



# 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](https://creativecommons.org/licenses/by-nc-sa/4.0/).

Autor: Fran Gómez



# Objetivos

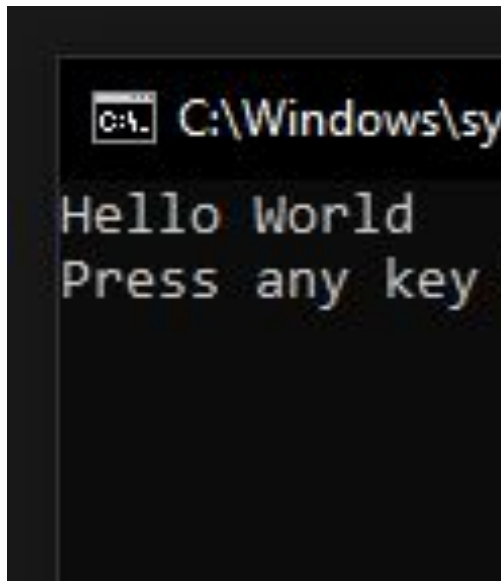
**RA1:** Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

- 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

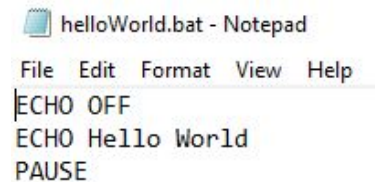
# Introducción



```
C:\Windows\system32>
Hello World
Press any key
```



```
World.bat
.
```



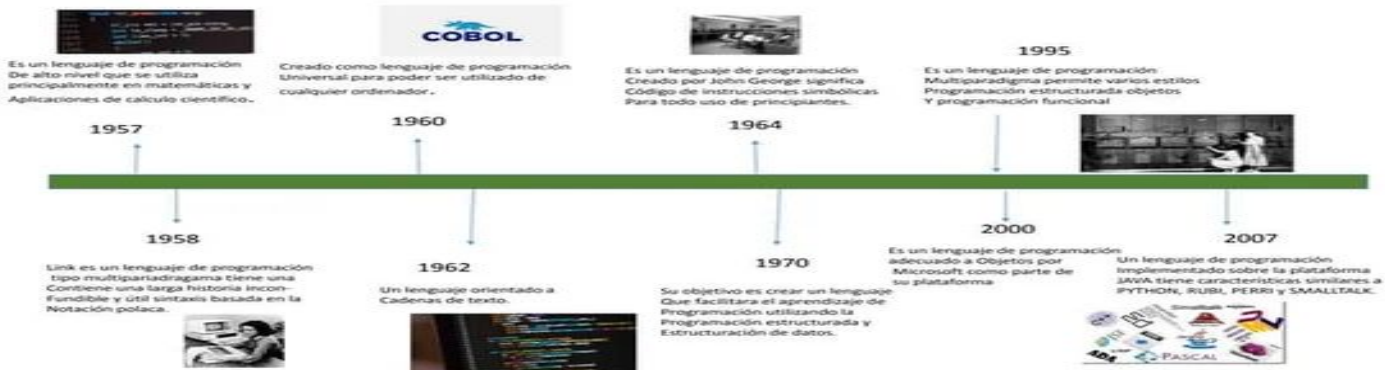
```
helloWorld.bat - Notepad
File Edit Format View Help
ECHO OFF
ECHO Hello World
PAUSE
```

# 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

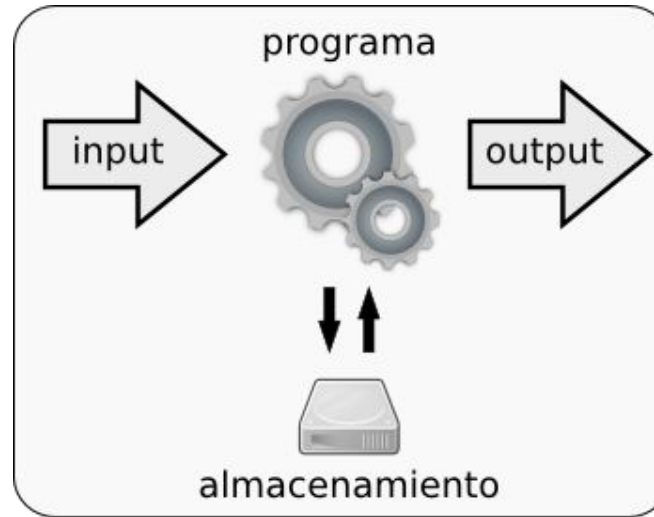


# Conceptos de programación

- 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)
  - Compilado
  - Marcado
- Uso:
  - Desarrollo Web
  - Juegos
  - Sistemas
  - Móvil
  - IA, BigData, Machine Learning...
- Plataforma:
  - Web
    - Frontend
    - Backend
  - Desktop
  - Móvil
  - Embeded
- Paradigma:

# Paradigmas de programación

## Ejercicio

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

- **Imperativa**  $\Rightarrow$  Instrucciones paso a paso
- **Estructurada** o procedural  $\Rightarrow$  Instrucciones siguiendo estructuras
- **Declarativa**  $\Rightarrow$  Resultado deseado
- **Modular**  $\Rightarrow$  Divide el programa en partes
- **Orientada a objetos**  $\Rightarrow$  Agrupa Datos y Lógica

Otros:

- Funcional  $\Rightarrow$  funciones matemáticas
- Lógico  $\Rightarrow$  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 sándwich

```
#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 sándwich\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 sándwich va cambiando a medida que se ejecutan las instrucciones.

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

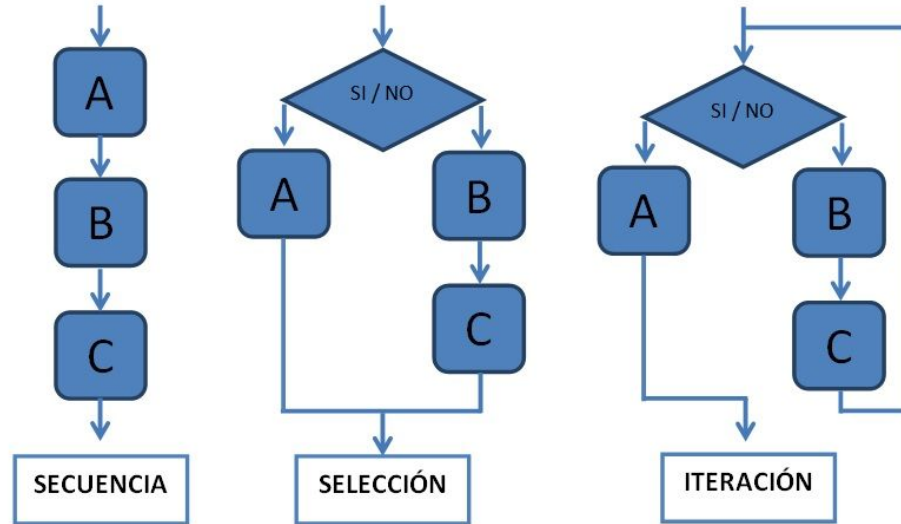
## Ejercicio 2

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

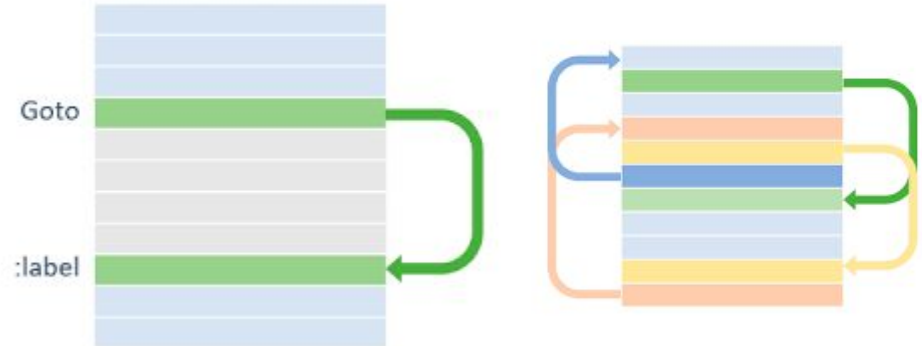


Ideas capacitación  
Programación estructurada.

# Programación Estructurada

Sentencia GOTO

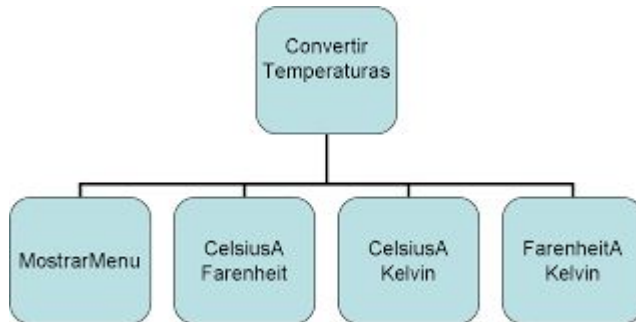
Código espagueti



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

# Programación Modular

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



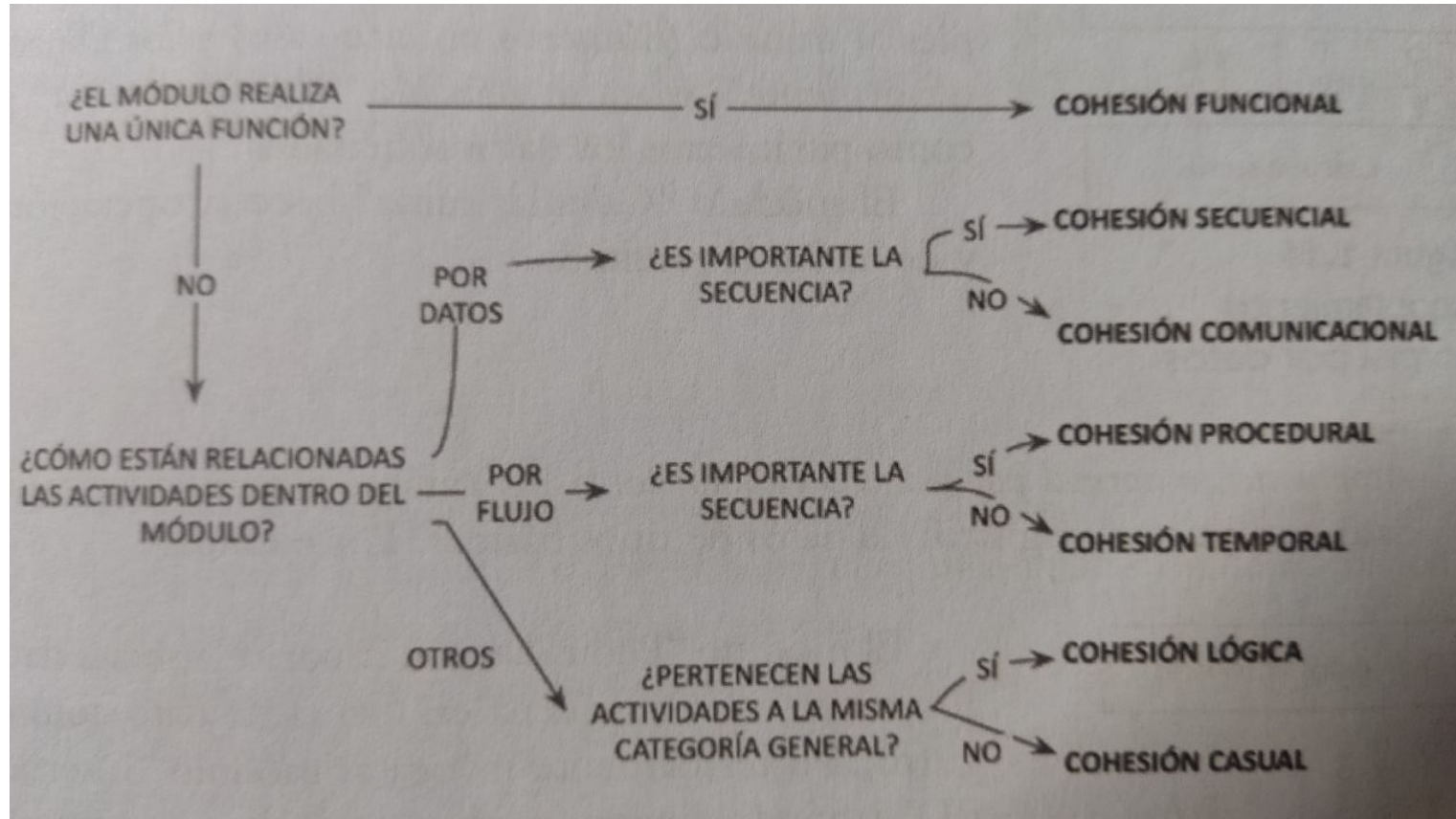


# ¿Cómo dividir en módulos un programa?

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

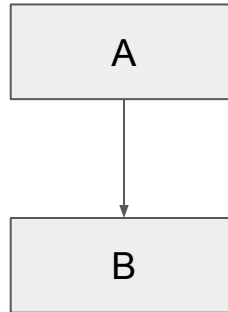


# Cohesión

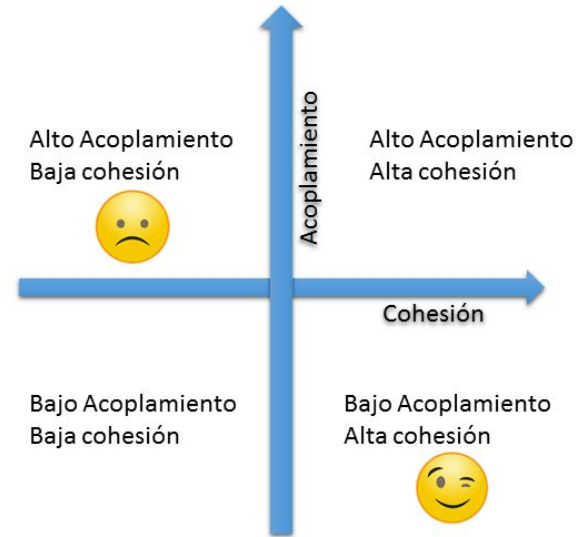
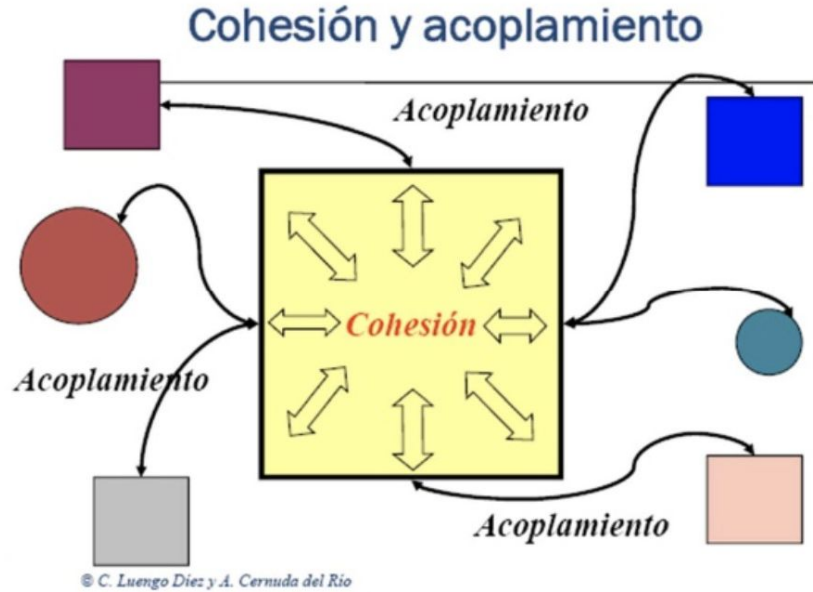


# Acoplamiento

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



# Elementos de un programa

```
instrucción 1
instrucción 2
.
.
.
instrucción n
```

Memoria      Variable

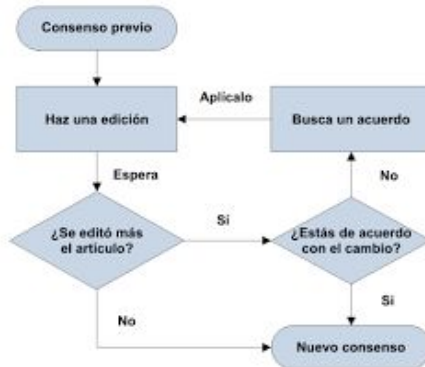
10	valor1
1000	pe

- **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.

# Elementos de un programa

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

5



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

# 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. <b>Palabras reservadas</b> | <i>Algoritmo</i><br><i>FinAlgoritmo</i><br><i>Escribe</i><br><i>Lee</i><br><i>...</i> |
| 5. Comentarios                |   |

# 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

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

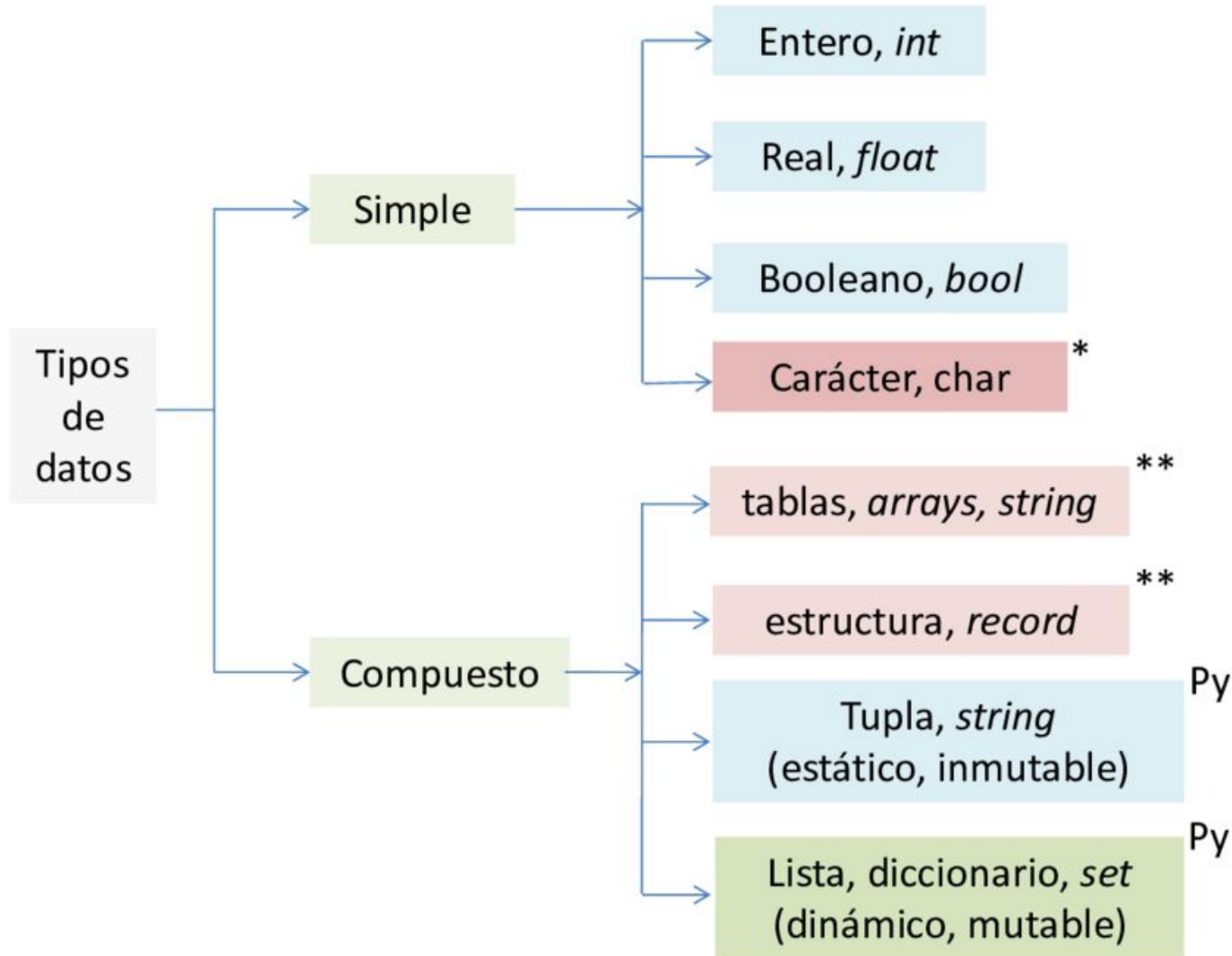
No podemos usarlas para nombres de variables, algoritmos, etc.

# Operadores

Aritméticos, relacionales, lógicos y especiales

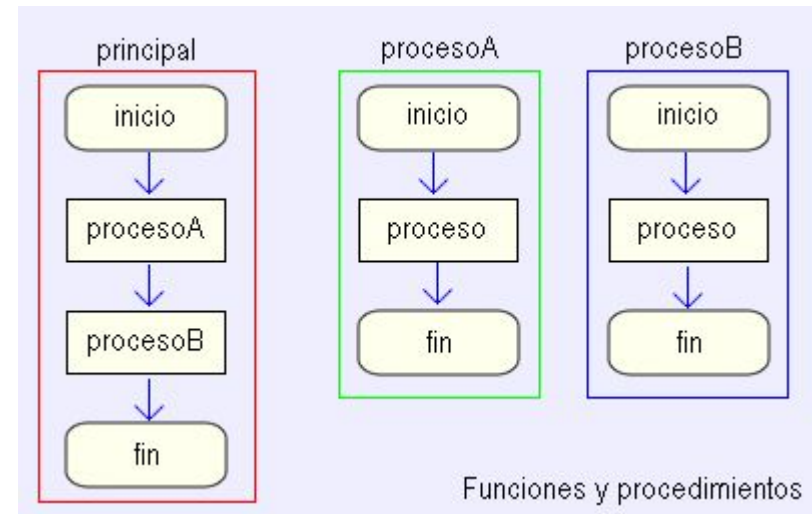
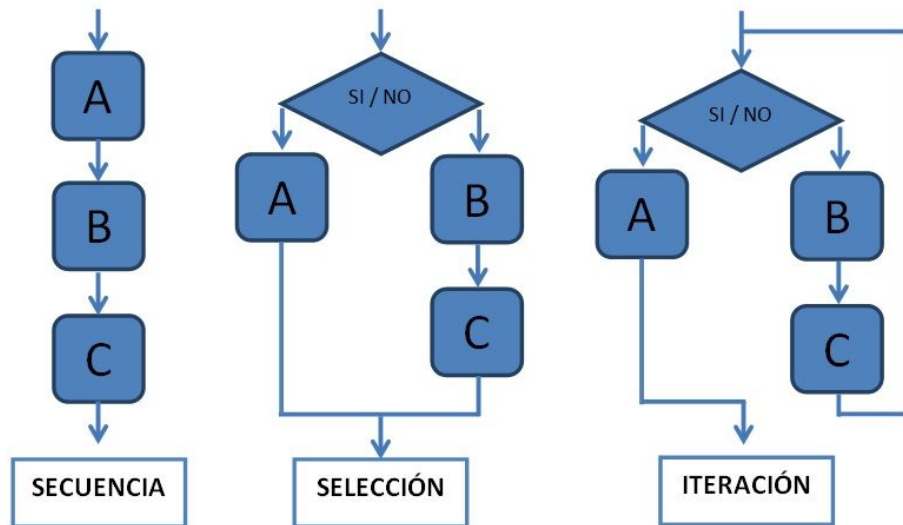
## Precedencia de operadores

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

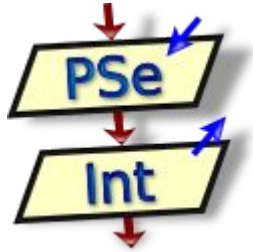


- Secuencial
- Selectiva o Alternativa
- Iterativa o Repetitiva
- Modular: Funciones y Procedimientos

## Estructuras



# Pseudocódigo



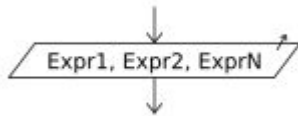
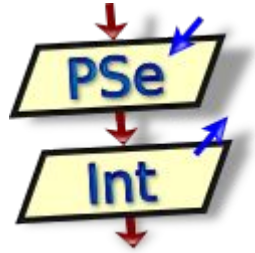
Crear un algoritmo sin usar un lenguaje de programación concreto.

- 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



Estructuras de  
control **secuencial**

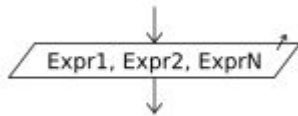
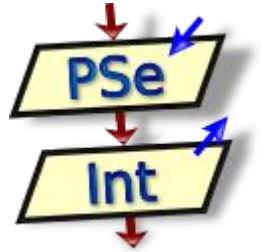
## Ejercicios:

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.

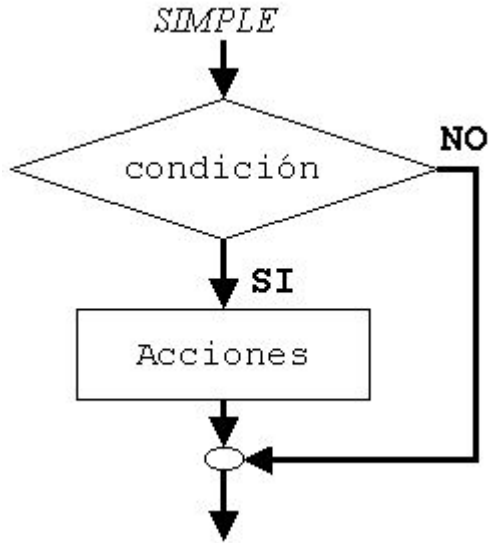
Ejemplo: si introducimos “Pepito” mostraría “*Hola Pepito*”. ¿Cómo harías para mostrar una admiración al final? Ej: “Hola Pepito!”

# Pseudocódigo



Estructuras de  
control **secuencial**

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!”
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.



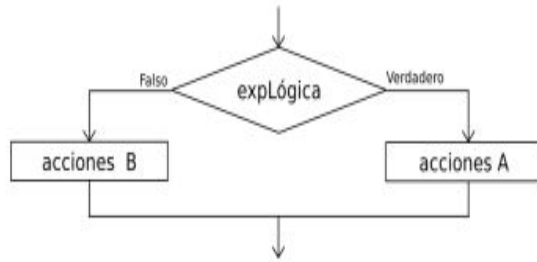
Estructuras de control **alternativas simples**

## 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.

# Pseudocódigo



Sentencias de control **alternativas múltiples**

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.



# 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:

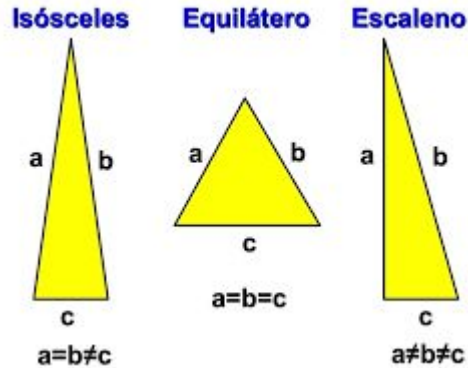
- 2024 es bisiesto → porque es divisible entre 4
- 1900 no bisiesto → divisible entre 4 pero también entre 100
- 2000 bisiesto → divisible entre 4, también por 100 pero también por 400

# 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



Sentencias de  
control  
**alternativas  
múltiples**

## Pseudocódigo

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

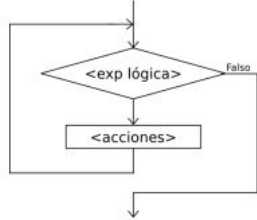
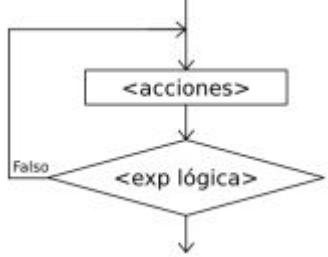
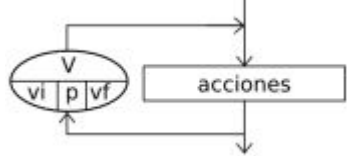
Tres lados forman un triángulo si la suma de las longitudes de dos cualesquiera es mayor que la longitud del tercero

**Equilátero:** todos los lados iguales

**Isósceles:** dos lados iguales

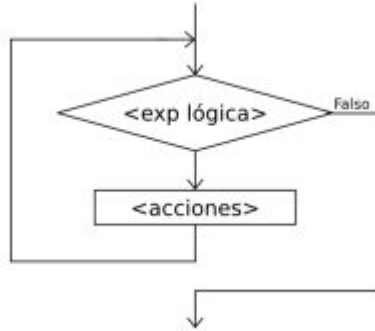
**Escaleno:** ningún lado igual

# 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



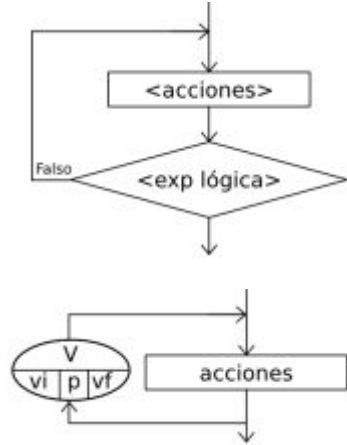
Sentencias de control **iterativa**

## 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.

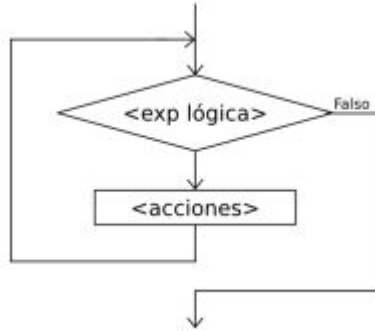
# Pseudocódigo



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**

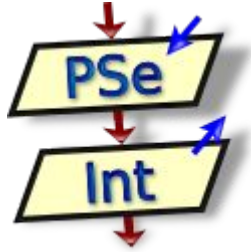
## 14. Suma hasta pulsar tecla cero:

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



Sentencias de  
control **iterativa**

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

# Módulos

Dividir las funcionalidades del programa en subprogramas:

- Procedimientos
- Funciones

Parámetros y argumentos



# Procedimiento

- Escribimos cada subprograma con la siguiente sintaxis:

```
Procedimiento nombre (tipo parámetro1, ... tipo parámetroN)  
    <instrucciones>  
Fin Procedimiento
```

- Invocamos el procedimiento así:

```
nombre (parámetro1, . . . parámetroN)
```

- No retorna ningún valor

# Función

- Escribimos cada subprograma con la siguiente sintaxis:

```
Función nombre (tipo parámetro1, ... tipo parámetroN) : tipoRetorno  
    <instrucciones>  
    retornar x  
Fin Función
```

- Invocamos el procedimiento así:

```
nombre (parámetro1, . . . parámetroN)
```

Puede usarse en una expresión

- Retorna valor de salida

## Módulos: Ejercicios

20. Crear un programa que invoque a una función que devuelva si el número leído en el programa principal es par o impar.
21. Adaptar el ejercicio anterior para hacerlo mediante un procedimiento en lugar de una función



# Funciones predefinidas

También existen funciones ya definidas en el lenguaje

Función	Significado
RC(X) o RAIZ(X)	Raíz Cuadrada de X
ABS(X)	Valor Absoluto de X
LN(X)	Logaritmo Natural de X
EXP(X)	Función Exponencial de X
SEN(X)	Seno de X
COS(X)	Coseno de X
TAN(X)	Tangente de X
ASEN(X)	Arcoseno de X
ACOS(X)	Arcocoseno de X
ATAN(X)	Arcotangente de X
TRUNC(X)	Parte entera de X
REDON(X)	Entero más cercano a X
AZAR(X)	Entero aleatorio en el rango [0;x-1]
ALEATORIO(A,B)	Entero aleatorio en el rango [A;B]
LONGITUD(S)	Cantidad de caracteres de la cadena S
MAYUSCULAS(S)	Retorna una copia de la cadena S con todos sus caracteres en mayúsculas
MINUSCULAS(S)	Retorna una copia de la cadena S con todos sus caracteres en minúsculas
SUBCADENA(S,X,Y)	Retorna una nueva cadena que consiste en la parte de la cadena S que comienza en la posición X y termina en la posición Y.
CONCATENAR(S1,S2)	Retorna una nueva cadena resultante de unir las cadenas S1 y S2.
CONVERTIRNUMERO(X)	Recibe una cadena de caracteres que contiene un número y devuelve su valor numérico.
CONVERTIRTEXTO(S)	Recibe un real y devuelve una variable numérica con la representación en texto.

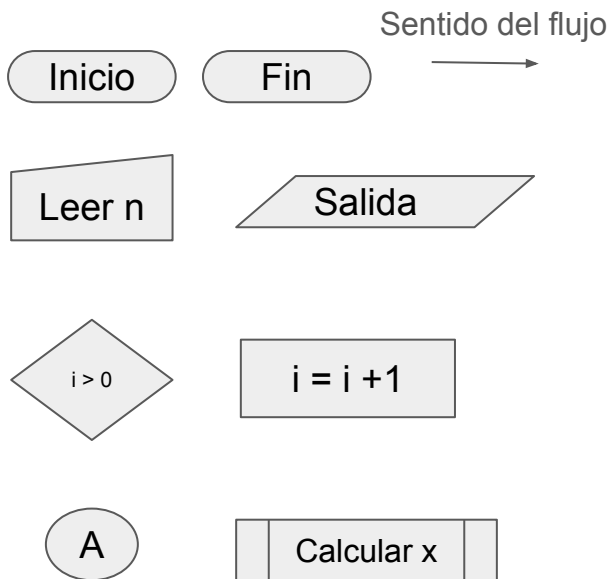
Funciones  
predefinidas en  
PseInt

## Ejercicios

22. Amplia el ejercicio de la calculadora para incluir la operación de la raíz cuadrada. Utiliza las funciones predefinidas.
23. Escribe un programa que muestre por pantalla “Hola “ y el nombre introducido por teclado, pero usando la función CONCATENA
24. Crea tu propia función para concatenar pero que añada además un espacio entre las dos palabras. Llama a esa función concatenaConEspacios

# Diagramas de flujo

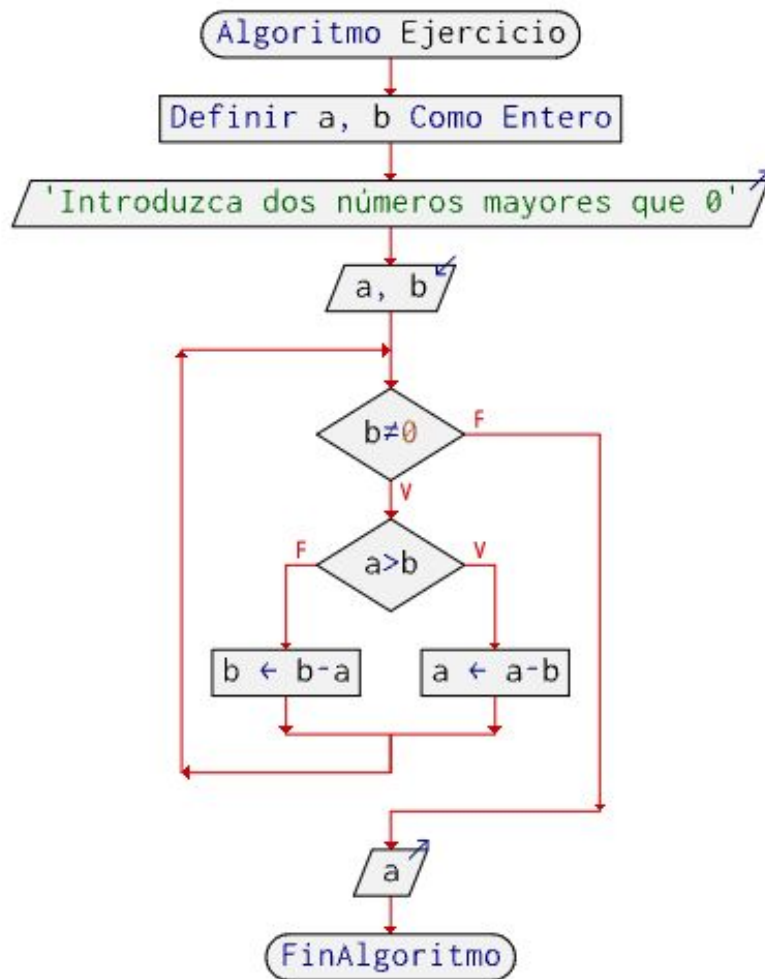
Representación gráfica de un programa



# Ejercicios



25. Escribe un pseudocódigo para este diagrama. ¿Qué hace el programa?



# Ejercicios

26. Escribe un flujograma para este pseudocódigo. ¿Qué hace el programa?

```
1  Algoritmo Ejercicio
2      Escribir "Ingrese el numero: "
3      Leer N
4      Escribir "Ingrese el divisor: "
5      Leer M
6      Si  $N \text{ MOD } M = 0$  Entonces
7          Escribir M, " es divisor exacto de ",N, "."
8      SiNo
9          Escribir "El resto de dividir ",N, " por ",M, " es: ",  $N \text{ MOD } M$ 
10     FinSi
11 FinAlgoritmo
```

## Ejercicios: Diagrama y Pseudocódigo

27. **Números primos:** Determinar si un número introducido por el usuario es primo. (Divisible sólo por sí mismo y la unidad)
28. **Promedio:** Pedir al usuario que introduzca varios números hasta pulsar la tecla espacio. Después muestra el promedio de todos los números introducidos.
29. Mejora el ejercicio 4 y 5 para que después de tener un usuario correcto pida la contraseña. Además, mientras la contraseña sea incorrecta, la siga solicitando.
30. Mejora el ejercicio de la calculadora para que no acabe nunca hasta que pulse una tecla determinada para salir



## Ejercicios: Diagrama y Pseudocódigo

**31.** Crea un programa para seleccionar un alumno de la clase de manera aleatoria. Debe contar con un menú con un listado con todos los alumnos de la clase y un número para cada uno. Además de indicar al usuario la tecla que tiene que pulsar para seleccionar un alumno al azar.



# Recursos y referencias

- Libro que tenéis en la biblioteca y en el departamento
- Diapositivas en Moodle
- Glosario en Moodle
- Chuleta en Moodle
- Ejercicios de refuerzo y de ampliación
- PseInt
- Vuestro portfolio