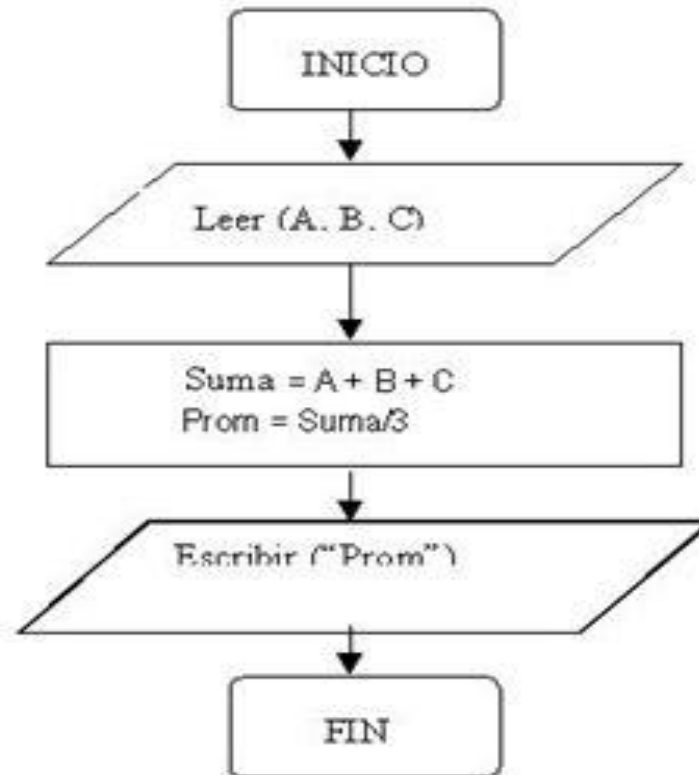
Para la representación de un algoritmo, antes de ser convertido a lenguaje de programación, se utilizan algunos métodos de representación escrita, gráfica o matemática. Los métodos más conocidos son:

- Diagramas de flujo.
- Pseudocódigo.
- Lenguaje natural (español, inglés, etc.).
- Fórmulas matemáticas.



Diagramas de flujo

1. Dadas 3 calificaciones, calcule su promedio



Símbolos



Marca el Inicio y el Fin del diagrama de flujo



Expresa lectura. Cuando el usuario introduce datos



Representa un proceso, Ej. operaciones aritmeticas asignaciones, etc.



Expresa una condición



Expresa salidade datos. Display



Llama a un procedimiento



Conecta las partes de un diagrama de flujo dentro de una misma página



Reglas

1. Los diagramas se escriben de arriba hacia abajo, y/o de izquierda a derecha.
2. Los símbolos se unen con líneas, las cuales tienen en la punta una flecha que indica la dirección que fluye la información procesos, se deben de utilizar solamente líneas de flujo horizontal o verticales (nunca diagonales).
3. Se debe evitar el cruce de líneas
4. No deben quedar líneas de flujo sin conectar
5. Todos los símbolos pueden tener más de una línea de entrada, a excepción del símbolo final.
6. Solo los símbolos de decisión pueden y deben tener más de una línea de flujo de salida.



Pseudocódigo

Sintaxis de pseudocódigo

1. Alcance del programa: Todo pseudocódigo está limitado por las etiquetas de INICIO y FIN. Dentro de estas etiquetas se deben escribir todas las instrucciones del programa.
2. Palabras reservadas con mayúsculas: Todas las palabras propias del pseudocódigo deben de ser escritas en mayúsculas.



Pseudocódigo

3. Sangría o tabulación: El pseudocódigo debe tener diversas alineaciones para que el código sea más fácil de entender y depurar.
4. Lectura / escritura: Para indicar lectura de datos se utiliza la etiqueta LEER. Para indicar escritura de datos se utiliza la etiqueta ESCRIBIR. La lectura de datos se realiza, por defecto, desde el teclado, que es la entrada estándar del sistema. La escritura de datos se realiza, por defecto, en la pantalla, que es la salida estándar del sistema.



Tipos de dato

Carácter ----> "A", "¿", "\$"

Cadena -----> "Hola", "Hola mundo", "Hola mundo (:"

Entero -----> 1, 1987, 110008

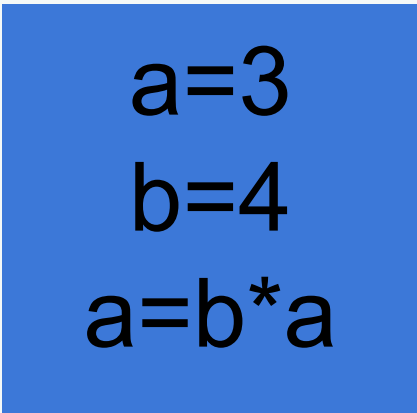
Flotante -----> 12.56, 8.0986

Booleano ---> True, False



Estructura de control secuencial

Sentencias o declaraciones que se realizan una a continuación de otra en el orden en el que están escritas.



```
graph TD; A[a=3] --> B[b=4]; B --> C[a=b*a];
```

$a=3$
 $b=4$
 $a=b*a$

Diagrama de Flujo

$a=3$
 $b=4$
 $a=b*a$

Pseudocodigo

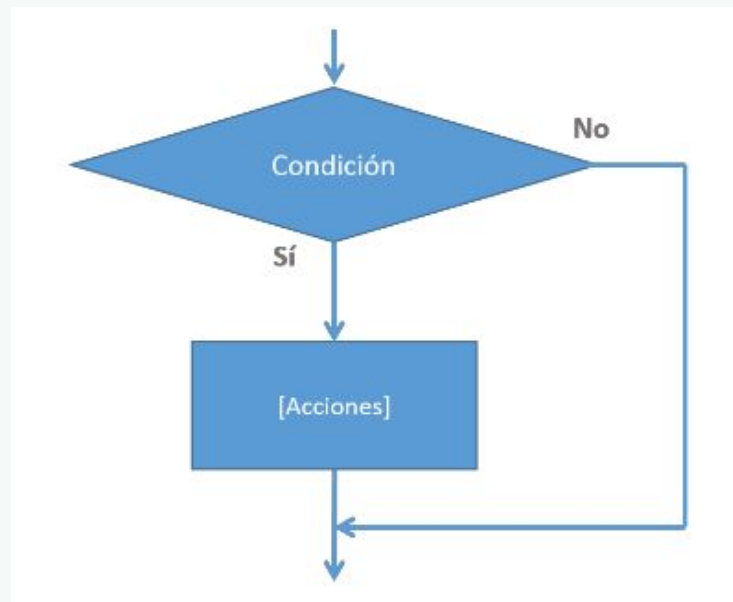
Operadores Aritméticos

Operador	Descripción	Ejemplo
+	Realiza la suma de dos números	$r=2+3$ #r vale 5
-	Realiza la resta de dos números	$r=3-2$ #r vale -1
*	Realiza la multiplicación de dos números	$r=3*2$ #r vale 6
/	Realiza la división de dos números	$r=3/2$ #r vale 1.5
%	Realiza la división y regresa el resto	$r=3\%2$ #r vale 1



Estructuras de control condicionales

Permiten evaluar una expresión lógica (condición que puede ser verdadera o falsa), y dependiendo del resultado, se realiza uno u otra instrucción.



SI condición ENTONCES
[ACCIONES]
FIN SI
DE LO CONTRARIO
[OTRAS ACCIONES]
FIN DE LO CONTRARIO
//resto de acciones

Pseudocódigo

Operadores Relacionales

Operador	Descripción	Ejemplo
$a == b$	¿Son iguales a y b?	$r = 5 == 3$ #es falso
$a != b$	¿Son distintos a y b?	$r = 5 != 3$ #es verdadero
$a < b$	¿Es a menor que b?	$r = 5 < 3$ #es falso
$a > b$	¿Es a mayor que b?	$r = 5 > 3$ #es verdadero
$a \leq b$	¿Es a menor o igual que b?	$r = 5 \leq 3$ #es falso
$a \geq b$	¿Es a mayor o igual que b?	$r = 5 \geq 3$ #es verdadero



Estructura de Control Repetitiva

Permiten ejecutar una serie de instrucciones mientras se cumpla la expresión lógica.

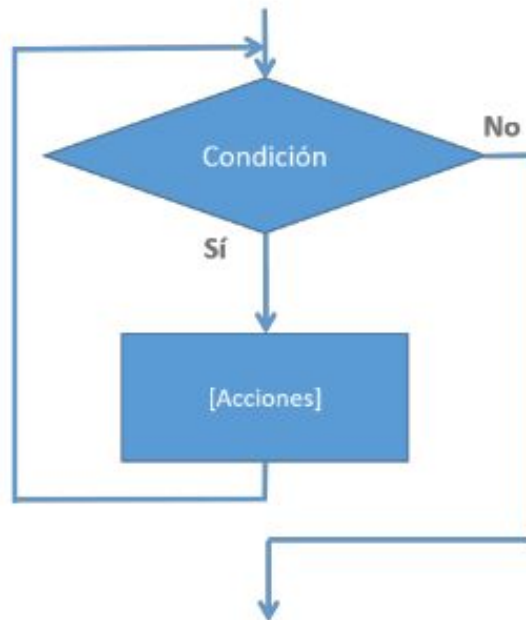


Diagrama de Flujo

```
MIENTRAS condicion  
    [acciones a ejecutar]  
FIN MIENTRAS  
//resto de acciones
```

Pseudocodigo



Paradigma de Programación

Un paradigma de programación es un estilo de desarrollo de programas, es decir, un modelo para resolver problemas computacionales.

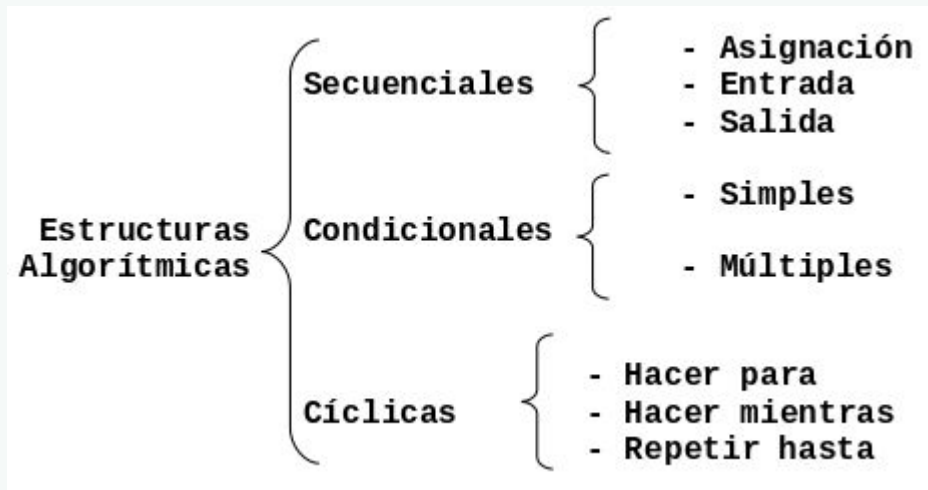
Un paradigma de programación representa un enfoque particular o filosofía para diseñar soluciones.



Paradigma Estructurado

Consiste en resolver un problema utilizando subrutinas (módulos) y tres estructuras básicas:

- Secuencia
- Selección (Condicional)
- Iteración (Repetición)



Paradigma Orientado a Objetos

Consiste en programar y utilizar entes que contienen características y acciones propias que permitiendo solucionar problemas de una forma organizada y rápida.

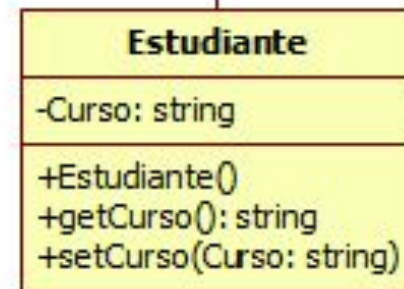
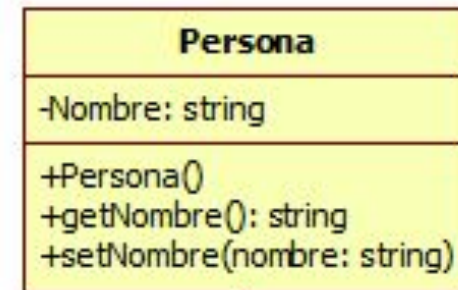
Pilares de POO

Abstracción

Encapsulamiento

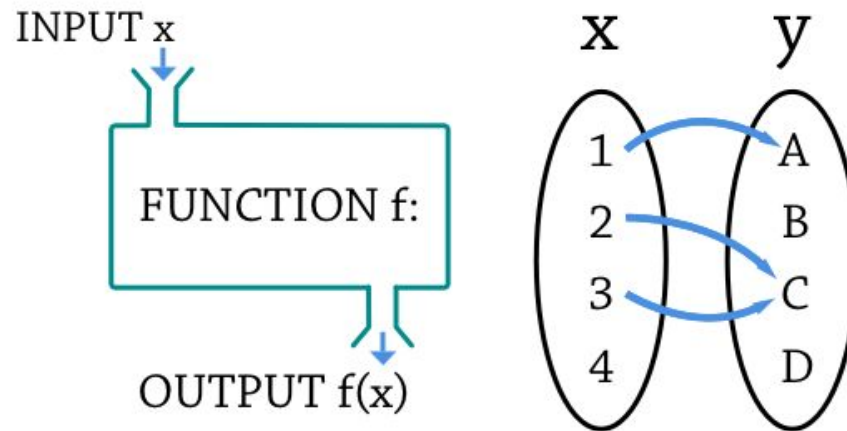
Herencia

Polimorfismo



Paradigma Funcional

Consiste en utilizar lenguajes expresivos y matemáticamente elegantes. No hay ciclos, ni variables.



Ejercicios

Problema 1: Obtener el mayor de dos números diferentes dados (Los números de entrada deben ser diferentes)

Problema 2: Determinar si un número dado es positivo o negativo (El número no puede ser cero)

Problema 3: Imprimir todos los números enteros divisibles entre 3 del 1 al 16



Problema 4: Obtener el valor absoluto de un número

Problema 5: Obtener el factorial de un número dado (El número de entrada debe ser entero positivo o cero)

Problema 6: Dados 3 números diferentes obtener el número menor.

