

TEMA 0. DESARROLLO SOFTWARE

CONTENIDOS

1.1. Objetivos

1.2. Algoritmos

1.3. Diagramas de flujo

1.4. Lenguajes de programación

1.5. Pseudocódigo

1.6. Webs para explorar.

1.7. Referencias.

OBJETIVOS

- APRENDER EL CONCEPTO DE ALGORITMO
- CONOCER LOS FUNDAMENTOS DE LOS LENGUAJES DE PROGRAMACIÓN
- ADQUISICIÓN Y ENTRENAMIENTO DEL PENSAMIENTO COMPUTACIONAL
- MODELAR ALGORITMOS SENCILLOS CON DIAGRAMAS DE FLUJO
- ESCRIBIR ALGORITMOS SENCILLOS CON PSEUDOCÓDIGO

ALGORITMO

- **Secuencia de pasos finito para resolver un problema.**
- Cuando el problema crece, imposible resolverlo con un algoritmo, siendo necesario descomponer el problema “***divides y vencerás***”

Definición: Secuencia precisa, finita y ordenada de pasos lógicos necesarios para resolver un problema. Es independiente del lenguaje de programación.

ALGORITMO

Un algoritmo es similar a una receta...cada paso es una instrucción, una orden

Inicio

¿Tengo hambre?

Si:

Enciende el fuego;

Pon la sartén al fuego;

Pon aceite en la sartén;

Mientras el aceite esté frío, hacer:
esperar;

Echar el huevo a la sartén.

Mientras el huevo esté crudo, hacer:
freir;

Servir en el plato;

Apagar el fuego;

No:

Salir de la cocina

Fin



ALGORITMO

CARACTERÍSTICAS

Conjunto ordenado instrucciones precisas que llevan a resolver un determinado problema. Ha de cumplir con las siguientes condiciones:

- Ser **correcto**: Resuelve el problema
- Ser **finito**: Conduce a la solución en un tiempo dado
- Ser **flexible**: No es exclusivo para un valor concreto de datos de entrada, sino que sirve como método general para distintos datos
- Ser **claro**: comprensible por otras personas
- Ser **eficiente**: ahorro de tiempo y recursos
- Ser **portable**: independiente de la máquina o del lenguaje utilizado

ALGORITMOS RESOLVER PROBLEMAS

El algoritmo **surge** de la **necesidad** de **resolver un problema dado**. Para ello, siempre será posible crear múltiples soluciones. El algoritmo que seleccionemos será aquel que obtenga los resultados esperados en el menor tiempo posible y con el menor coste. Para ello debemos reflexionar sobre:

- ¿Qué información o resultados se quieren obtener?
- ¿A través de qué procesos se podrán obtener los resultados?
- ¿Se requiere algún proceso intermedio?
- ¿Qué tipo de datos serán necesarios?
- ¿Qué variables?
- ¿Qué condiciones exige el problema para su solución?

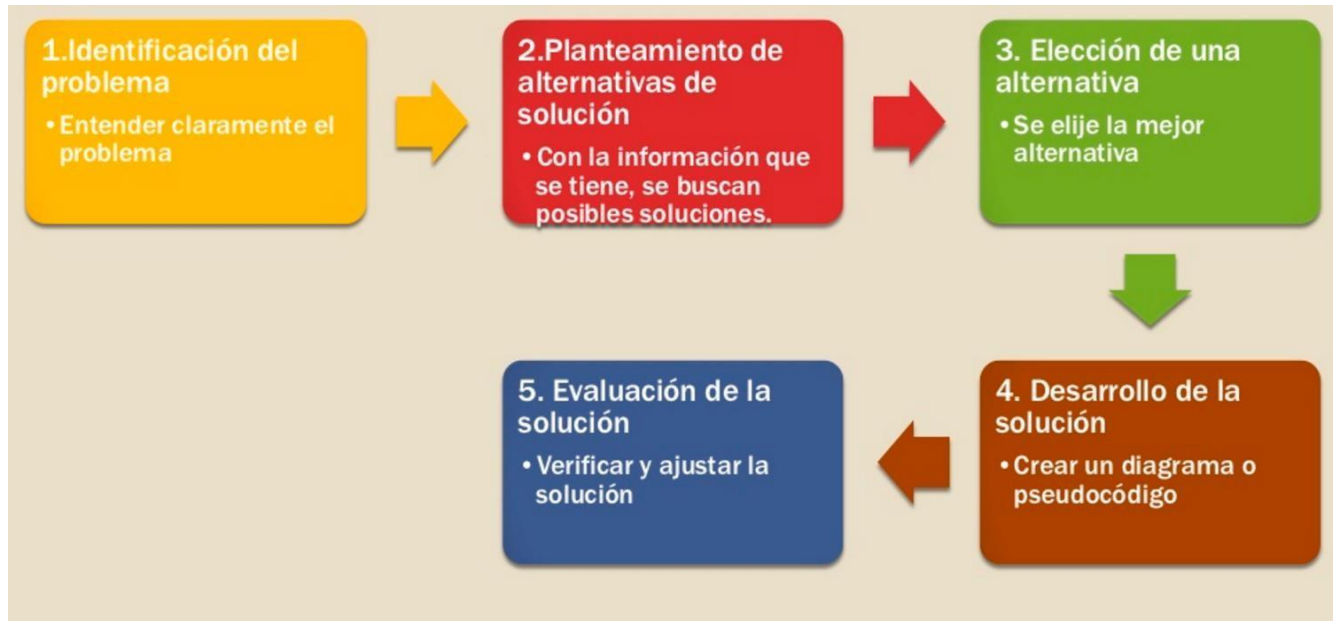
ALGORITMOS. REPRESENTACIÓN

Una vez que se ha elegido la mejor alternativa para solucionar el problema o reto para el que se crea el algoritmo es el momento de representarlo siguiendo alguno de estos métodos:

- LENGUAJE NATURAL (ESPAÑOL, INGLÉS, ETC.)
- DIAGRAMAS DE FLUJO
- PSEUDOCÓDIGO

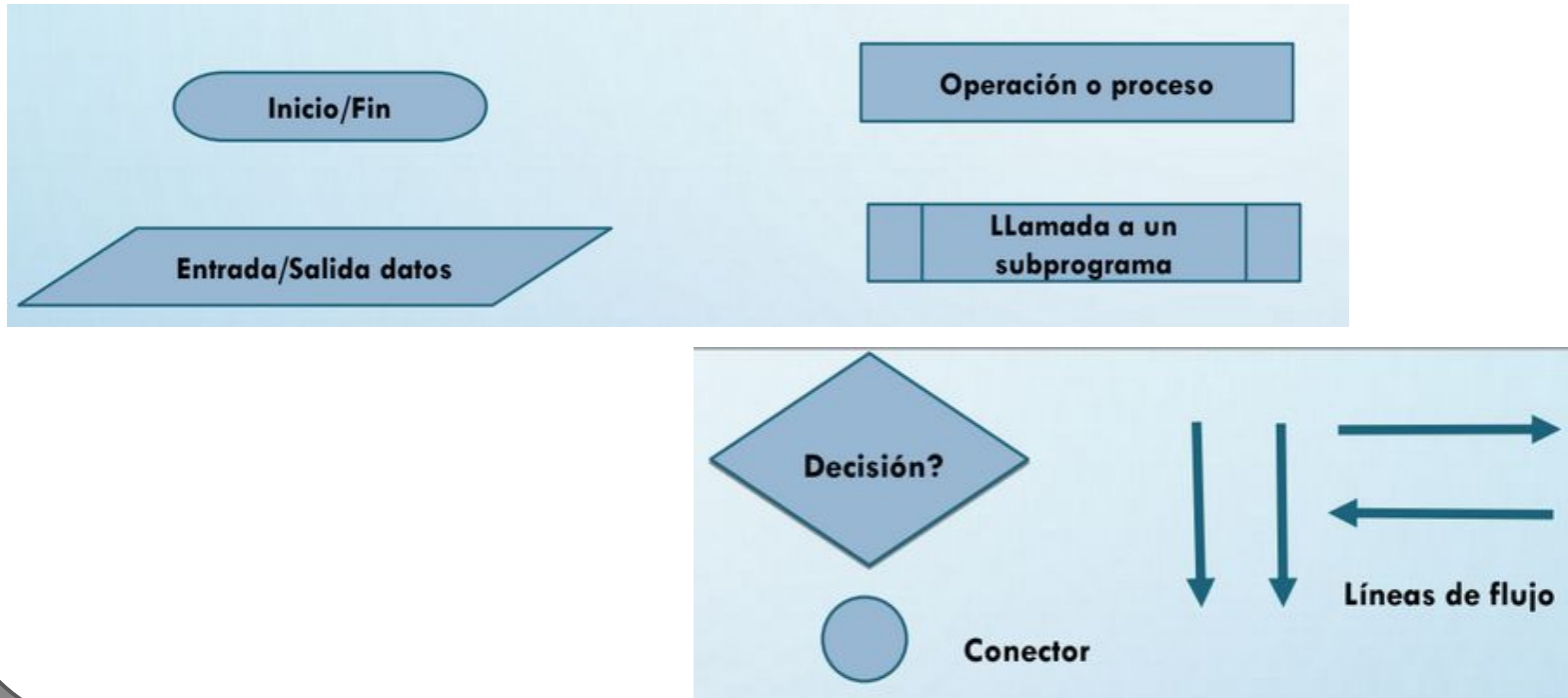
ALGORITMOS. PLANIFICACIÓN. DISEÑO. DESARROLLO

Método apropiado para desarrollar un algoritmo:

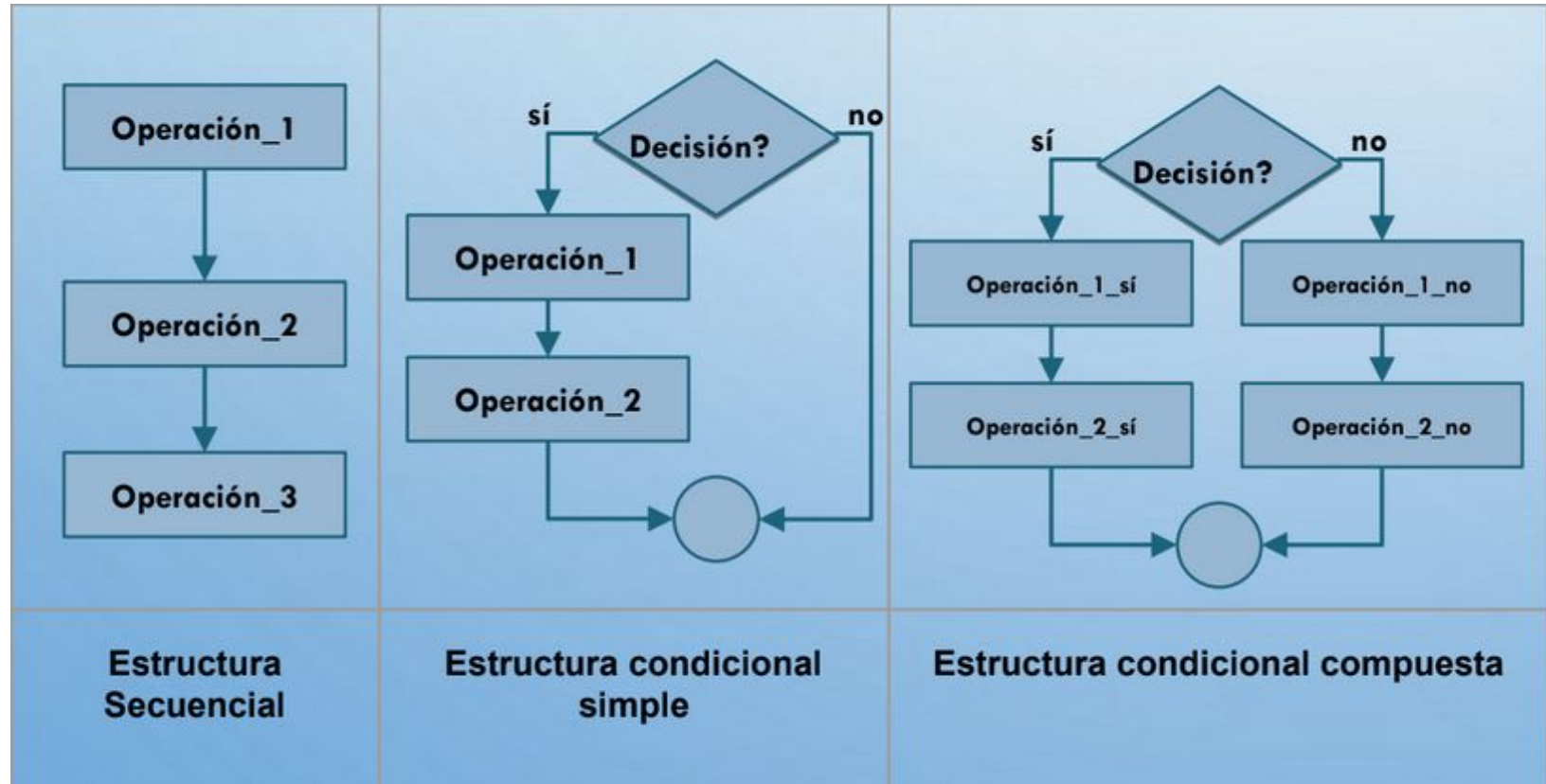


DIAGRAMAS DE FLUJO

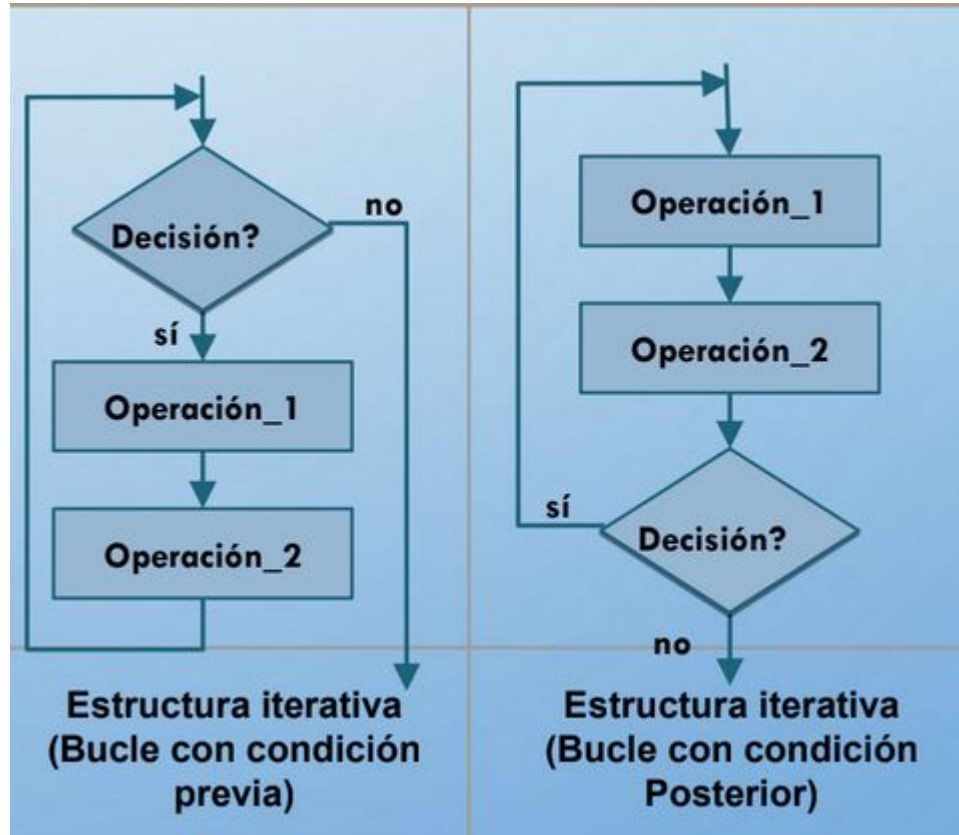
Un diagrama de flujo es una representación gráfica del flujo de ejecución de un algoritmo. Utiliza un conjunto de símbolos estandarizados para expresar de forma clara los posibles caminos de ejecución.



DIAGRAMAS DE FLUJO: ESTRUCTURAS BÁSICAS



DIAGRAMAS DE FLUJO: ESTRUCTURAS BÁSICAS



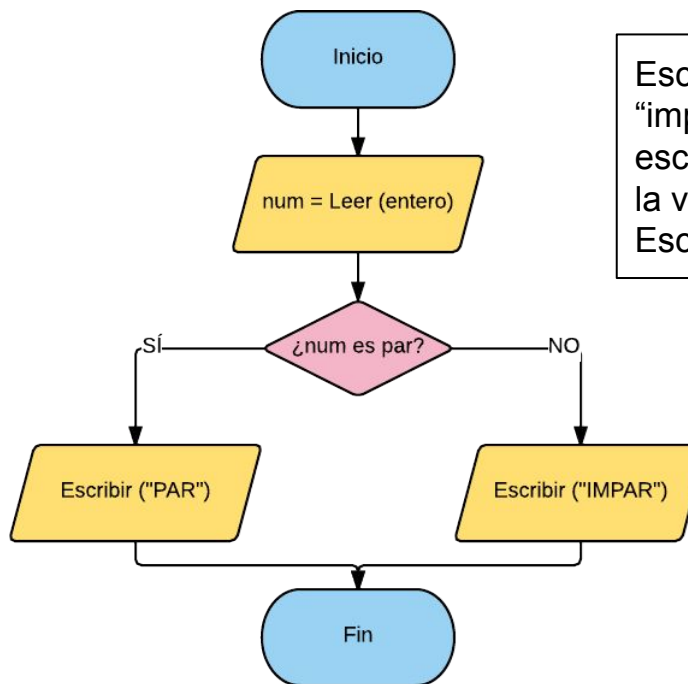
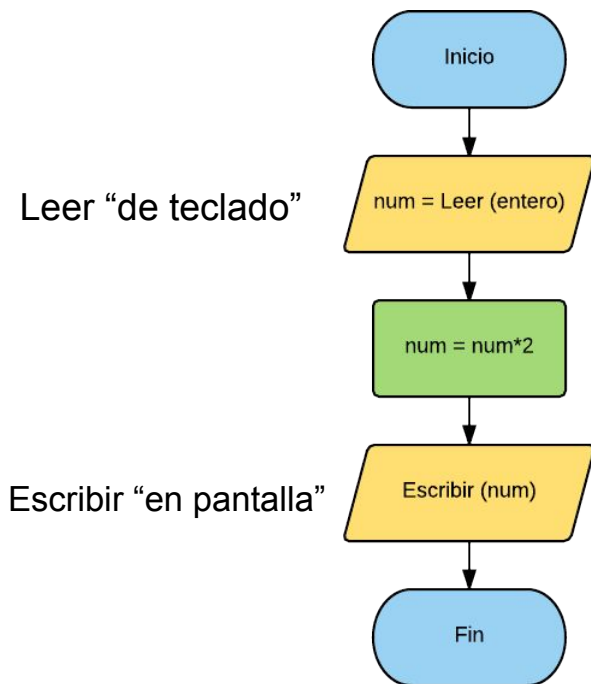
DIAGRAMAS DE FLUJO. REGLAS

Reglas que hay que cumplir en la creación de diagramas de flujo

- ❑ **Todos los símbolos** utilizados deben estar **unidos por líneas de flujo** mostrando la flecha el sentido correcto de la secuencialidad.
- ❑ **No se pueden cruzar las líneas de flujo.**
- ❑ A un **símbolo** de proceso pueden **llegarle varias líneas de flujo pero sólo puede salir una de él.**
- ❑ **Al símbolo de inicio no puede llegarle ninguna línea de flujo.**
- ❑ **Del símbolo de fin no puede salir ninguna línea de flujo pero sí le pueden llegar varias.**

DIAGRAMAS DE FLUJO: EJEMPLOS

- USAREMOS **DRAW.IO** □ [HTTPS://APP.DIAGRAMS.NET/](https://app.diagrams.net/)



Escribir en pantalla cadena "impar". También se pueden escribir variables y cadenas a la vez. Pj:
Escribir("mi numero es", num)

PROGRAMA INFORMÁTICO

Un conjunto de uno o varios algoritmos escritos es un lenguaje de programación con el fin de ser ejecutado en un computador.

Se compone de un conjunto de algoritmos (instrucciones) que indican a la CPU que acción debe ejecutar y sobre qué datos.

Podemos diferenciar:

- ❑ **CÓDIGO FUENTE** (source code): programa escrito por el programador en algún lenguaje de programación. No es directamente ejecutable por la computadora.
- ❑ **CÓDIGO OBJETO** (code object): programa obtenido de la compilación (traducción) del código fuente.
- ❑ **CÓDIGO MÁQUINA:** resultado de enlazar el código objeto con rutinas y librerías para obtener el código directamente ejecutable por la máquina.

LENGUAJES DE PROGRAMACIÓN

DEFINICIÓN

Un **Lenguaje de programación** es un lenguaje estructurado que consta de reglas estrictas.

- **SÍMBOLOS PERMITIDOS Y REGLAS LÉXICAS**
- **REGLAS SINTÁCTICAS**
- **REGLAS SEMÁNTICAS**

LENGUAJES DE PROGRAMACIÓN

DEFINICIÓN

SÍMBOLOS Y REGLAS LÉXICAS: Conjunto de símbolos permitidos y reglas que marcan cómo se construyen las “palabras” del lenguaje, llamadas unidades léxicas o tokens.

Las reglas léxicas reconocen:

- Palabras reservadas del lenguaje:
 - Tipos de datos predefinidos: int, float, char...
 - Construcciones: if, else, where, for...
- Identificadores de variables y funciones: comprueban que los nombres de los identificadores sean válidos (no comience con un número, etc.)
- Operadores: +, -, *, =, etc.
- Otros símbolos del lenguaje: ; { }
- Presencia de comentarios, que son ignorados por el compilador.
- Valores literales: cadenas de caracteres, números.

LENGUAJES DE PROGRAMACIÓN

DEFINICIÓN

SÍMBOLOS Y REGLAS LÉXICAS: Conjunto de símbolos permitidos y reglas que marcan cómo se construyen las “palabras” del lenguaje, llamadas unidades léxicas o tokens.

Por ejemplo, la instrucción en Java: `suma = x + 2;`

generaría 6 tokens:


suma	=	x	+	2	;
------	---	---	---	---	---

LENGUAJES DE PROGRAMACIÓN

DEFINICIÓN

REGLAS SINTÁCTICAS : Reglas que establecen la gramática del lenguaje. Esto es, como deben estar ordenadas las “palabras” o tokens para formar construcciones (“oraciones”) correctas del lenguaje.

Por ejemplo, una declaración y asignación en Java es:

tipo identificador = valor_literal;  int mivar = 5;

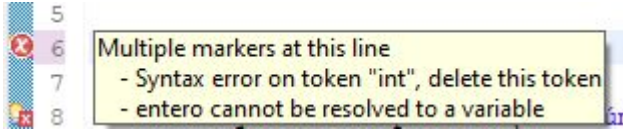
Si no escribimos el tipo, no reconoce la declaración, o si se escribe en cualquier otro orden, dará error, porque no conoce la instrucción como una construcción válida del lenguaje.

LENGUAJES DE PROGRAMACIÓN

DEFINICIÓN

```
entero int = 6;
```

Error en el orden de la declaración



```
double real_error = 6,7;
```

Los números reales, en la parte decimal, se usa el puno '.'. Está reconociendo como número solamente el 6, reconociendo el "7" como otro token.



```
int 5entero = 8;
```

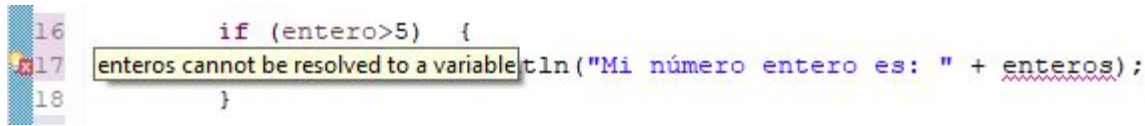
Los IDE actuales, como eclipse, remarcan los errores léxicos y sintácticos

LENGUAJES DE PROGRAMACIÓN

DEFINICIÓN

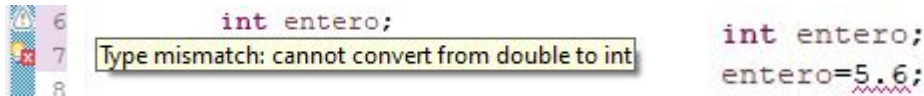
REGLAS SEMÁNTICAS

Comprueban que las declaraciones son correctas, se verifican los tipos de todas las expresiones, si las operaciones se pueden realizar sobre esos tipos, y que existan todas las referencias a variables o funciones.



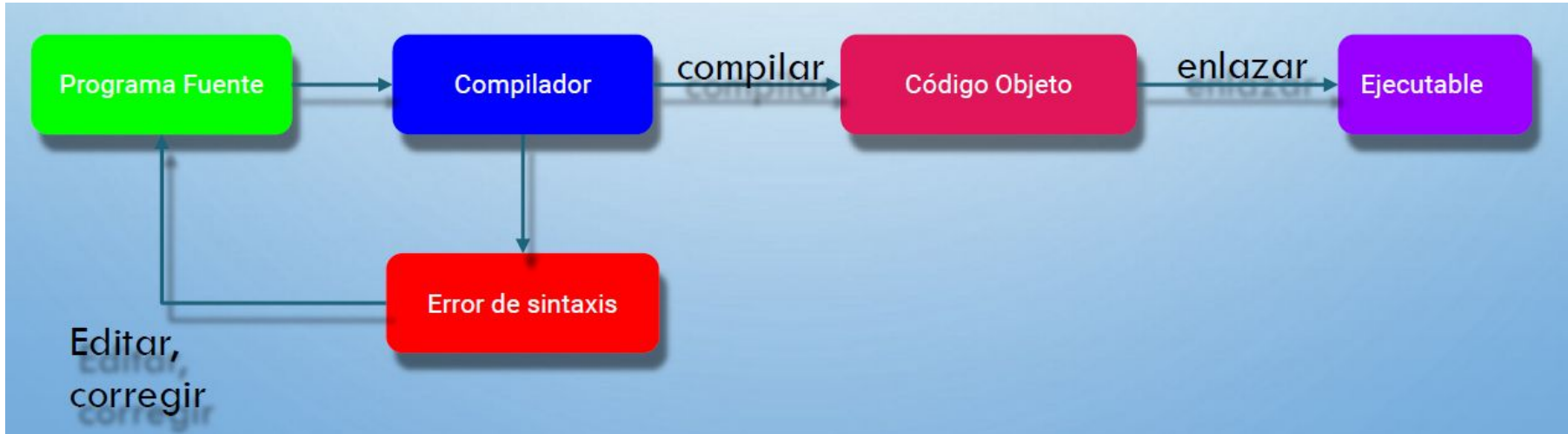
```
16         if (entero>5) {  
17             enteros cannot be resolved to a variable; println("Mi número entero es: " + enteros);  
18         }
```

Las reglas semánticas se encargan de la comprobación de tipos. En Java, si asigno un valor real a una variable de tipo entera, dará error al compilar.



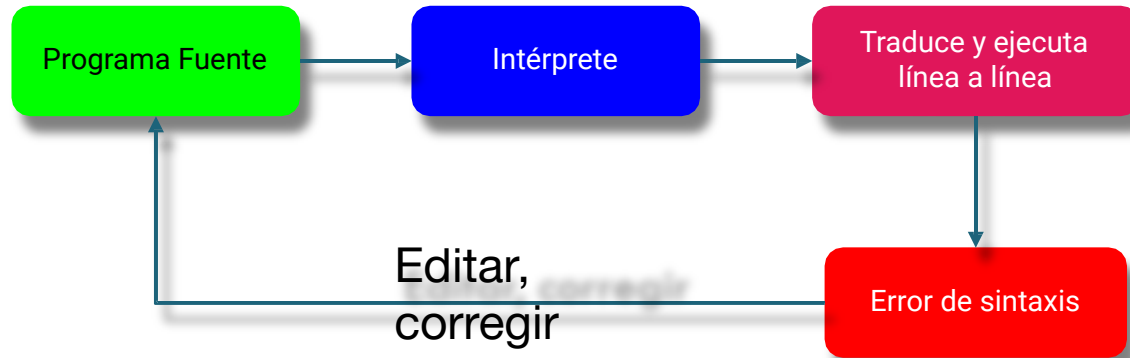
```
6         int entero;  
7         entero=5.6;  
8     }
```

COMPILADORES. Se emplean para traducir lenguajes de alto nivel, generando un archivo ejecutable (.exe en windows). El compilador analiza el código fuente y sólo finaliza la compilación si el programa es correcto de forma léxica, sintáctica y semántica.



TRADUCTORES: INTÉRPRETES

- **INTÉRPRETES.** A diferencia de los compiladores, los intérpretes analizan, traducen y ejecutan instrucción a instrucción sin componer un programa completo en código fuente.



Python es un lenguaje interpretado

LENGUAJES DE PROGRAMACIÓN

PARADIGMAS DE PROGRAMACIÓN

Los paradigmas de la programación hacen referencia a la metodología de un lenguaje de programación. Los paradigmas son importantes porque definen un lenguaje de programación y cómo funciona. Éstos incluyen un conjunto de ideas que un lenguaje de programación puede usar para realizar tareas en términos de código.

Algunos lenguajes son específicos de un paradigma de programación en concreto, sin embargos otros son multiparadigmas.

PARADIGMAS DE PROGRAMACIÓN

PROGRAMACIÓN ESTRUCTURADA

Es una técnica que permite crear programas más sencillos, fáciles de entender, de mantener y de probar. El principal inconveniente de este método de programación, es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático su manejo. Un **programa estructurado** es un programa con **un solo punto de entrada y un solo punto de salida** que se basa en el uso de **tres estructuras lógicas de control**:

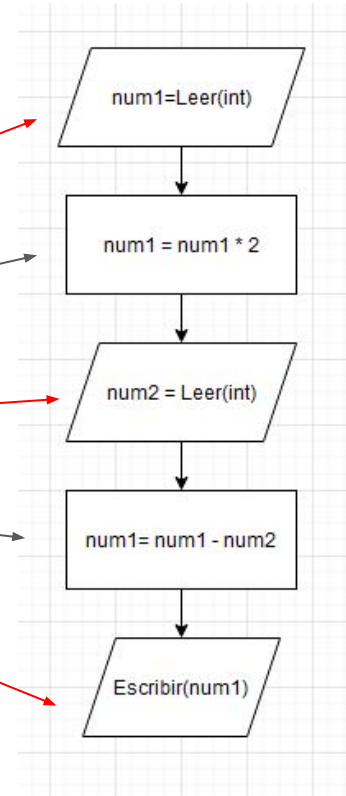
- **Secuencial**: sucesión simple de dos o más operaciones.
- **Selección**: bifurcación condicional de una o más operaciones.
- **Iteración**: repetición de una o más operaciones mientras se cumple una condición.

PARADIGMAS DE PROGRAMACIÓN

- **Secuencial:** sucesión simple de dos o más operaciones.

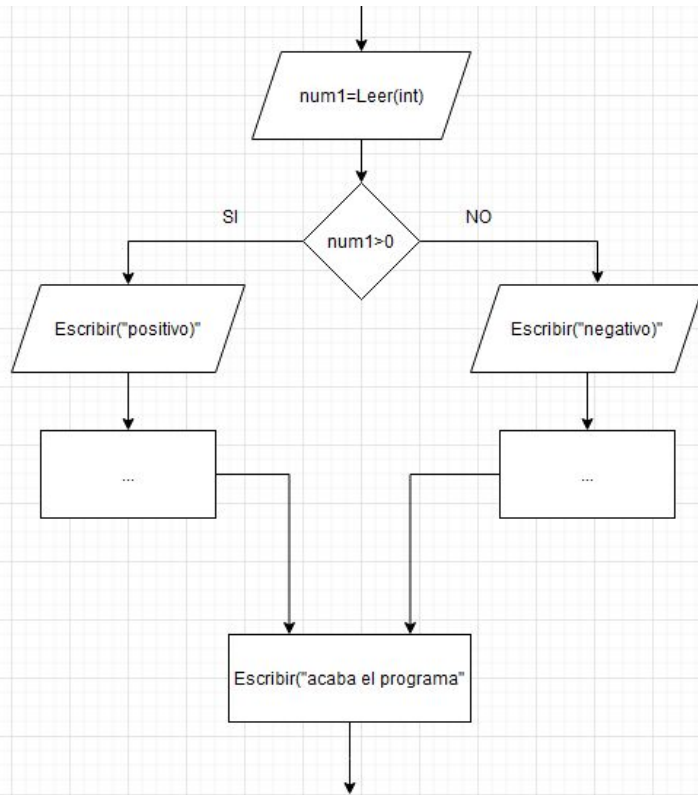
flujo de
ejecución

```
//num1 = leer(int)  
num1 = num1 * 2;  
//num2 = leer(int)  
num1 = num1 - num2;  
System.out.println(num1);
```



PARADIGMAS DE PROGRAMACIÓN

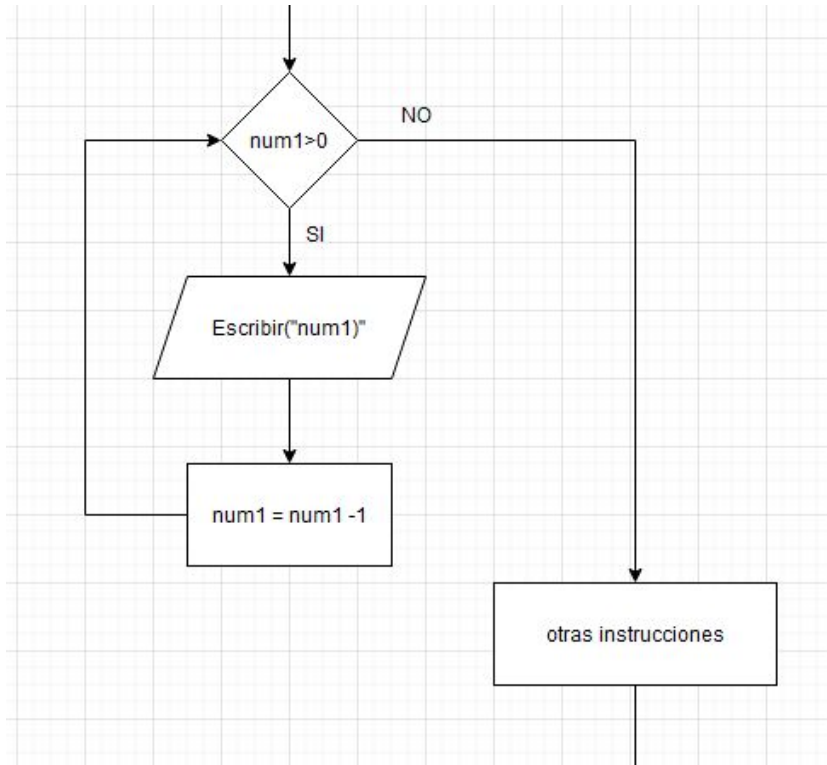
- **Selección:** bifurcación condicional de una o más operaciones.



```
if (num1>0) {  
    System.out.println("positivo");  
    //otras instrucciones secuenciales  
}  
else {  
    System.out.println("negativo");  
    //otras instrucciones secuenciales  
}  
  
//otras instrucciones secuenciales  
System.out.println("acaba el programa");
```

PARADIGMAS DE PROGRAMACIÓN

- **Iteración:** repetición de una o más operaciones mientras se cumple una condición.



```
// otras instrucciones secuenciales  
  
while (num1 > 0) {  
    System.out.println(num1);  
    num1 = num1 - 1;  
}  
  
// otras instrucciones secuenciales
```

PARADIGMAS DE PROGRAMACIÓN

Programación modular

Resuelve el principal problema de la programación estructurada dividiendo el programa en un conjunto de módulos, cada uno de los cuales desempeña una tarea necesaria para el correcto funcionamiento del programa global. Los módulos son interdependientes, y son codificados y compilados por separado.

Programación orientada a objetos

La programación orientada a objetos expresa un programa como un conjunto de objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

PSEUDOCÓDIGO

EL PSEUDOCÓDIGO DESCRIBE UN ALGORITMO UTILIZANDO UNA MEZCLA DE FRASES EN LENGUAJE COMÚN, INSTRUCCIONES DE PROGRAMACIÓN Y PALABRAS CLAVE QUE DEFINEN LAS ESTRUCTURAS BÁSICAS. SU OBJETIVO ES PERMITIR QUE EL PROGRAMADOR SE CENTRE EN LOS ASPECTOS LÓGICOS DE LA SOLUCIÓN A UN PROBLEMA.

EXISTEN VARIAS VENTAJAS DE UTILIZAR UN PSEUDOCÓDIGO A UN DIAGRAMA DE FLUJO:

- PERMITE REPRESENTAR DE FORMA FÁCIL OPERACIONES REPETITIVAS COMPLEJAS.
- ES MÁS SENCILLA LA TAREA DE PASAR DE PSEUDOCÓDIGO A UN LENGUAJE DE PROGRAMACIÓN FORMAL.
- SIGUIENDO LAS REGLAS DE INDENTACIÓN SE PUEDE OBSERVAR CLARAMENTE LOS NIVELES EN LA ESTRUCTURA DEL PROGRAMA

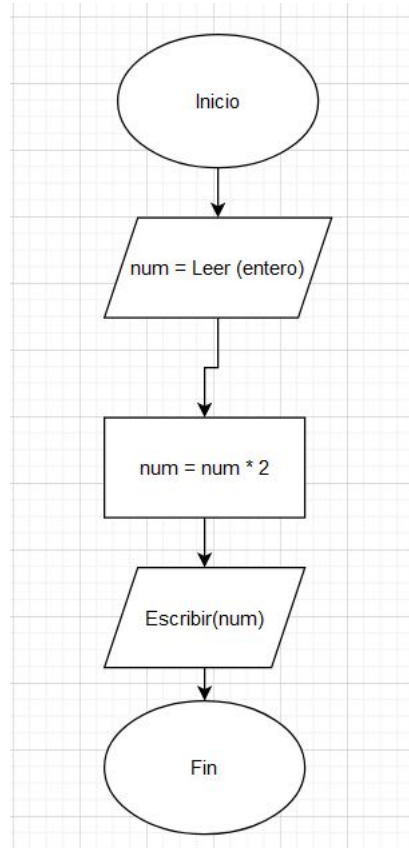
PSEUDOCÓDIGO: ESTRUCTURAS BÁSICAS (O COMPONENTES)

- FUNCIONES Y OPERACIONES
 - ASIGNACIONES (2 formas): $X \leftarrow Y$; resultado = $2 * 4$
 - ENTRADA: LEER (TIPO) Y SALIDA: ESCRIBIR (VARIABLE)
- ESTRUCTURA SECUENCIAL (1ª COLUMNA).
- ESTRUCTURAS DE CONTROL: ALTERNATIVA (2ª Y 3ª COLUMNA) E ITERATIVA (4ª, 5ª Y 6ª COLUMNA)

Estructuras básicas en pseudocódigo

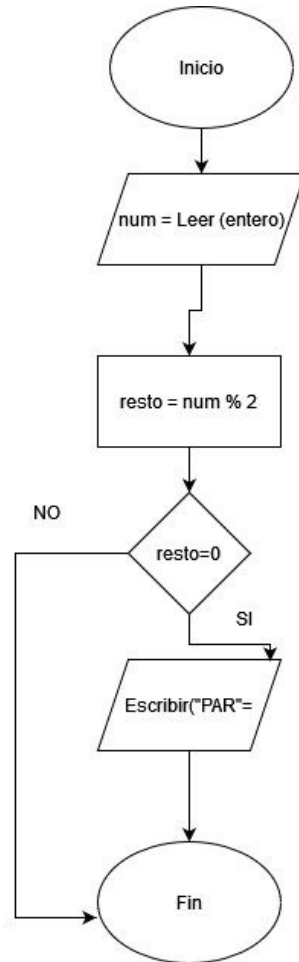
acción_1 acción_2 ... acción_n	SI condición ENTONCES instrucciones_si FIN SI	SI condición ENTONCES instrucciones_si SiNo instrucciones_no FIN SI	MIENTRAS condición HACER instrucciones_mientras FIN MIENTRAS	REPETIR instrucciones_hacer HASTA QUE condición	PARA n veces instrucciones_repetir_n FIN REPETIR
Estructura Secuencial	Estructura condicional simple	Estructura condicional compuesta	Estructura iterativa (Bucle con condición previa)	Estructura iterativa (Bucle con condición Posterior)	Estructura iterativa (Bucle que itera un número « n » conocidos de veces.)

Secuencial



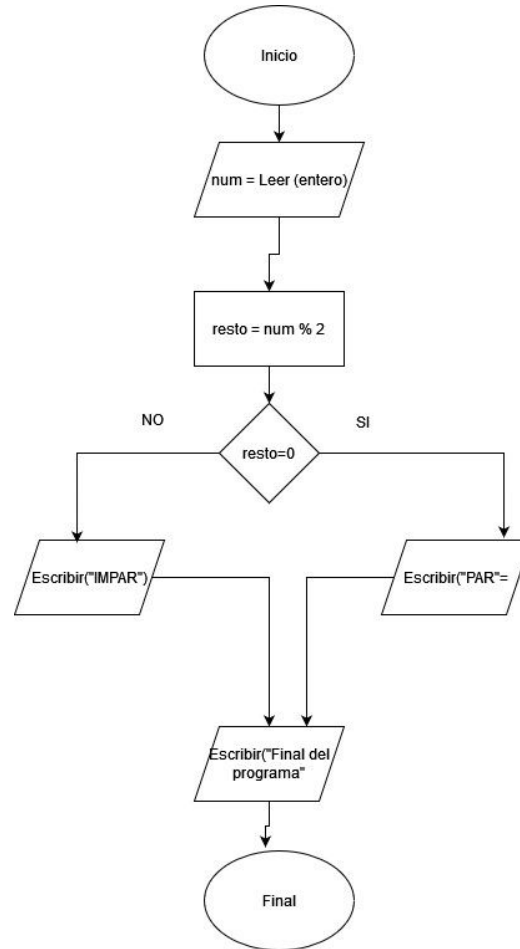
```
1  Algoritmo secuencial
2      Leer num
3      num = num * 2
4      Escribir num
5  FinAlgoritmo
6
```


Selección simple



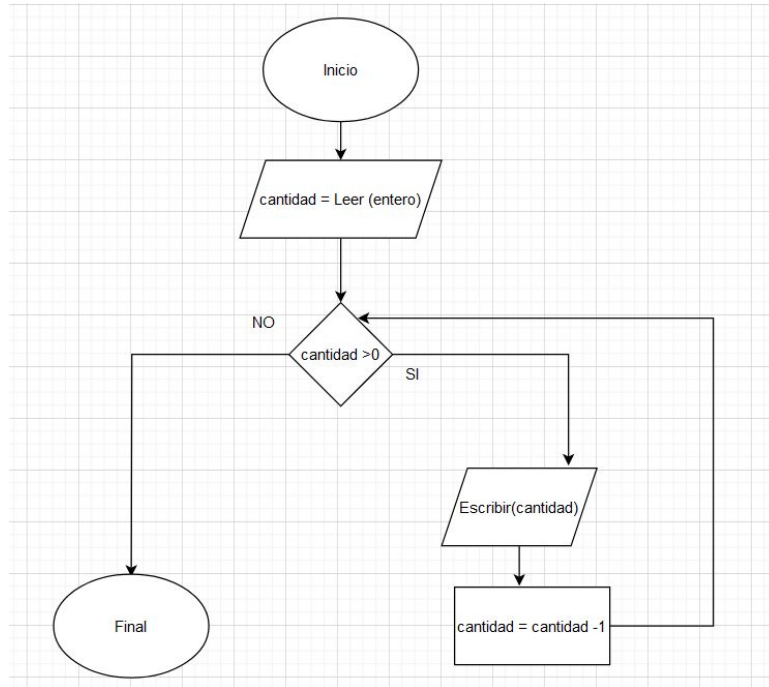
```
1  Algoritmo secuencial
2      Leer num
3      resto = num % 2
4      Si resto=0 Entonces
5          ..... Escribir "PAR"
6  +      Fin Si
7  FinAlgoritmo
8
```

Selección doble



```
1  Algoritmo secuencial
2      Leer num
3      resto = num % 2
4      Si resto=0 Entonces
5          Escribir "PAR"
6      SiNo
7          Escribir "IMPAR"
8      Fin Si
9      Escribir "Final del programa"
10 FinAlgoritmo
11
```

Iteración mientras



```
1  Algoritmo secuencial
2      Leer cantidad
3      Mientras cantidad > 0 Hacer
4          Escribir cantidad
5          cantidad = cantidad - 1
6      Fin Mientras
7  FinAlgoritmo
8
```

Iteración mientras

Verdadero/Falso

```
1  Algoritmo ejemplo
2      salir = Falso
3      suma = 0
4      Mientras salir = Falso Hacer
5          Escribir "Inserte un número"
6          Leer numEntero
7          suma = suma + numEntero
8          Si suma > 20 Entonces
9              salir = Verdadero
10         FinSi
11     Fin Mientras
12     Escribir "La suma es :", suma
13     Escribir "Final del algoritmo"
14 FinAlgoritmo
15 |
```

Iteración Para (for)

```
1  Algoritmo ejemplo
2      // Para en PseInt va desde 1 hasta 10
3      Para x = 1 Hasta 10 Con Paso 1 Hacer
4          Escribir x
5      Fin Para
6  FinAlgoritmo
7
```

Función Escribir con concatenación

```
1  Algoritmo ejemplo
2
3      Para i=0 Hasta 4 Con Paso 1 Hacer
4          //En una misma llamada a "Escribir" pasamos literales cadenas y variables
5          Escribir "Numero de ciclo: ", i
6      Fin Para
7      Escribir "Final del algoritmo"
8  FinAlgoritmo
9
```

Para/For anidados. Las estructuras de control como If-else, while, for se pueden anidar unas dentro de otras.

```
1  Algoritmo ejemplo
2      Para i=0 Hasta 3 Con Paso 1 Hacer
3          Para j=0 Hasta 3 Con Paso 1 Hacer
4              Escribir "Resultado de ", i, " * ", j, " es: ", i*j
5          Fin Para
6      Fin Para
7  FinAlgoritmo
8
```