



UT 1

Introducción a la programación

—

Módulo de Programación 1º DAW



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

Autor: Fran Gómez

Objetivos

RA1: Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

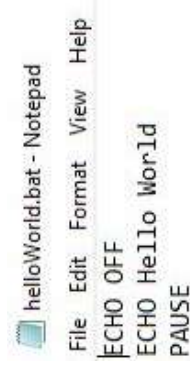
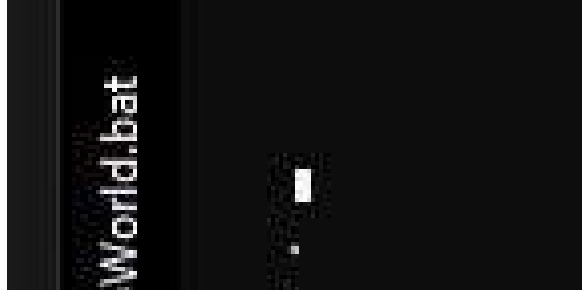
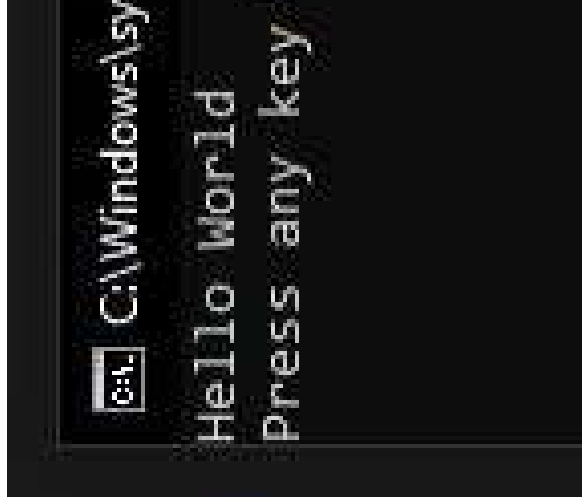
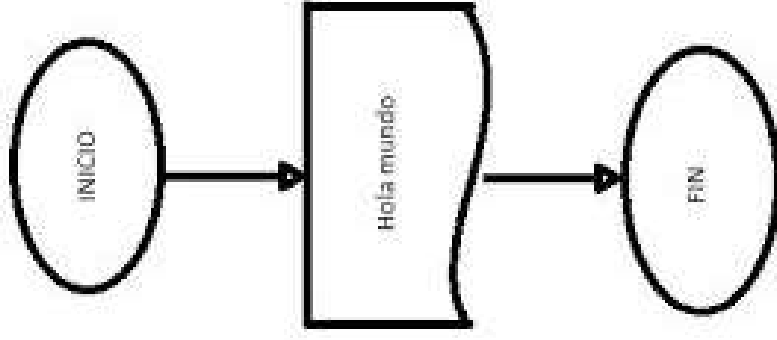
- Conocer los conceptos básicos relacionados con la programación y el diseño de aplicaciones
- Describir los paradigmas de programación más usados
- Aprender a utilizar sistemas de descripción de programas de alto nivel

Índice de contenidos

1. Introducción
 - 1.1. Evolución de la programación
 - 1.2. Conceptos básicos
 - 1.3. Clasificar los lenguajes de programación
2. Paradigmas de programación
 - 2.1. Declarativa
 - 2.2. Imperativa
 - 2.3. Estructurada
 - 2.4. Modular (cohesión y acoplamiento)
3. Elementos de un programa
4. Palabras reservadas
5. Operadores
6. Tipos de datos
7. Estructuras
 - 7.1. Secuenciales
 - 7.2. Selectivas
 - 7.3. Iterativas
 - 7.4. Modulares
8. Pseudocódigo
9. Diagramas de flujo
- ~~10. Tablas de decisión~~
- ~~11. Ciclo de vida del software~~
12. Recursos y referencias

Flickr

Introducción

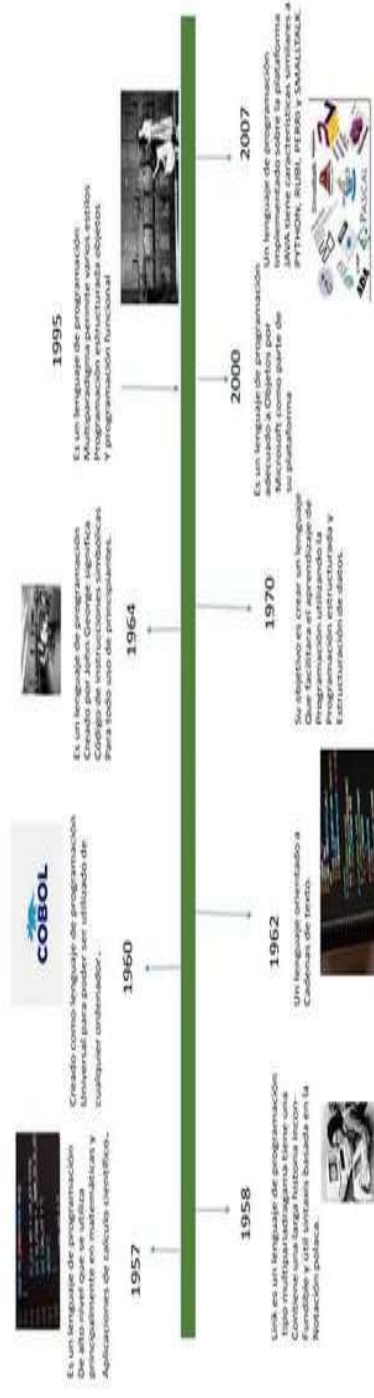


Orígenes de la programación

Ejercicio

Crea una línea del tiempo con los items que más destacarías en la historia de la programación.

1. Año
2. Nombre (logotipo)
3. Creador
4. Una característica

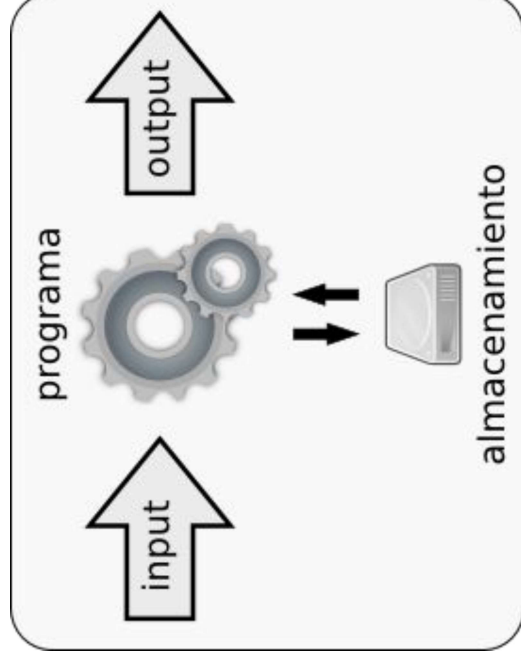


Conceptos de programación

- Programación
- Algoritmo
- Programar
- Código
- Lenguaje binario
- Lenguaje ensamblador
- Lenguajes de alto nivel

- Entrada
- Salida
- Cambio de estado

- Lógica
- Datos



Clasificación de los lenguajes

- Propósito:
 - General vs específico
- Tipo:
 - Scripting (o interpretado)
 - Frontend
 - Backend
 - Compilado
 - Desktop
 - Marcado
 - Móvil
- Uso:
 - Desarrollo Web
 - Juegos
 - Sistemas
 - Móvil
 - IA, BigData, Machine Learning...
- Plataforma:
 - Web
- Paradigma:

Paradigmas de programación

Ejercicio

Cita todos los paradigmas de programación que encuentres en la Web y un lenguaje de ejemplo

- **Imperativa** ⇒ Instrucciones paso a paso
 - **Estructurada** o procedural => Instrucciones siguiendo estructuras
 - **Declarativa** ⇒ Resultado deseado
 - **Modular** ⇒ Divide el programa en partes
 - **Orientada a objetos** ⇒ Agrupa Datos y Lógica
- Otros:
- Funcional ⇒ funciones matemáticas
 - Lógico ⇒ reglas lógicas
- Son subtipos de **Declarativa**

Paradigmas de programación

| Característica | Declarativa | Imperativa | Estructural | Modular | Orientada a Objetos |
|----------------------|---|---|--|--|---|
| Definición | Se enfoca en el <i>qué</i> se quiere lograr, sin describir el <i>cómo</i> . | Se enfoca en el <i>cómo</i> se deben realizar las tareas, con pasos detallados. | Extensión del paradigma imperativo, pero centrada en bloques de código llamados estructuras. | Divide el programa en módulos o unidades independientes que pueden ser reutilizados. | Organiza el código en objetos, que combinan datos y comportamiento. |
| Ejemplo de lenguajes | SQL, Prolog, Haskell | C, Assembly, Python (en algunos contextos), JavaScript | C, Pascal, ALGOL, Fortran | Ada, Modula-2, Python (por sus módulos) | Java, C++, Python, Ruby, Smalltalk |

Programación Declarativa

Queremos programar la obtención de los clientes de Madrid mayores de 30 años.

```
SELECT nombre, apellido  
FROM clientes  
WHERE ciudad = 'Madrid' AND edad > 30;
```

Declaración, no instrucción: No le dices al ordenador cómo buscar a los clientes, simplemente **declaras** que quieres los nombres y apellidos de aquellos que cumplen las condiciones.

Foco en el resultado: Te centras en el qué quieres obtener, no en el **cómo** lo vas a obtener.

Motor de base de datos: El motor de base de datos se encarga de optimizar la consulta y encontrar los datos que cumplen con los criterios especificados.

Programación Imperativa

Queremos programar la elaboración de un sandwich

```
#include <stdio.h>

int main() {
    printf("Tomando una rebanada de pan\n");
    printf("Untando mantequilla\n");
    printf("Colocando queso\n");
    printf("Tomando otra rebanada\n");
    printf("Cerrando el sandwich\n");
    return 0;
}
```

Instrucciones secuenciales: Cada línea de código representa una instrucción específica que se ejecuta una después de la otra.

Estado mutable: El estado del sandwich va cambiando a medida que se ejecutan las instrucciones.

Foco en el proceso: Nos enfocamos en describir cómo hacer el sandwich, paso a otro.

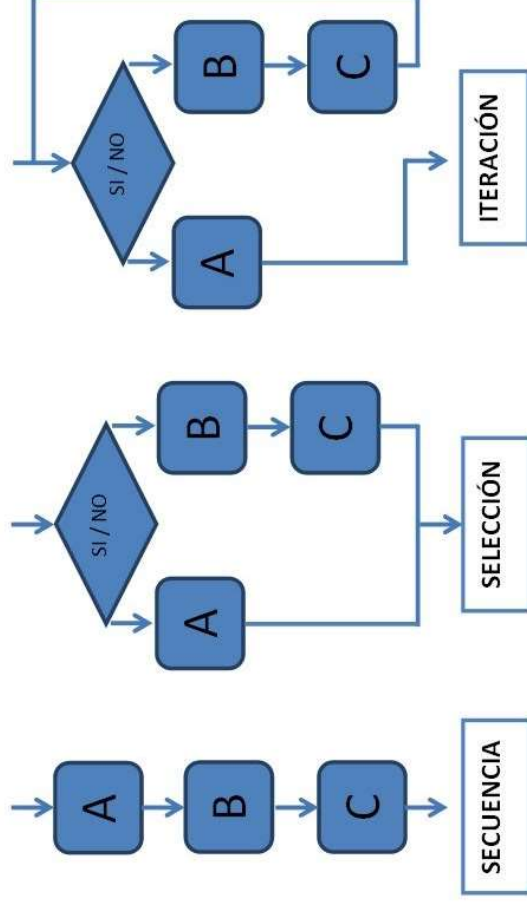
Ejercicio 2

Rellena una tabla como la siguiente, comparando los distintos lenguajes de programación

| Lenguaje | Propósito | Tipo | Uso Principal | Plataforma | Paradigma | Ventajas | Desventajas |
|-----------------------|-----------|--------------|--------------------------------------|---------------------------|----------------|-----------------------------------|------------------------|
| Python | General | Interpretado | Ciencia de datos, ML, desarrollo web | Web, ciencia de datos, ML | Multiparadigma | Fácil de aprender, gran comunidad | Velocidad de ejecución |
| JavaScript | | | | | | | |
| Java | | | | | | | |
| C# | | | | | | | |
| C | | | | | | | |
| C++ | | | | | | | |
| PHP | | | | | | | |
| GO | | | | | | | |
| Rust | | | | | | | |
| Swift | | | | | | | |
| Ruby | | | | | | | |
| Bash/Batch/PowerShell | | | | | | | |

Programación Estructurada

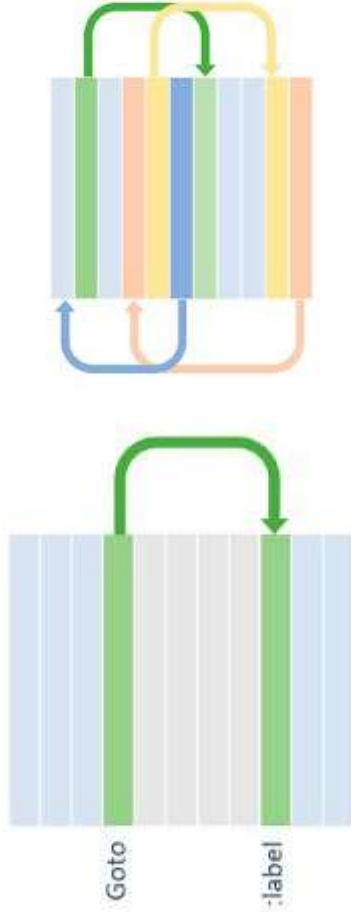
- Secuencial
- Alternativa
- Iterativa



Ideas capacitación
Programación estructurada.

Programación Estructurada

Sentencia GOTO



```
goto etiqueta;  
...  
...  
...  
etiqueta: sentencia;
```

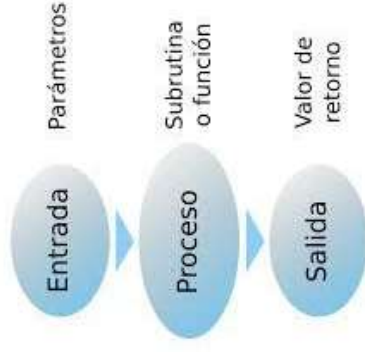
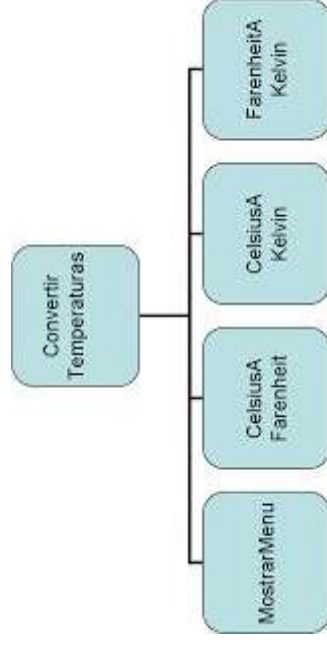
Código espagueti



Java Hispano

Programación Modular

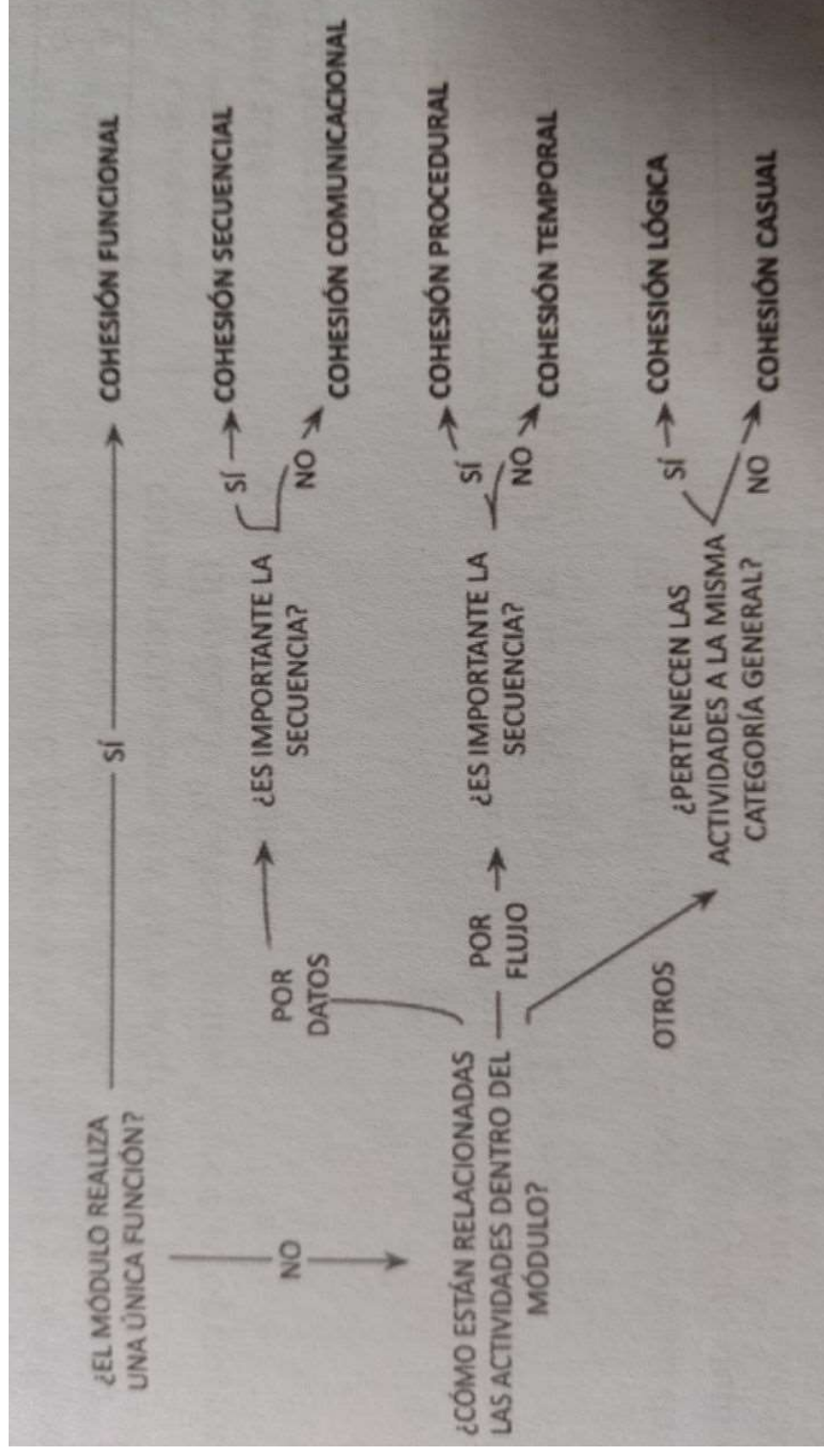
- Los programas crecen y se hacen más complejos \Rightarrow dividirlos en módulos
- Módulo = subprograma o rutina \Rightarrow funciones
- Librerías



¿Cómo dividir en módulos un programa?

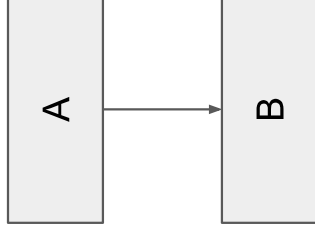
- Único punto de entrada y único punto de salida
- Función bien definida
- Caja negra
- Pequeños
- Máxima cohesión
- Mínimo acoplamiento

Cohesión

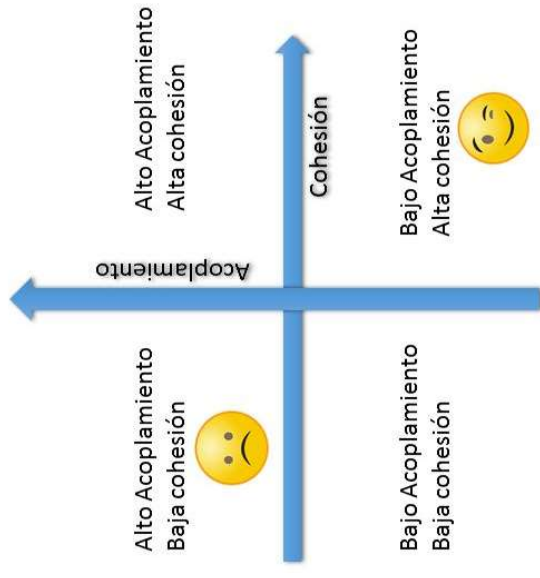
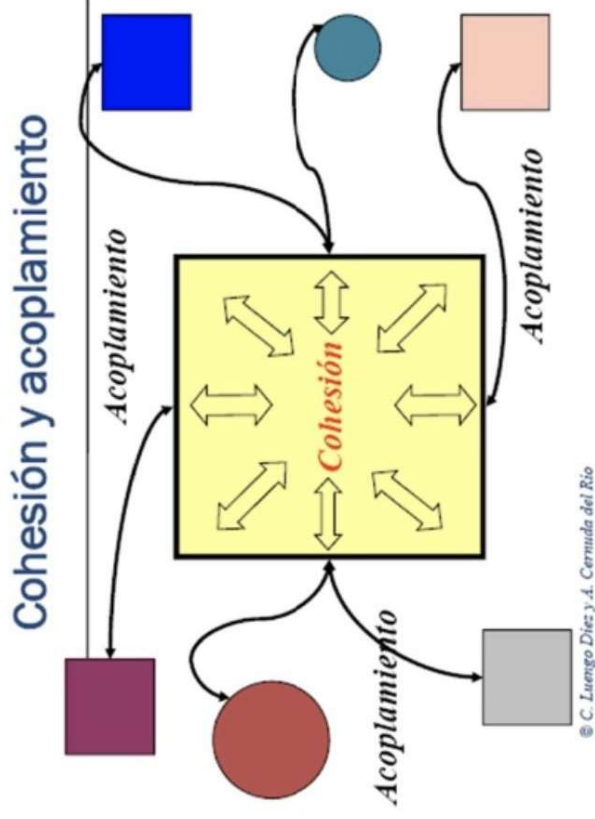


Acoplamiento

Si para hacer cambios en un módulo del programa es necesario hacer cambios en otro módulo distinto, existe acoplamiento entre ambos módulos.



Cohesión y Acoplamiento

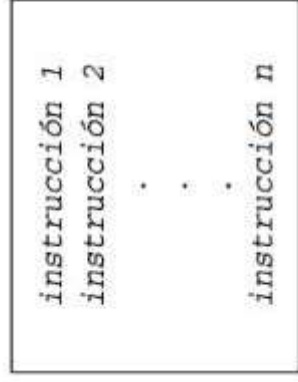


Elementos de un programa

1. **Entrada**
 - **Datos:** La información que el programa recibe del usuario o de otros dispositivos para realizar sus cálculos o tomar decisiones.
2. Procesamiento
3. Salida
4. Palabras reservadas
 - **Dispositivos de entrada:** Teclado, mouse, escáner, micrófonos, etc.



Elementos de un programa

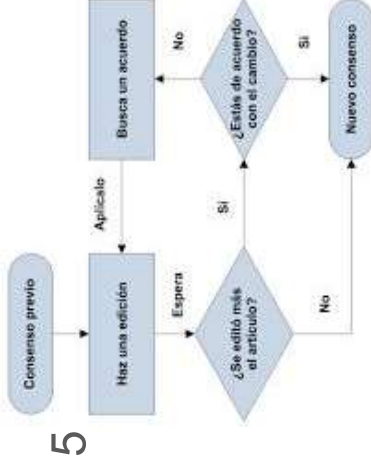


- **Instrucciones:** Órdenes. Unidad mínima.
- **Variables:** Espacios en la memoria del ordenador donde se almacenan los datos que el programa está utilizando.
- **Constantes:** como una variable cuyo valor inicial no cambia a lo largo del programa
- **Literales:** datos “*a pelo*” en el código.

| Memoria | Variable |
|---------|----------|
| 10 | valor1 |
| 1000 | pe |

Elementos de un programa

1. Entrada
2. **Procesamiento**
3. Salida
4. Palabras reservadas



- **Operadores:** Símbolos que permiten realizar operaciones matemáticas, lógicas o de comparación sobre los datos.
- **Estructuras de control:** Mecanismos que permiten controlar el flujo de ejecución del programa, como las condicionales (si, entonces, sino) y los bucles (para, mientras).

Elementos de un programa

1. Entrada
2. **Procesamiento**
 - **Expresiones:** Son combinaciones de literales, constantes, variables y operadores para ejecutar una operación.
 $7 + 3$ $\text{edad} < 12$ $\text{PI} * r^2$
 - **Asignación:** Operación que toma el valor de una expresión y lo almacena en una variable
 $\text{nombre} = \text{"Fran"}$ $\text{suma} = 2 + 3$
3. Salida
4. Palabras reservadas
5. Comentarios

Elementos de un programa

1. Entrada
 2. Procesamiento
 3. Salida
 4. Palabras reservadas
- **Resultados:** La información que el programa genera a partir del procesamiento de los datos de entrada.
 - **Dispositivos de salida:** Pantalla, impresora, altavoces, etc.



Elementos de un programa

1. Entrada Comandos que entiende el programa

2. Procesamiento

3. Salida

4. **Palabras
reservadas**

5. Comentarios

Algoritmo
FinAlgoritmo
Escribe
Lee
...

Elementos de un programa

1. Entrada
2. Procesamiento
3. Salida
4. Palabras reservadas
5. **Comentarios**
 - Textos que añaden claridad al código, explicando qué hace cada parte del programa. Son muy útiles para documentar el código y facilitar su comprensión por parte de otros programadores o por uno mismo en el futuro.

Ejemplo

Programa que calcula el área de un círculo.

Entrada: El radio del círculo (un número ingresado por el usuario).

Procesamiento: Una variable llamada "radio" para almacenar el valor ingresado. Una constante llamada "pi" con el valor 3.14159. Una sentencia que multiplica el radio por sí mismo y luego por pi para calcular el área.

Salida: El resultado del cálculo (el área del círculo), mostrado en pantalla.

Comentarios: Explicaciones sobre cómo se realiza el cálculo y el significado de cada variable.

Palabras reservadas

Son las palabras que forman parte del lenguaje



Java keywords

| short | if | implements | finally | throw |
|-----------|-----------|------------|------------|--------------|
| boolean | void | int | long | while |
| case | do | switch | private | interface |
| abstract | default | byte | else | try |
| for | double | class | catch | extends |
| final | transient | float | instanceof | package |
| continue | native | public | break | char |
| protected | return | static | super | synchronized |
| this | new | throws | import | volatile |

No podemos usarlas para nombres de variables, algoritmos, etc.

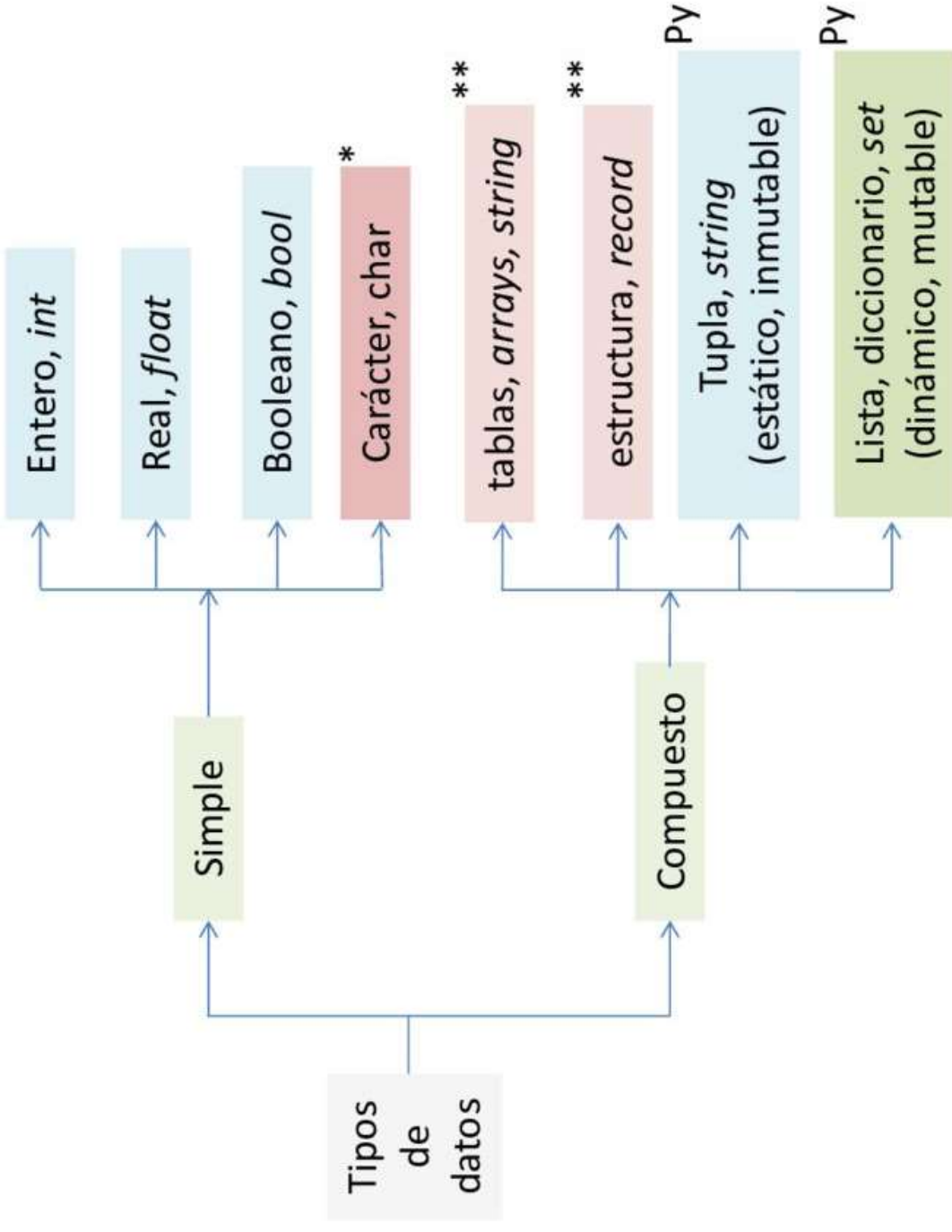
Operadores

Aritméticos, relacionales, lógicos y especiales

Precedencia de operadores

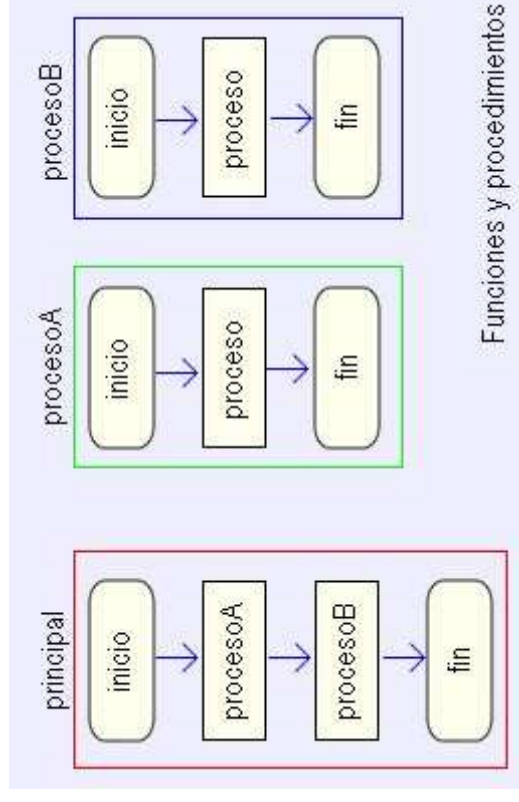
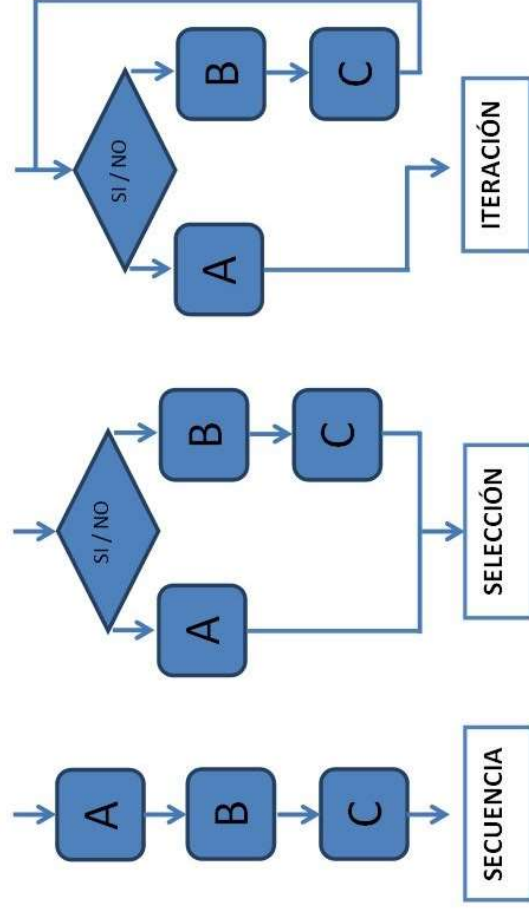
| Descripción | Operadores |
|---------------------------|--------------|
| Postfijos | i++, i-- |
| Unarios | ++, --, ~, ! |
| Multiplicación y división | *, /, % |
| Suma y resta | +, - |
| Relacionales | >, <, >=, <= |
| Equivalencia | ==, != |
| AND lógico | && |
| OR lógico | |
| Asignación | = |

http://es.wikibooks.org/wiki/Programación_en_Java/Precedencia_de_operadores

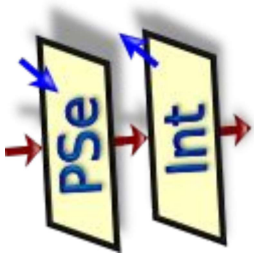


- Secuencial
- Selectiva o Alternativa
- Iterativa o Repetitiva
- Modular: Funciones y Procedimientos

Estructuras



Pseudocódigo



Crear un algoritmo sin usar un lenguaje de programación concreto.

- Lenguaje informal → Cada uno con sus palabras
- Ayuda a ver la lógica antes de pasar a codificar en un lenguaje real
- Facilita comunicación entre programadores
- Sirve para documentar el código
- Útil para aprender a programar

```
1  Algoritmo calcular_area_circulo
2  // Declaración de variables
3  Definir radio, area Como Real
4
5  // Entrada de datos
6  Escribir "Ingrese el radio del círculo: "
7  Leer radio
8
9  // Cálculo del área
10 area ← pi * radio * radio
11
12 // Salida de resultados
13 Escribir "El área del círculo es: ", area
14
15 FinAlgoritmo
```

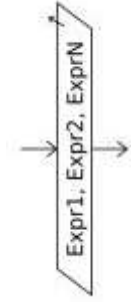
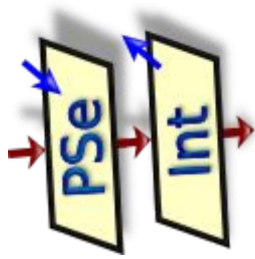

Pseudocódigo

Ejercicios:

Instalamos PSeInt y empezamos a escribir nuestro pseudocódigo.

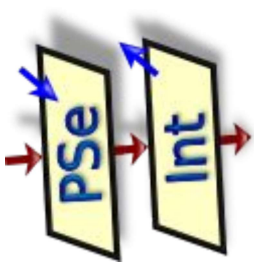
1. **Hola mundo:** Escribe un programa que muestre por pantalla “Hola mundo”
2. **Hola usuario:** Escribe un programa que muestre por pantalla “Hola “ y el nombre introducido por teclado.

Ejemplo: si introducimos “Pepito” mostraría “*Hola Pepito*”. ¿Cómo harías para mostrar una admiración al final? Ej: “Hola Pepito!”

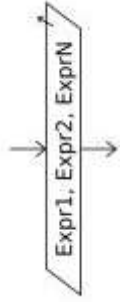


Estructuras de
control **secuencial**

Pseudocódigo



2. **Hola usuario:** Escribe un programa que muestre por pantalla “Hola “ y el nombre introducido por teclado. Ejemplo: si introducimos “Pepito” mostraría “Hola Pepito”.
¿Cómo harías para mostrar una admiración al final? Ej: “Hola Pepito!”



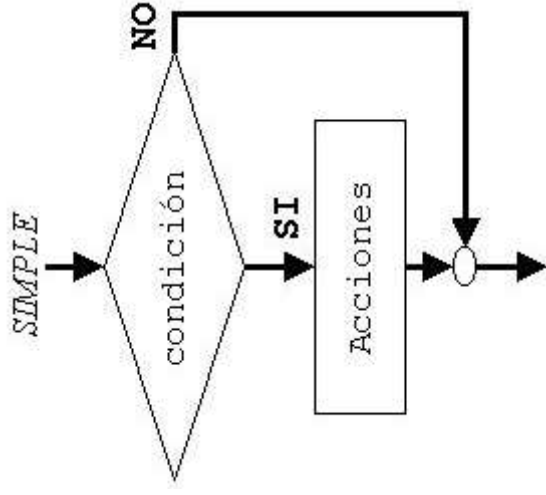
Estructuras de
control **secuencial**

3. **suma 2 números:** Escribe un programa que sume dos números enteros introducidos por teclado y muestre el resultado por pantalla.
Utiliza variables para cada número.

Pseudocódigo

4. **Login:** Escribe un programa que pida el nombre de usuario y si es correcto muestre “¡Bienvenido <usuario>!”.

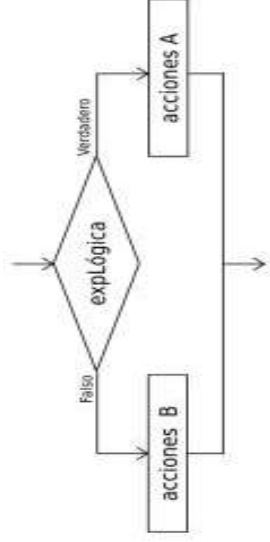
Un usuario será correcto si coincide con el valor establecido literalmente por el programador en el código.



Estructuras de control **alternativas simples**

Pseudocódigo

5. Ahora añade que si el usuario no es correcto, entonces muestre “Usuario incorrecto”
6. **Mayor o menor:** Solicitar al usuario dos números y mostrar cuál es el mayor.
¿Cómo harías para evaluar además si son iguales?
7. **Par o impar:** Pedir un número al usuario y determinar si es par o impar.
8. **Calculadora básica:** Crear un programa que realice las cuatro operaciones básicas (suma, resta, multiplicación y división) según la opción que seleccione el usuario.



Sentencias de
control **alternativas**
múltiples

Pseudocódigo

7. **Año bisiesto:** Determinar si un año ingresado por el usuario es bisiesto.

Un año es bisiesto si cumple **a la vez** de estas dos condiciones:

- a. Es divisible por 4
- b. No es divisible por 100 o sí lo es por 400

Ejemplos:

- 2024 es bisiesto → porque es divisible entre 4
- 1900 no bisiesto → divisible entre 4 pero también entre 100
- 2000 bisiesto → divisible entre 4, también por 100 pero también por 400



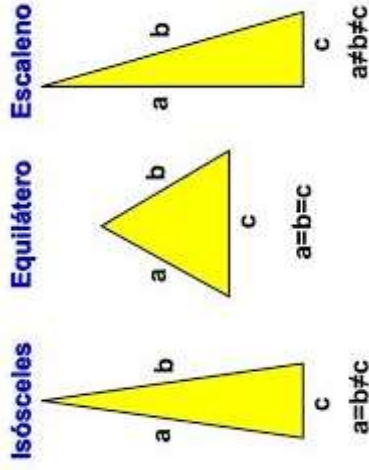
Pseudocódigo

Sentencias de control
alternativas
múltiples

8. **Calificación:** Solicitar una nota numérica y mostrar la calificación correspondiente (Sobresaliente, Notable, Bien, Suficiente, Insuficiente) según una escala determinada.

| | |
|------|---------------|
| 0-4 | Insuficiente |
| 5 | Suficiente |
| 6 | Bien |
| 7-8 | Notable |
| 9-10 | Sobresaliente |

Pseudocódigo



9. **Triángulo:** Dado tres lados, determinar si pueden formar un triángulo **y, en caso afirmativo, indicar si es equilátero, isósceles o escaleno.**

Tres lados forman un triángulo si la suma de las longitudes de dos cualesquiera es mayor que la longitud del tercero

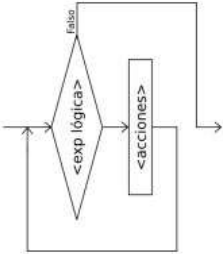
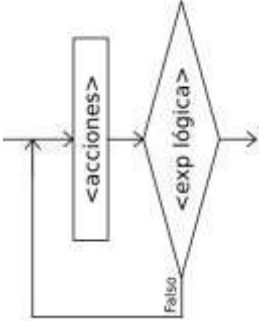
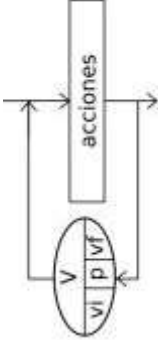
Equilátero: todos los lados iguales

Isósceles: dos lados iguales

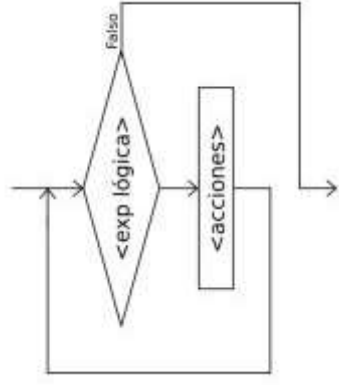
Escaleno: ningún lado igual

Sentencias de control
alternativas múltiples

Iteraciones o bucles

| | | | |
|----------|---------------------|---------------------------------------|---|
| while | Sale al principio | Mientras |  |
| do while | Sale al final | Hacer mientras / Repetir hasta que |  |
| for | Contador automático | Para |  |
| for each | Conjuntos | Para cada | |

Pseudocódigo



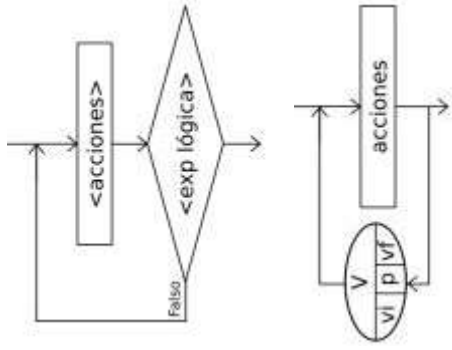
Sentencias de
control **iterativa**

10. Contar hasta un número:

Escribe un programa que pida al usuario un número entero positivo y cuente desde 1 hasta ese número.

Ejemplo: si introduzco el 3, el programa muestra: 1, 2, 3.

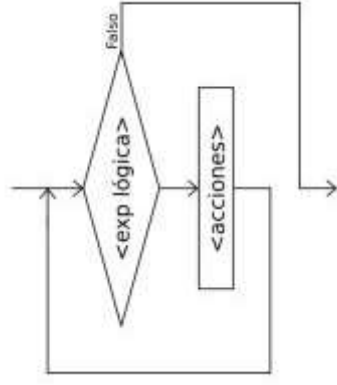
Pseudocódigo



11. Contar hasta un número usando “Hacer mientras”
12. Contar hasta un número usando “Repetir hasta”
13. Contar hasta un número usando “Para”

Sentencias de control **iterativa**

Pseudocódigo



Sentencias de
control **iterativa**

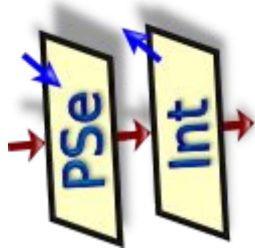
14. Suma hasta pulsar tecla cero:

Suma de números hasta que el usuario ingrese 0.

Cada vez que el usuario introduzca un número lo sumo a lo que ya tenía y le pregunto otra vez.

Este tipo de algoritmos se llaman acumuladores

Pseudocódigo



15. **Tabla de multiplicar:** Imprimir la tabla de multiplicar de un número.
16. **Suma de números:** Calcular la suma de los números del 1 al 100.
17. **Suma inversa:** realiza la suma desde el 100 al 1
18. **Suma pares:** realiza la suma sólo de los números pares del 1 al 100
19. **Factorial:** Calcular el factorial de un número introducido por el usuario
- 20.

Sentencias de
control **iterativa**