

* PROGRAMACIÓN

1^a TUTORÍA COLECTIVA

ALGORITMOS y PROGRAMAS

- * **ALGORITMO:** Secuencia ordenada de pasos, descritos sin ambigüedad, para conseguir la solución de un problema.
- * **PROGRAMA:** Algoritmo escrito en un lenguaje de programación.

* Lenguaje natural:

comunicación hombre \Leftrightarrow hombre
(sílabas, palabras, frases)

* Lenguaje máquina:

comunicación máquina \Leftrightarrow máquina
(señales eléctricas, valores 0 y 1)

* Lenguaje de programación:

comunicación hombre \Leftrightarrow ordenador

LENGUAJE DE PROGRAMACIÓN: Conjunto de reglas sintácticas y semánticas, símbolos y palabras especiales establecidas para la construcción de programas.

- * Palabras reservadas: IF, WHILE, LOOP, ...
- * Instrucciones (frases)

*** Lenguaje ensamblador:**

El más cercano al lenguaje máquina.

Palabras reservadas difíciles de recordar (trabaja directamente con el procesador)

*** Lenguaje de alto nivel:**

Más cercano al lenguaje natural.

Abstacta al programador del funcionamiento interno de la máquina.

*** Lenguaje compilado:**

Analizan todo el programa FUENTE en busca de errores y generan código OBJETO. Lo enlazan (LINK) añadiendo bibliotecas y obtienen el código EJECUTABLE.

*** Lenguaje interpretado:**

Se analizan y ejecutan línea a línea. Si hay un fallo en una línea, esta no se ejecuta y pasa a la siguiente, a no ser que la instrucción que falla sea parte importante en el programa.

*** Lenguaje pseudo-compilado:**

El código FUENTE se compila para obtener código binario(BYTECODES) que es interpretado por la JVM, según la máquina en la que se ejecute. (INDIFERENTE DE LA PLATAFORMA).

CICLO DE VIDA DEL SOFTWARE

- * Análisis
- * Diseño (pseudocódigo)
- * Codificación (lenguaje de programación)
- * Pruebas y documentación
- * Instalación y paso a Producción
- * Mantenimiento

ESTRUCTURAS DE DATOS

DATOS

Tienen tres atributos:

- * Nombre: con el que se identifica (*sueldo*)
- * Tipo: conjunto de valores que puede tomar (*entero*)
- * Valor: elemento del tipo que se le asigna en un determinado momento (2500)

¿Qué es un dato?

Desde el punto de vista de la Informática un dato podemos definirlo como todo aquello que puede ser almacenado de forma independiente.



The diagram illustrates five types of data:

- Frase:** En un lugar de la mancha...
- Imagen:** A ladybug icon.
- Número:** 5
- Sonido:** Speaker icon.
- Carácter:** K

TIPOS DE DATOS BÁSICOS

- * Numéricos

- * Enteros: 13

- * Reales: 13.45

- * No numéricos

- * Carácter: 'A'

- * Lógicos: Verdadero, Falso

- * Estructurados

- * Cadenas de caracteres: 'hoy es lunes'

VARIABLES

Datos cuya información puede ser **variable** durante la ejecución del programa. Deben ser definidas con:

- * Un identificador
- * Un tipo de dato: Numéricas, alfanuméricas, lógicas

CONSTANTES

Datos cuyo valor **no cambia** durante la ejecución del programa.

EXPRESIONES

Combinaciones de constantes, variables, símbolos de operación (operadores) y funciones especiales.

OPERADORES BÁSICOS

* ARITMÉTICOS: +, -, *, /, %, ^

* RELACIONALES: =, <>, >, >=, <, <=

* LÓGICOS: NO, Y, O

* ALFANUMÉRICOS: +

		Negación	Conjunción	Disyunción
A	not A	A and B	A or B	
V	F	V	V	V
F	V	F	V	V
F	F	F	F	F

Expresiones.

Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales, con reglas específicas de construcción. Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones existentes.

Según sea el resultado que producen y los operadores que utilizan se clasifican en:

- Numéricas: son las que producen resultados de tipo numérico.
- Alfanuméricos: son las que producen resultados alfanuméricos.
- Lógicas: son las que producen resultados verdadero o falso.

Funciones.

Operadores espaciales que se denominan “funciones internas”, incorporadas o estándar.

Cada lenguaje de programación tiene sus propias funciones, entre las comunes y más utilizadas están las siguientes:

- $\text{sqrt}(x)$ raíz cuadrada de un número positivo
- $\text{abs}(x)$ valor absoluto
- $\text{cos}(x)$ coseno
- $\text{sin}(x)$ seno

Cada lenguaje de programación utiliza su propia simbología para algunos operadores. Por ejemplo en Java el operador lógico AND se representa mediante `&&`, el operador lógico OR mediante `||`, el operador MOD se representa mediante `%`.

Orden de prioridad de los operadores.

Dentro de las expresiones hay que tener un orden de prioridad de los operadores, que depende del lenguaje de programación utilizado, pero que de forma general se puede establecer de mayor a menor prioridad de la siguiente forma:

- Paréntesis (Comenzando por los mas internos).
- Signo.
- Potencia.
- Producto, división, módulo.
- Suma, resta.
- Concatenación.
- Relacionales.
- Negación.
- Conjunción. AND
- Disyunción. OR

La evaluación de los operadores de igual orden se realiza siempre de izquierda a derecha.

CONSTRUCCIÓN DE ALGORITMOS

PSEUDOCÓDIGO

Es un lenguaje de especificación de algoritmos que utiliza palabras reservadas y exige la indentación, o sea sangría en el margen izquierdo de algunas líneas. En nuestros pseudocódigos usaremos determinadas palabras en español como palabras reservadas.

Debe posibilitar la descripción de los siguientes elementos:

- * Instrucciones de E/S
- * Instrucciones de proceso
- * Sentencias de control de flujo
- * Subprogramas
- * Comentarios

PSEUDOCÓDIGO - Ejemplo

Programa ConversionCentigradosKelvin

Entorno

 centigrados, kelvin: Numérico real

inicio

 leer centigrados

 kelvin ← centigrados + 273.15

 escribir "El equivalente en kelvin es ", kelvin

fin

```
1 Proceso ConversionCentigradosKelvin
2     leer centigrados
3     Kelvin <-centigrados + 273.15
4     escribir "El equivalente en kelvin es ",Kelvin
5 FinProceso
```

INSTRUCCIONES PRIMITIVAS

* **ENTRADA:** recogen datos y los cargan en memoria

Leer nombre

* **SALIDA:** devuelven datos en un dispositivo de E/S

Escribir salarioNeto

* **ASIGNACIÓN:**

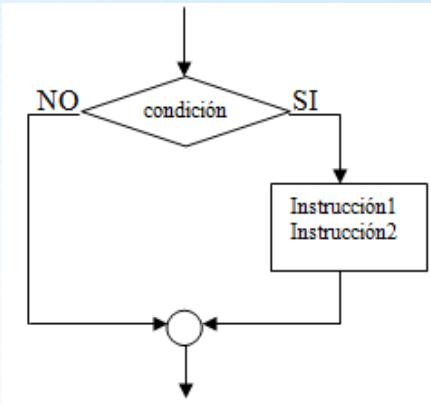
salarioNeto ← salarioBruto - descuento

INSTRUCCIONES ALTERNATIVAS o DE CONTROL

Controlan la ejecución de uno o varios bloques de instrucciones

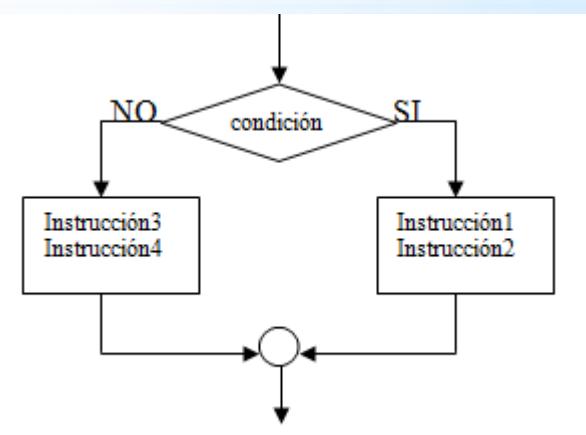
* SIMPLE:

```
si <condicion> entonces  
    <acciones>  
finsi  
si a=b entonces  
    Escribir "Valores iguales"  
Finsi
```



* DOBLE:

```
si <condicion> entonces  
    <acciones1>  
sino  
    <acciones2>  
finsi  
si a=b entonces  
    Escribir "Valores iguales"  
sino  
    Escribir "Valores diferentes"  
finsi
```



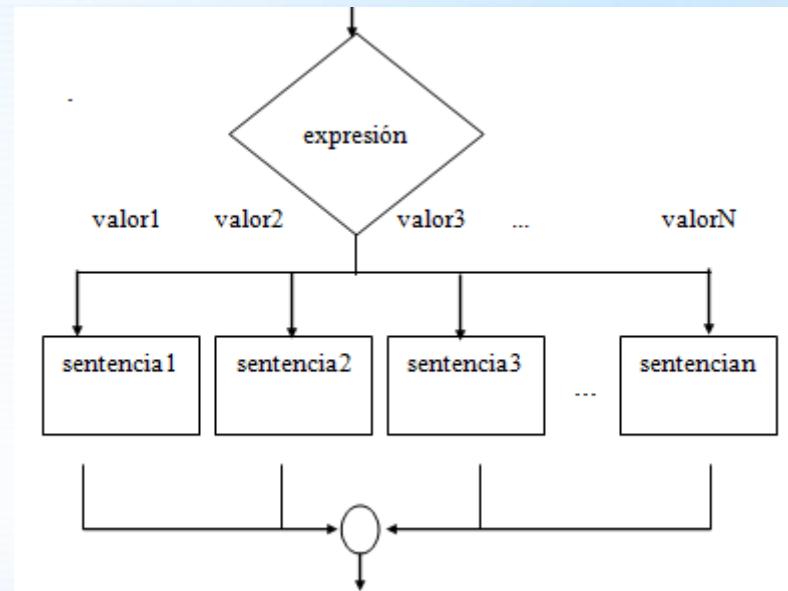
INSTRUCCIONES ALTERNATIVAS o DE CONTROL

Controlan la ejecución de uno o varios bloques de instrucciones

* MÚLTIPLE:

```
según valor_expresión hacer
    valor 1: instrucciones1
    valor 2: instrucciones2
    ...
    valor n: instrucciones n
    en otro caso: otra_instrucción
finsegún
```

```
según color hacer
    'R': Escribir 'Rojo'
    'B': Escribir 'Blanco'
    'A': Escribir 'Azul'
    en otro caso: Escribir 'Carácter erróneo'
finsegún
```



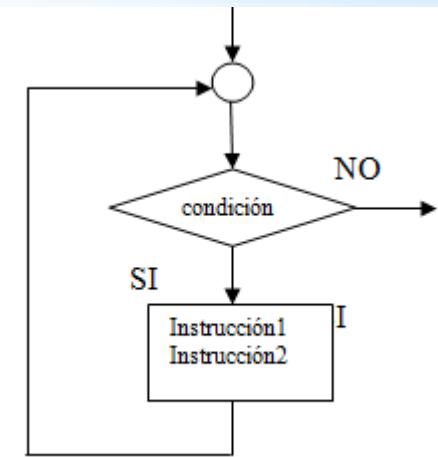
INSTRUCCIONES REPETITIVAS

Ejecutan repetidamente un conjunto de instrucciones, el número de veces que indica una condición.

* **MIENTRAS:** Puede que las instrucciones no se ejecuten nunca.

```
mientras condición hacer  
    instrucción 1  
    instrucción 2  
    ...  
    instrucción n  
finmientras
```

```
n ← 1  
mientras n<=10 hacer  
    Escribir n  
    n ← n+1  
finmientras
```

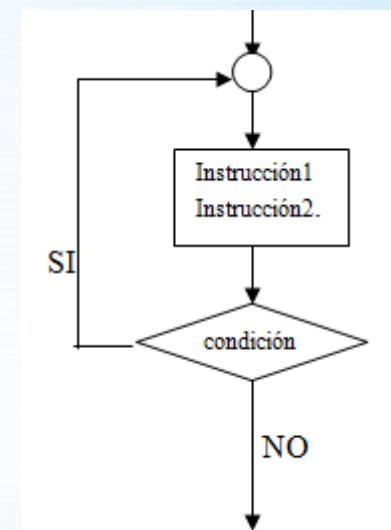


INSTRUCCIONES REPETITIVAS

* **REPETIR:** Las instrucciones se ejecutan al menos una vez.

```
repetir  
    instrucción 1  
    instrucción 2  
    ...  
    instrucción n  
mientras condición
```

```
n ← 1  
repetir  
    Escribir n  
    n ← n+1  
mientras n<10
```

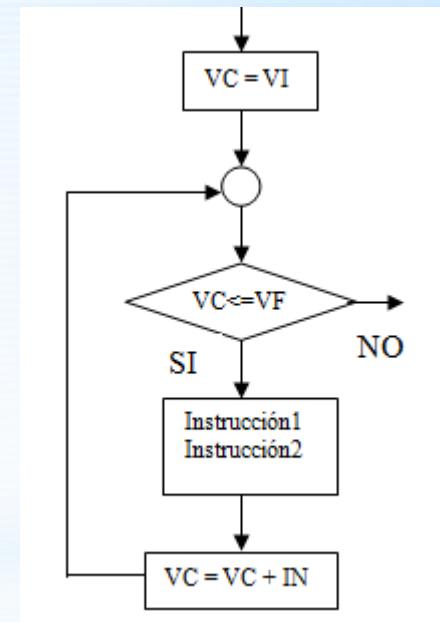


INSTRUCCIONES REPETITIVAS

* PARA: Las instrucciones se ejecutan un número determinado de veces que se conoce de antemano.

```
para Vc de Vi a Vf con Inc=n  
    instrucción 1  
    instrucción 2  
    ...  
    instrucción n  
finpara
```

```
para n de 1 a 10 con Inc=1  
    Escribir n  
finpara
```



ELEMENTOS AUXILIARES

***CONTADOR:** Es una variable cuyo valor se incrementa en una cantidad fija, positiva o negativa.

Ejemplo: leer una secuencia de notas de una asignatura y contabilizar el número de suspensos y de aprobados. La secuencia de notas termina al teclear una nota negativa.

```
PROGRAMA: Notas
ENTORNO:
    CA, CS: numérica entera //CA: Contador de aprobados
    N: numérica real           //CS: Contador de suspensos
ALGORITMO:
    CA←0
    CS←0
    Escribir "Introduzca nota: "
    Leer N
    mientras N >= 0 hacer
        si N >= 5 entonces
            CA ←CA + 1
        sino
            CS ←CS + 1
        finsi
        Escribir "Introduzca nota: "
        Leer N
    finmientras
    si CA = 0 Y CS = 0 entonces
        Escribir "No introdujo ninguna nota"
    sino
        Escribir "El número de aprobados es: ", CA
        Escribir "El número de suspensos es: ", CS
    finsi
FIN Notas
```

ELEMENTOS AUXILIARES

* **ACUMULADOR:** Es una variable cuyo valor se incrementa sucesivas veces en cantidades variables.

Ejemplo: Programa que permita calcular la suma de los 10 primeros números naturales pares.

```
PROGRAMA: Acumular
ENTORNO:
    S, P, N: numérica entera
ALGORITMO:
    S←0           //S: acumulador para suma
    P←1           //P: acumulador para producto
    para N de 0 a 20 con Inc=2
        S←S+N
        P←P*N
    finpara
    Escribir "La suma es: ", S
    Escribir "El producto es: ", P
FIN Acumular
```

ELEMENTOS AUXILIARES

* **INTERRUPTOR o SWITCH:** Es una variable que solamente puede tomar dos valores exclusivos (0/1, verdadero/falso, si/no, etc.)

Ejemplo: Programa que sume independientemente los pares e impares de los números comprendidos entre -100 y 100.

```
PROGRAMA: Pares_impares
ENTORNO:
    SP, SI, N, SW: numérica entera
ALGORITMO:
    SPar<0
    SImpar<0
    SW<-1      //switch: si vale 1 es par, si vale -1 es impar
    para N de -100 a 100 con Inc=1
        SW = -1 * SW
        si SW = 1 entonces
            SPar <= SPar + N
        sino
            SImpar <= SImpar + N
        finsi
    finpara
    Escribir "La suma de los pares es: ", SPar
    Escribir "La suma de los impares es: ", SImpar
FIN Pares_impares
```

PROGRAMACIÓN MODULAR

- * El resultado final de la programación modular es la **división** de un programa en un conjunto de módulos independientes que se comunican entre sí a través de llamadas.
- * Un módulo representa **una tarea** determinada, identificada por un nombre (nombre del módulo) por el cual será invocada desde otros módulos.
- * Siempre debe haber un módulo que describa la solución completa y que servirá de nexo de unión entre los demás, es el **programa principal**.
- * Los módulos se **comunican** mediante variables de enlace a las que se denomina **parámetros** o argumentos.
- * Los módulos se llaman funciones si devuelven algún argumento y procedimientos si no devuelven ningún valor. En Java hablamos de métodos.
- * **Variables Globales.**- Son aquellas variables que pueden ser accedidas y modificadas desde cualquier módulo o función. En general las variables globales se declaran fuera de cualquier módulo.
- * **Variables Locales.** - Son aquellas variables que pueden ser accedidas y modificadas únicamente desde el módulo o función en que fueron declaradas.

PROGRAMACIÓN MODULAR con PSeInt

```
1 // funcion que no recibe ni devuelve nada
2 SubProceso Saludar()
3   Escribir "Hola mundo!"
4 FinSubProceso
5
6 // funcion que recibe un parámetro y devuelve su doble
7 SubProceso res1 <- CalcularDoble(num)
8   res1 <- num*2 // retorna el doble
9 FinSubProceso
10
11 // funcion que recibe un parámetro y devuelve su triple
12 SubProceso res2 <- Triplicar(num)
13   res2 <- num*3
14 FinSubProceso
15
16 // proceso principal, que invoca a las funciones antes declaradas
17 ▼ Proceso PruebaFunciones
18   Escribir "Llamada a la función Saludar:"
19   Saludar() // como no recibe argumentos pueden omitirse los paréntesis vacíos
20   Escribir "Introduce un valor numérico para x:"
21   Leer x
22   Escribir "Llamada a la función CalcularDoble"
23   Escribir "El doble de ",x," es ", CalcularDoble(x)
24   Escribir "Llamada a la función Triplicar"
25   Escribir "El tripleY de ",x," es ", Triplicar(x)
26 FinProceso
```

num es el parámetro

res1 es el valor que devuelve