



IES Sant Vicent Ferrer
Algemés



**GENERALITAT
VALENCIANA**
CONSELLERIA D'EDUCACIÓ,
INVESTIGACIÓ, CULTURA I ESPORT



Fondo Social Europeo

Programación

UD 1: Introducción a la Programación

Ciclo Formativo de Grado Superior
Desarrollo de Aplicaciones Web

Jose Chamorro Molina

Introducción a la Programación

ORDEN 60/2012, de 25 de septiembre, de la Conselleria de Educación, Formación y Empleo por la que se establece para la Comunitat Valenciana el currículo del ciclo formativo de Grado Superior correspondiente al título de Técnico Superior en Desarrollo de Aplicaciones Web. [2012/9149]

Contenidos:

- 1.- Identificación de los elementos de un programa informático:
 - 1.1.- Estructura y bloques fundamentales.
 - 1.2.- Soluciones y proyectos.
 - 1.3.- Utilización de los entornos integrados de desarrollo.

Real Decreto 686/2010, de 20 de mayo, por el que se establece el título de Técnico Superior en Desarrollo de Aplicaciones Web y se fijan sus enseñanzas mínimas.

Resultados de aprendizaje:

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

Criterios de evaluación:

- 1.a) Se han identificado los bloques que componen la estructura de un programa informático.
- 1.b) Se han creado proyectos de desarrollo de aplicaciones
- 1.c) Se han utilizado entornos integrados de desarrollo.

Competencias profesionales, personales y sociales:

- p) Adaptarse a las nuevas situaciones laborales, manteniendo actualizados los conocimientos científicos, técnicos y tecnológicos relativos a su entorno profesional, gestionando su formación y los recursos existentes en el aprendizaje a lo largo de la vida y utilizando las tecnologías de la información y la comunicación.

Introducción a la Programación

- 1.- Algoritmos y Programas
- 2.- Lenguajes de Programación. Tipos
- 3.- Entornos de desarrollo
- 4.- Representación de los algoritmos
 - 4.1.- Pseudocódigo
 - 4.2.- Diagramas de Flujo

1.- Algoritmos y programas

1.- Algoritmos y programas

Definiciones:

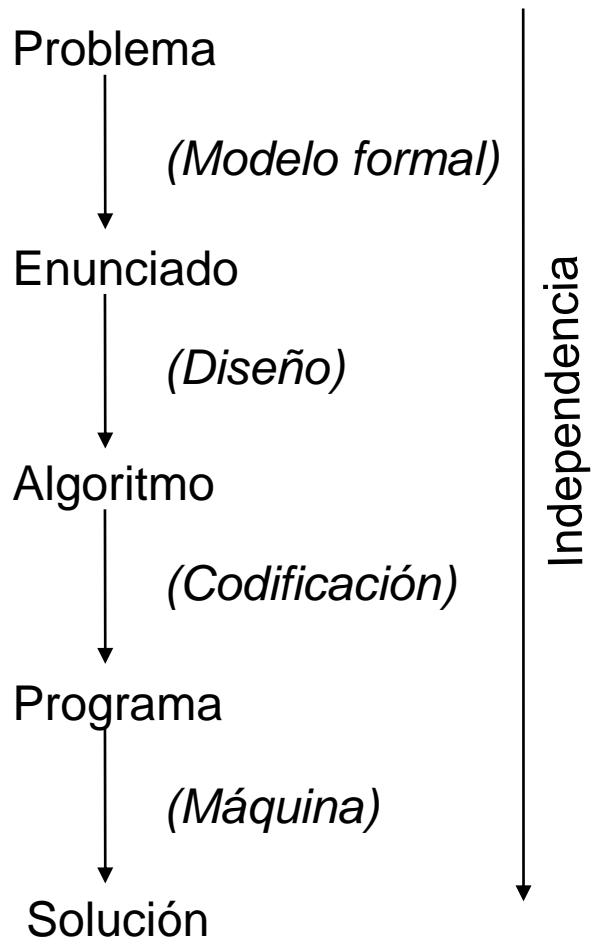
Algoritmo: Secuencia finita de reglas o instrucciones que especifican un conjunto de operaciones, que al ser ejecutadas por un agente ejecutor (máquina real o abstracta), resuelve cualquier problema de un tipo determinado en un tiempo finito.

Programa informático: Conjunto de instrucciones que implementan un algoritmo. Una vez ejecutadas, las instrucciones realizarán una o varias tareas en un ordenador.

Programación: Es el proceso por el cual una persona desarrolla un programa valiéndose de una herramienta que le permita escribir el código (el cual puede estar en uno o varios lenguajes, tales como C++, Java y Python entre otros) y de otra que sea capaz de “traducirlo” a lo que se conoce como lenguaje de máquina, el cual puede ser entendido por un microprocesador.

PROGRAMACIÓN = ALGORITMOS + ESTRUCTURAS DE DATOS

1.- Algoritmos y programas



- Dado un problema intentaremos encontrar un modelo formal que nos permita representarlo como un enunciado.
- Mediante una técnica de diseño realizaremos un algoritmo que resuelva el problema
- Mediante un lenguaje de programación realizaremos el programa.
- Una vez ejecutado el programa por un agente ejecutor obtendremos un resultado que nos dará la solución del problema.

1.- Algoritmos y programas

Algoritmos:

Datos y variables:

ALGORITMO = Técnica para resolver problemas a través de una serie de pasos intermedios hasta llegar a resultado.

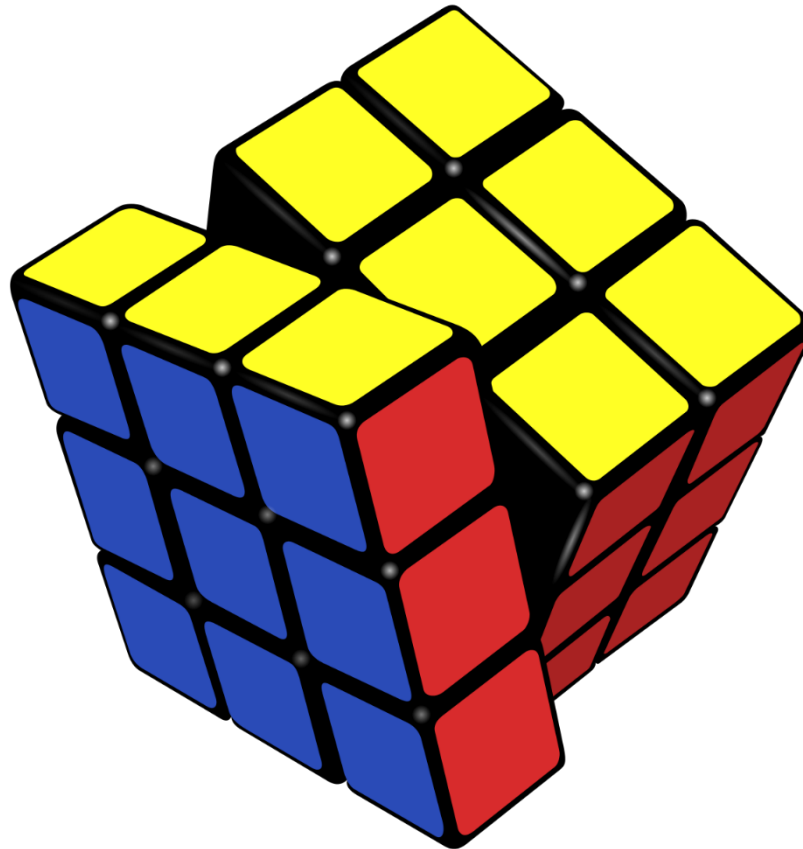
Pero siempre vamos a manejar distintos tipos de datos en un algoritmo ... tiempo, euros, cantidad de productos ...

Y necesitaremos almacenar los resultados de los cálculos intermedios de cada algoritmo.

1.- Algoritmos y programas

Ejemplo de algoritmo:

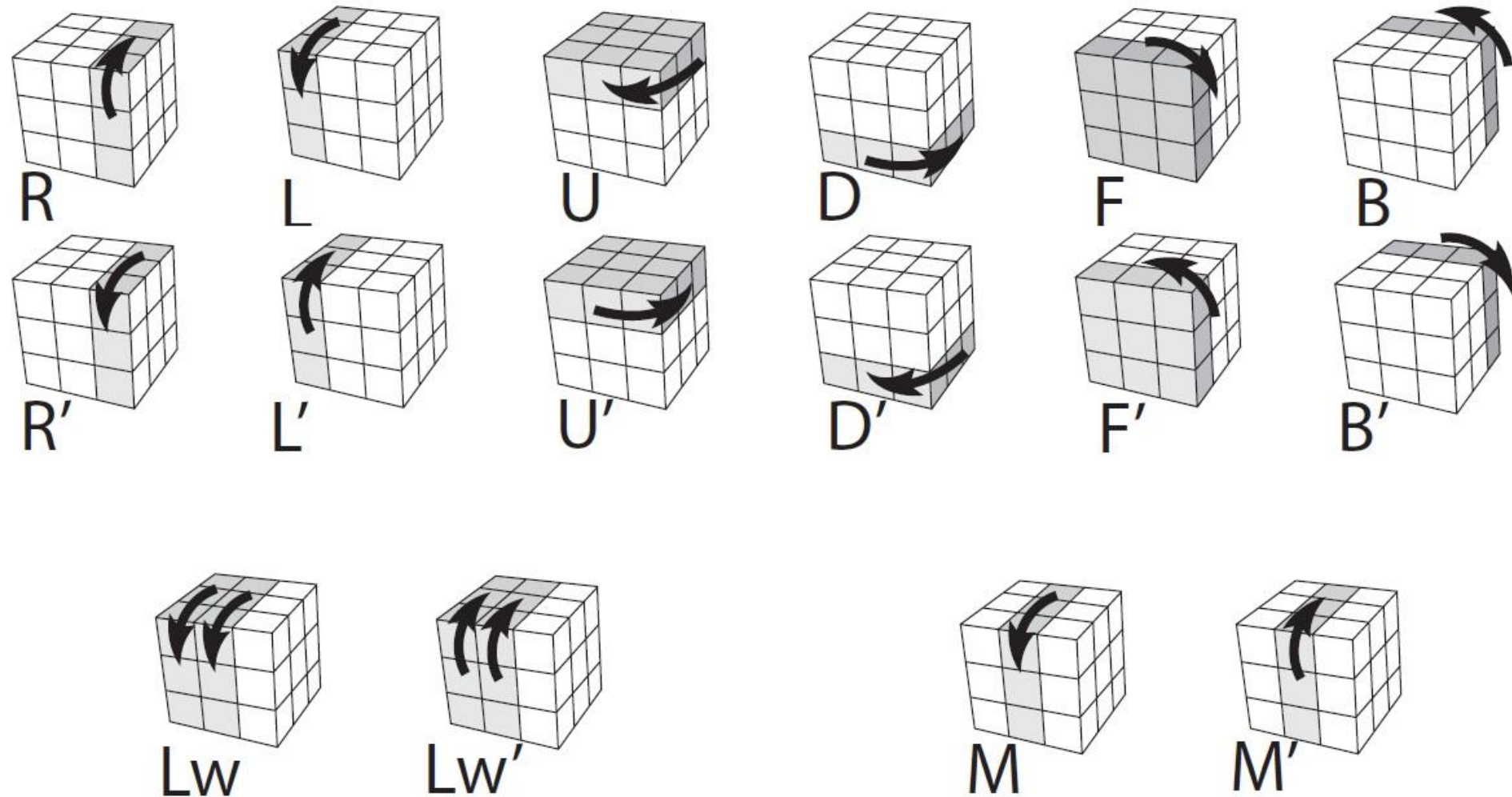
Resolver un cubo de rubik



<https://www.youtube.com/watch?v=CLzWY-SKAqk>

1.- Algoritmos y programas

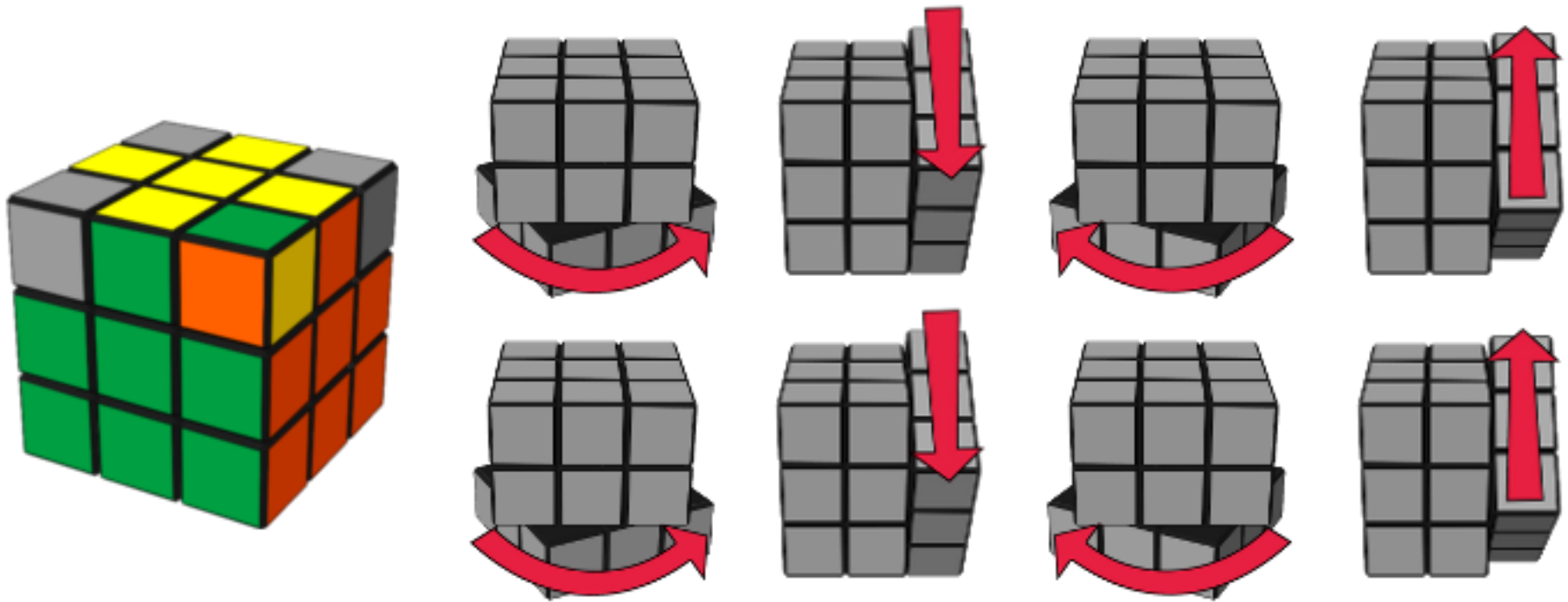
Movimientos en un cubo de rubik



1.- Algoritmos y programas

Ejemplo de algoritmo:

Resolver un cubo de rubik

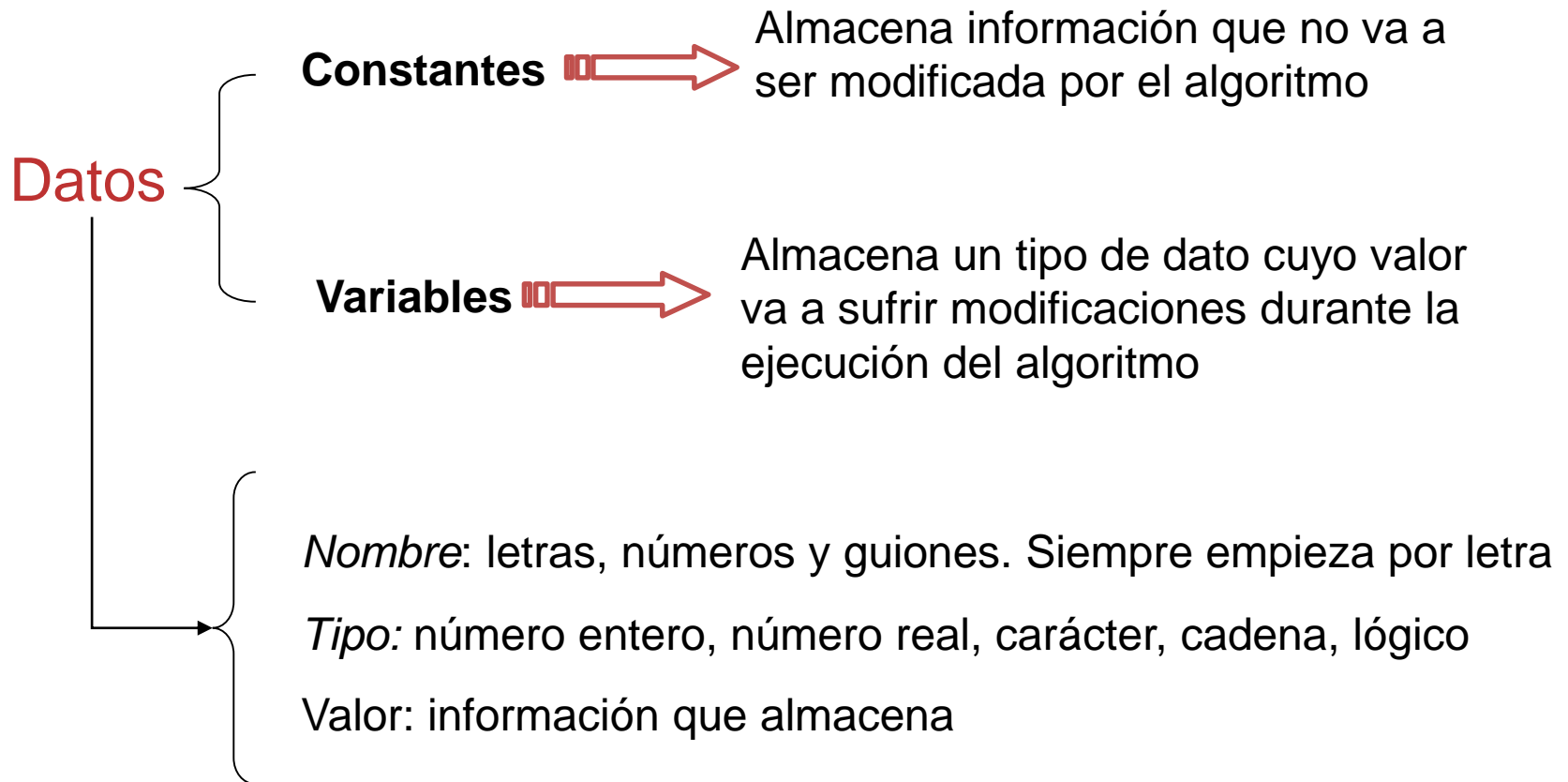


Lo difícil es obtener el algoritmo,
realizarlo (o programarlo) es sencillo

$D - R' - D' - R$

$D - R' - D' - R$

1.- Algoritmos y programas



1.- Algoritmos y programas

Ejemplo de algoritmo:

Reparación de un ordenador

- Me traen un ordenador estropeado
- Empiezo a contar el tiempo.
- Compruebo el pc y detecto las averias.
- Cambio las piezas estropeadas y ... funciona.
- Anoto las piezas cambiadas.
- Le pregunto al dueño si paga con tarjeta o en efectivo. Con tarjeta se recarga un 2%.
- Cobro al dueño del pc por el tiempo trabajado por horas y las piezas cambiadas.
- Anoto sus datos de cliente en base datos. Fin

1.- Algoritmos y programas

Elementos usados

He almacenado el tiempo de reparación. Variable numérica entera.

- El precio por hora es constante.
- El precio de cada pieza no es exacto en euros, tiene céntimos.
Variable numérica real.
- Pago con tarjeta. Verdadero o falso. Variable lógica.
- Almaceno la suma total en variable. ¿Tipo?
- Almaceno los datos de cliente en variable tipo cadena de caracteres.

1.- Algoritmos y programas

Tipos de datos

Enteros: Son los números enteros. Como horas exactas o euros sin céntimos

Reales: Son los números con decimales. Como precios de productos.

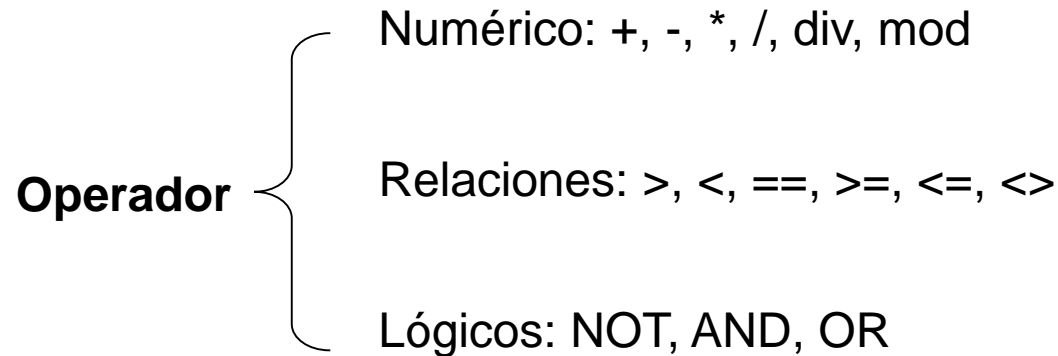
Lógicos: Tienen dos valores Verdadero o Falso.

Carácter: Son las letras del alfabeto.

Cadena de caracteres: Son un conjunto de caracteres como el nombre y apellidos de una persona.

1.- Algoritmos y programas

Instrucciones



Operando: es una variable, una constante, etc.. un elemento que tiene un valor.

Expresión: constante o variable, es un conjunto de operadores y operandos.

Ejemplo: $x = 12 + 3 * 4$

1.- Algoritmos y programas

Características de los algoritmos

Un algoritmo debe ser:

- ✓ **Preciso**, debe indicar el orden de realización de cada paso.
- ✓ **Definido**, si se sigue un algoritmo dos veces se debe obtener el mismo resultado cada vez.
- ✓ **Finito**, debe terminar en un número finito de pasos

2.- Lenguajes de Programación. Tipos

2.- Lenguajes de Programación

Definición: “Un lenguaje de programación es un lenguaje formal que especifica una serie de instrucciones para que una computadora produzca diversas clases de datos. Los lenguajes de programación pueden usarse para crear programas que pongan en práctica algoritmos específicos que controlen el comportamiento físico y lógico de una computadora.” ([Wikipedia](#))

Ejemplo. Hola mundo en c++

```
#include <iostream>

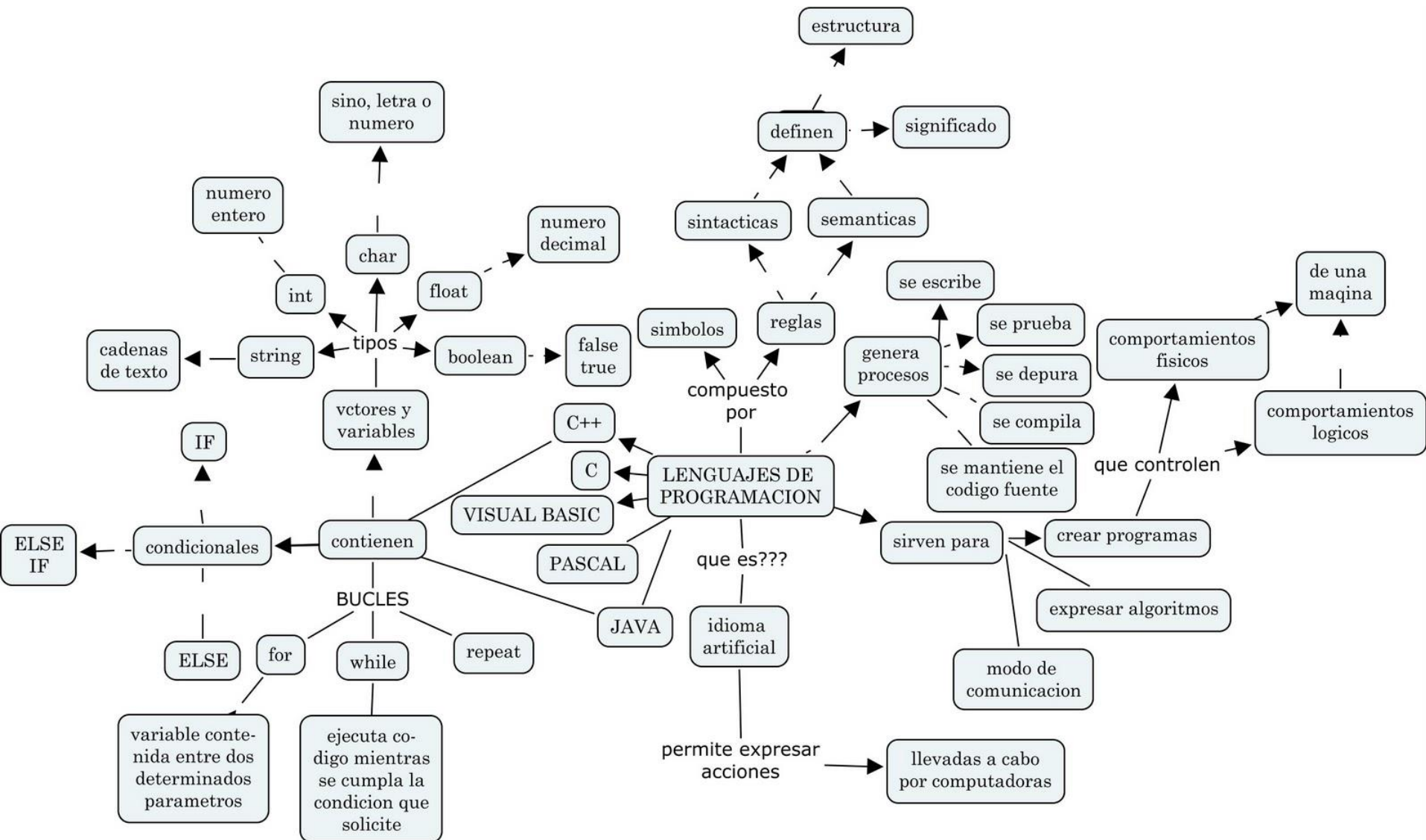
using namespace std;

int main() {
    cout << "Hola Mundo" << endl;
    return 0;
}
```

Ejemplo. Hola mundo en Java

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hola mundo");
    }
}
```

2.- Lenguajes de Programación



2.- Lenguajes de Programación



2.- Lenguajes de Programación

Ejercicio:

- ✓ Otras definiciones de Lenguaje de Programación
- ✓ Lenguajes de 1era, 2nda, 3era y 4ta generación (¿5ta?)
- ✓ Clasificación de los lenguajes de programación según:
 - La proximidad del lenguaje a la máquina (Alto nivel Vs Bajo nivel)
 - En función del paradigma de programación (Imperativos Vs Declarativos)
 - La traducción al código máquina (Interpretados Vs Compilados)
 - Según su funcionalidad

3.- Entornos de Desarrollo

3.- Entornos de Desarrollo

Definición: “Un entorno de desarrollo integrado, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.” ([Wikipedia](#))

Los IDEs pueden estar dedicados a un lenguaje de programación específico o servir para distintos lenguajes aunque hoy en día suelen ser multilenguaje.

Existen IDEs multiplataforma, es decir, se pueden ejecutar sobre distintos SO y arquitecturas. Normalmente desarrollados en JAVA.

3.- Entornos de Desarrollo

Un IDE, consta al menos de los siguientes elementos:

- ✓ Un editor de texto o código. Actualmente con sintaxis coloreada, predicción de texto y navegación por el código
- ✓ Un compilador y/o intérprete.
- ✓ Un depurador de errores. (Breakpoints, ejecución paso a paso, visualización de variables, pila, etc...)
- ✓ Opcionalmente. Funciones para la construcción de interfaces gráficas (GUI)
- ✓ Opcionalmente. Algún sistema de control de versiones.
- ✓ Opcionalmente. Herramientas de generación de pruebas y documentación de código.
- ✓ Etc, etc...

3.- Entornos de Desarrollo

Algunos de los IDE más utilizados son:

| | Windows | Linux | Java |
|----------------|---|---|---|
| Código abierto | <ul style="list-style-type: none">- DevC++. IDE completo para utilizar MinGW (Minimalist GNU for Windows)- Visual-MinGW. Diseñado para utilizar MinGW | <ul style="list-style-type: none">- Emacs, Vim. Editores de textos tradicionales de Unix, muy engorrosos.- Anjuta. C/C++, incorpora las herramientas GNU gcc, make, gdb, entre otros- Kdevelop. C/C++, Fortran, Pascal, Perl... Permite desarrollo de interfaces gráficas. | <ul style="list-style-type: none">- Eclipse. IDE independiente de la plataforma. Extensible mediante módulos. Da soporte por defecto para Java ampliable a otros lenguajes. Recomendado por Google para el desarrollo para Android- Netbeans. Idem eclipse |
| Propietarios | <ul style="list-style-type: none">- Visual Studio. El IDE más popular de Microsoft. Admite C#, C++ y Visual Basic- C++ Builder. Delphi. RAD multiplataforma de Embarcadero basados en ObjectPascal y C++ | <ul style="list-style-type: none">- Code Forge. Admite mas de 30 lenguajes.- Maguma Workbench | <ul style="list-style-type: none">- Jbuilder. El más popular de los IDE comerciales para Java. Producto de Embarcadero compañía que cuenta también con Delphi y C++ Builder.- AIDE. Android para Android |

3.- Entornos de Desarrollo

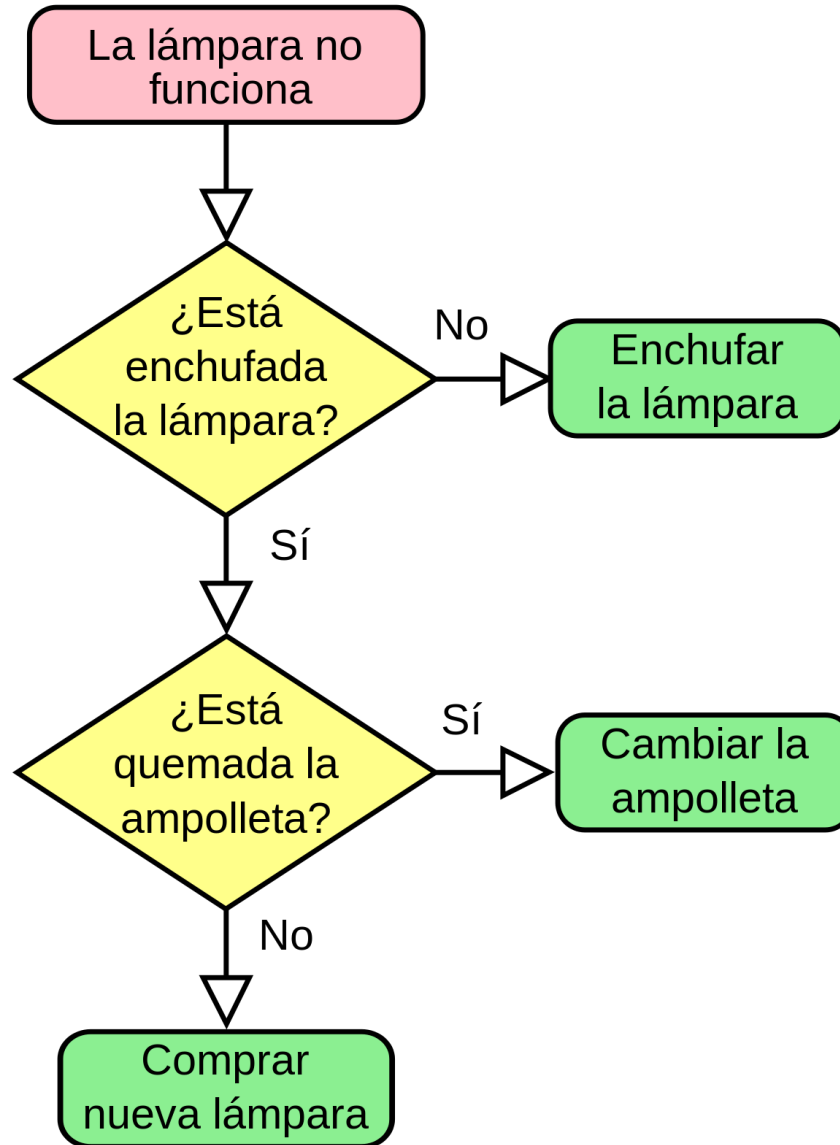
IDE para 1ºDAW

La última versión de Eclipse



4.- Representación de algoritmos

4.- Representación de algoritmos



4.- Representación de algoritmos

1. Inicio del programa

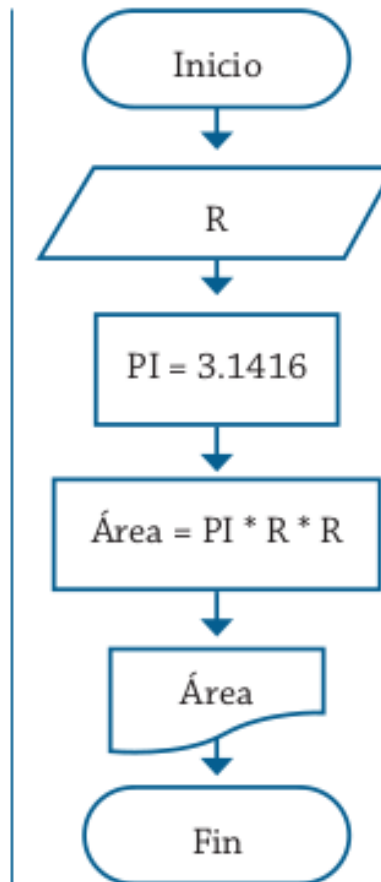
2. Se lee el valor de la variable R.

3. Se establece la variable constante Pi.

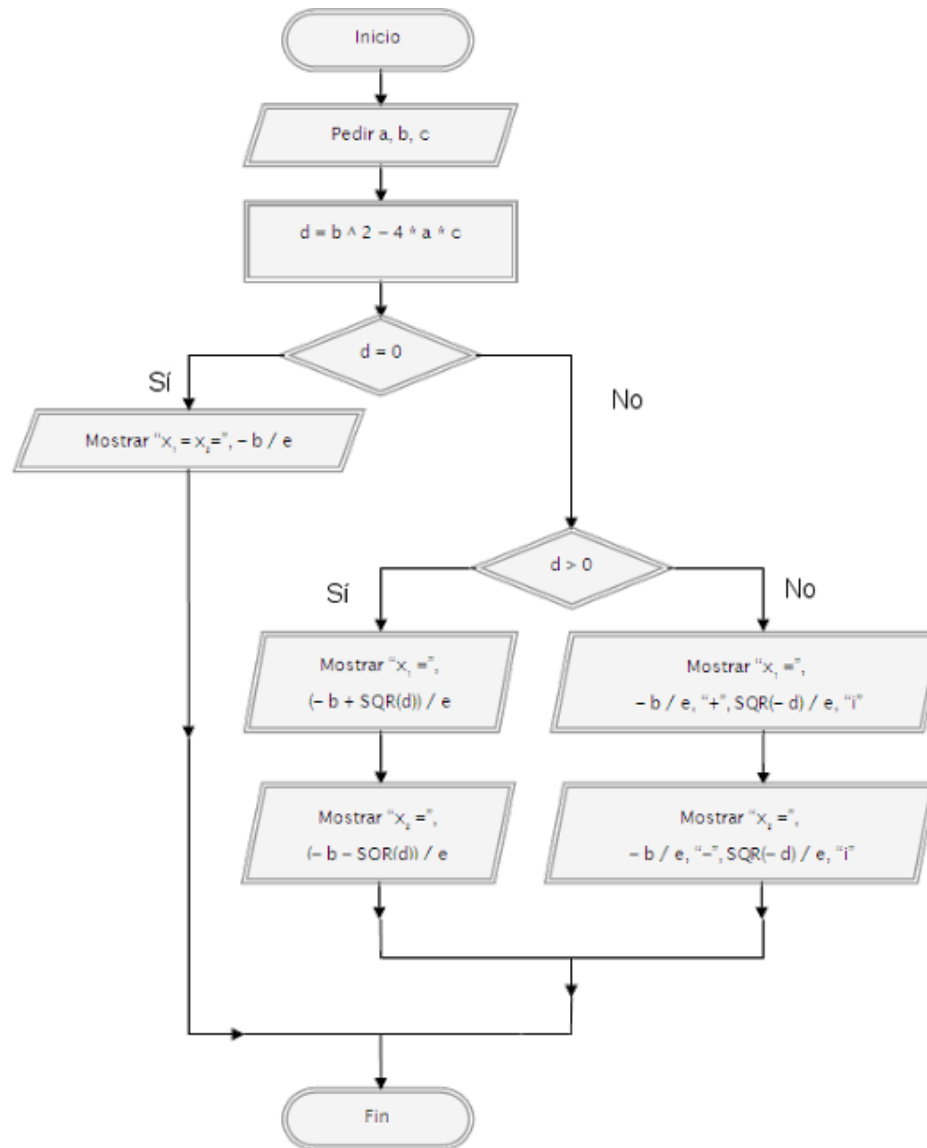
4. Se calcula el valor de la variable **Area** por medio de una formula.

5. Se imprime en pantalla el valor de la variabl **Area**.

6. Fin de Programa.

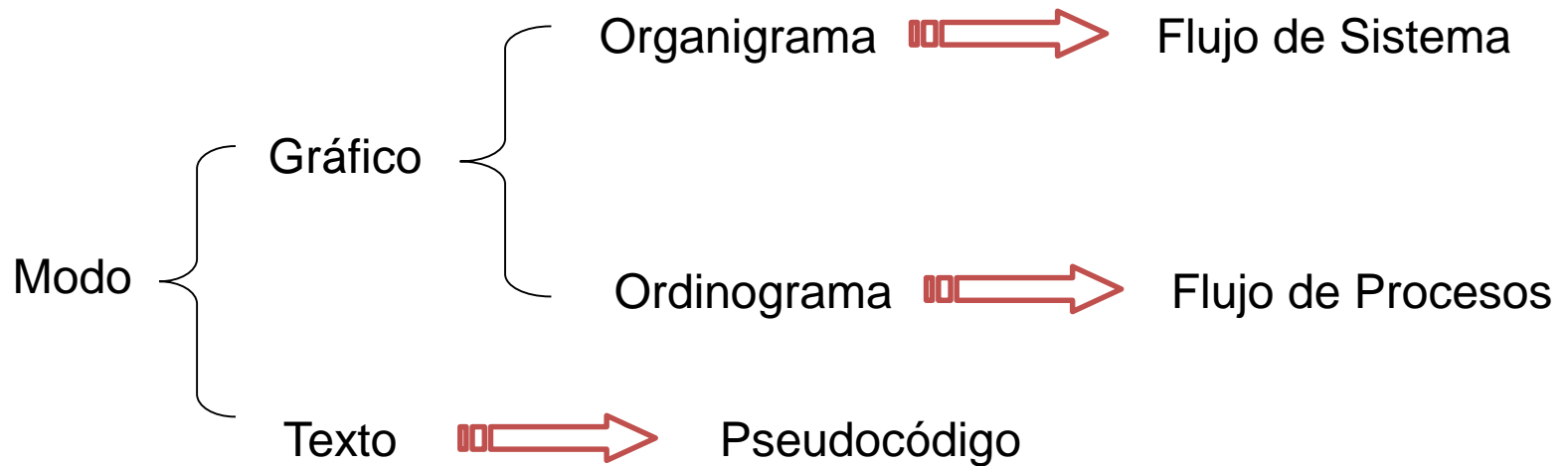


4.- Representación de algoritmos



4.- Representación de algoritmos

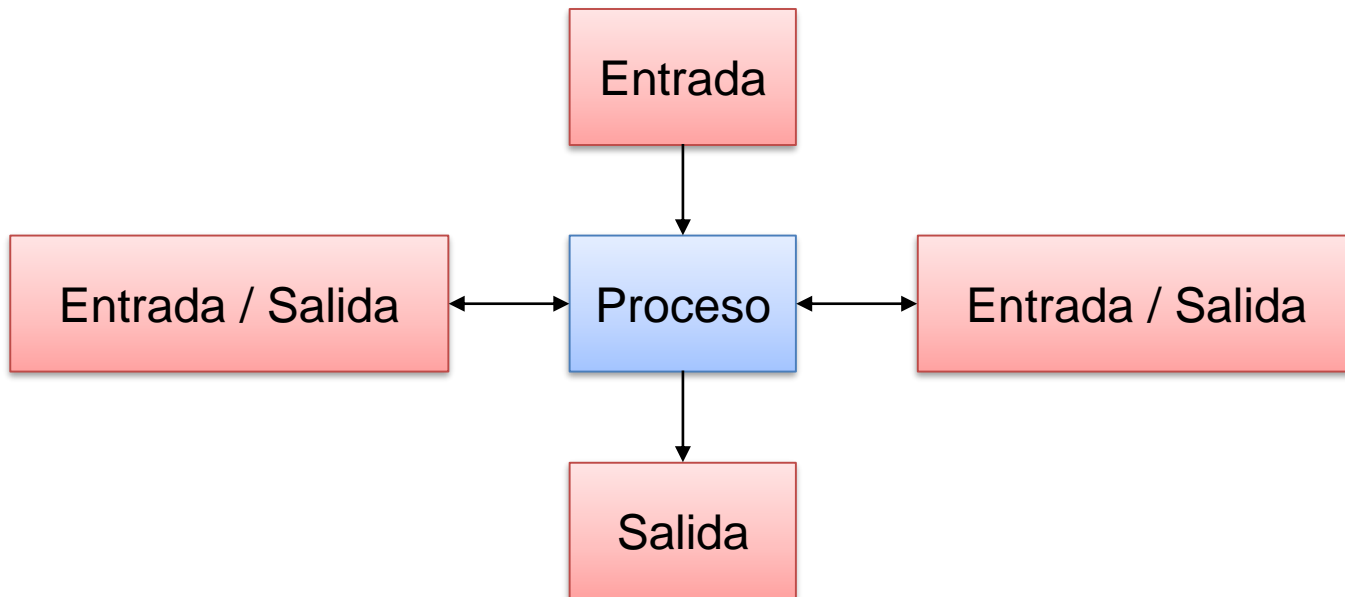
Los algoritmos se pueden representar de las siguientes formas:



4.- Representación de algoritmos

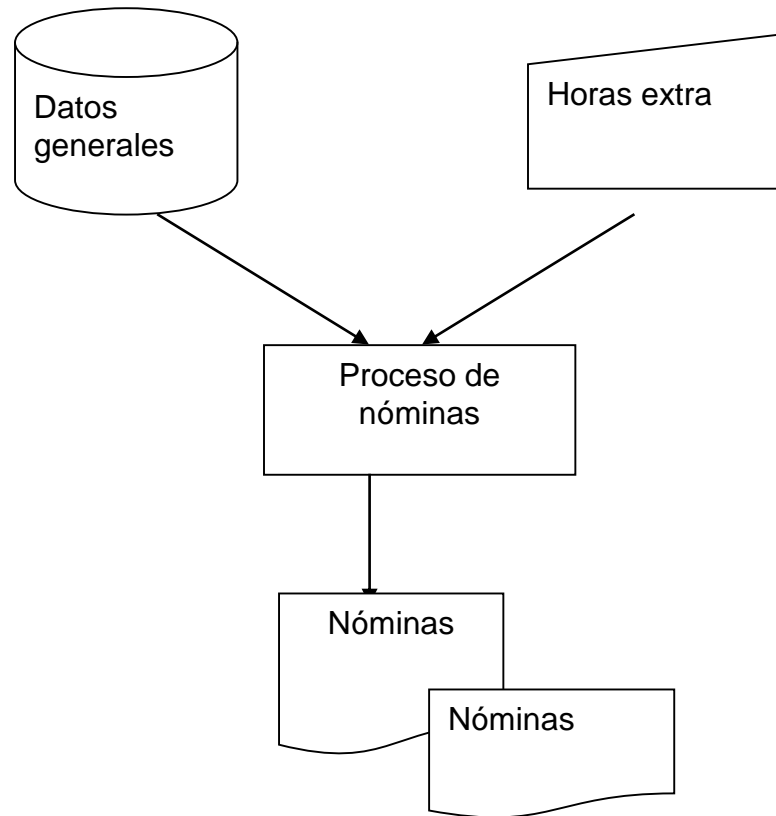
Organigrama:

Representa el flujo de datos y soportes que intervienen en un sistema



4.- Representación de algoritmos

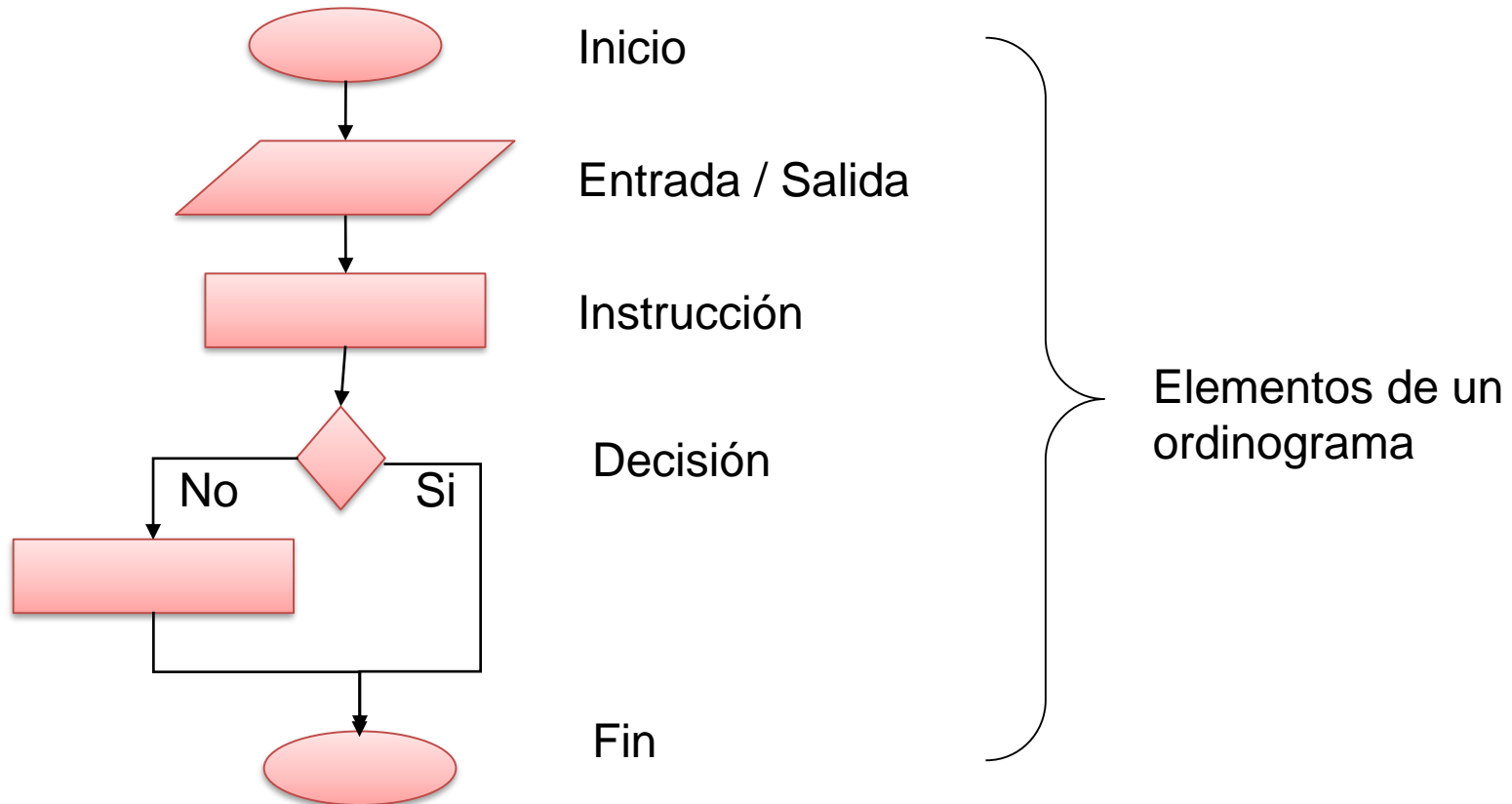
Organigrama (ejemplo):



4.- Representación de algoritmos

Ordinograma (Diagrama de flujo):

Representa el flujo de datos de un proceso

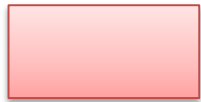


4.- Representación de algoritmos

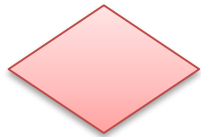
Ordinograma. Elementos



Terminal. Representa el inicio y fin de un programa



Proceso. Acciones del programa



Decisión. Indica operaciones lógicas o de comparación, así como expresiones



Entrada/Salida. Nos permite introducir datos en un periférico, así como mostrarlos

4.- Representación de algoritmos

Ordinograma. Elementos



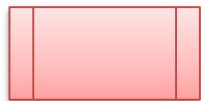
Salida. Usado para mostrar datos por impresora.



Conector. Se coloca al principio y fin de un pedazo de programa, enlaza dos partes de un programa.



Linea de flujo. Indica la dirección de ejecución del algoritmo.



Subprograma. Usado para realizar una llamada a un subprograma o subrutina.

4.- Representación de algoritmos

Diagrama de flujo

Características

- Inicio y Fin
- Arriba abajo, izquierda derecha
- Un símbolo por acción
- A todos los procesos debe llegar una flecha y salir otra.
- Simetría en los elementos

4.- Representación de algoritmos

Pseudocódigo

- Lenguaje natural
- Permite escribir las instrucciones que conducen a la resolución de problema utilizando estructuras básicas de programación
- Reglas
 - Cada instrucción en una línea
 - Conjunto de palabras reservadas en minúsculas: si, entonces, fsi, mientras, fmientras, etc ...
 - Referencia a módulos entre <NOMBRE-MODULO>
 - Código indentado

4.- Representación de algoritmos

Pseudocódigo. Estructura

Programa: NOMBRE correspondiente al programa
Entorno:
 Declaración de las estructuras de datos en general.
Algoritmo:
 Secuencia de instrucciones que forman el programa.
Fin del programa.

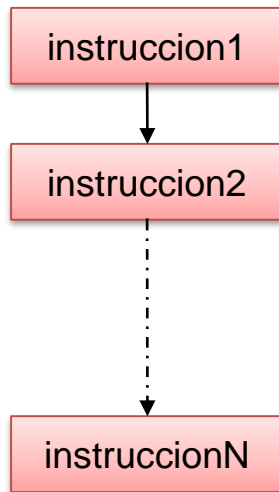
Ejemplo



Programa: ARRANCA_COCHE
Entorno:
Algoritmo:
 Pisar embrague con pie izquierdo
 Poner punto muerto
 Dar a llave de contacto
 Pisar embrague
 Meter la marcha primera
 Quitar el freno de mano
 Levantar el pie del embrague
Fin del programa

4.- Representación de algoritmos

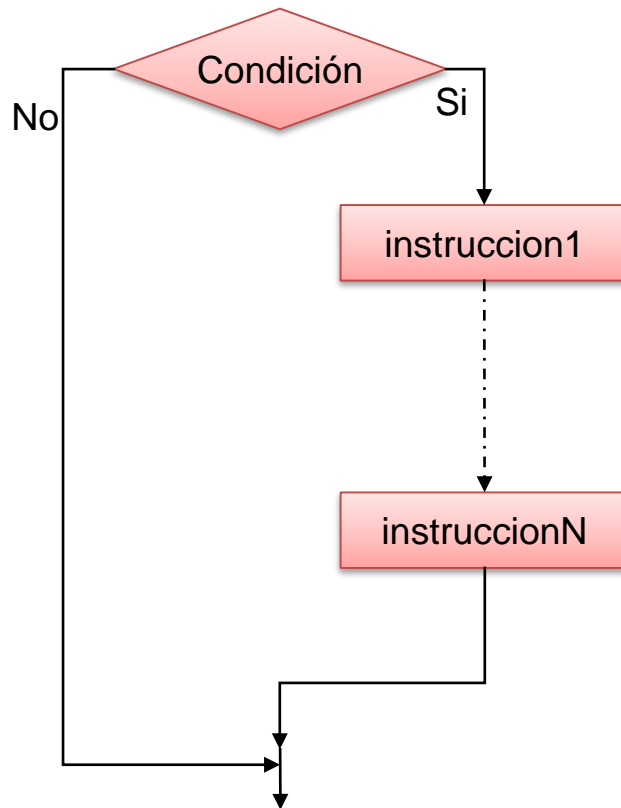
Estructura secuencial



```
instruccion1;  
instruccion2;  
instruccion3;  
....  
instruccionN;
```


4.- Representación de algoritmos

Estructura Condicional Simple



si condición **entonces**

instruccion1;

instruccion2;

instruccion3;

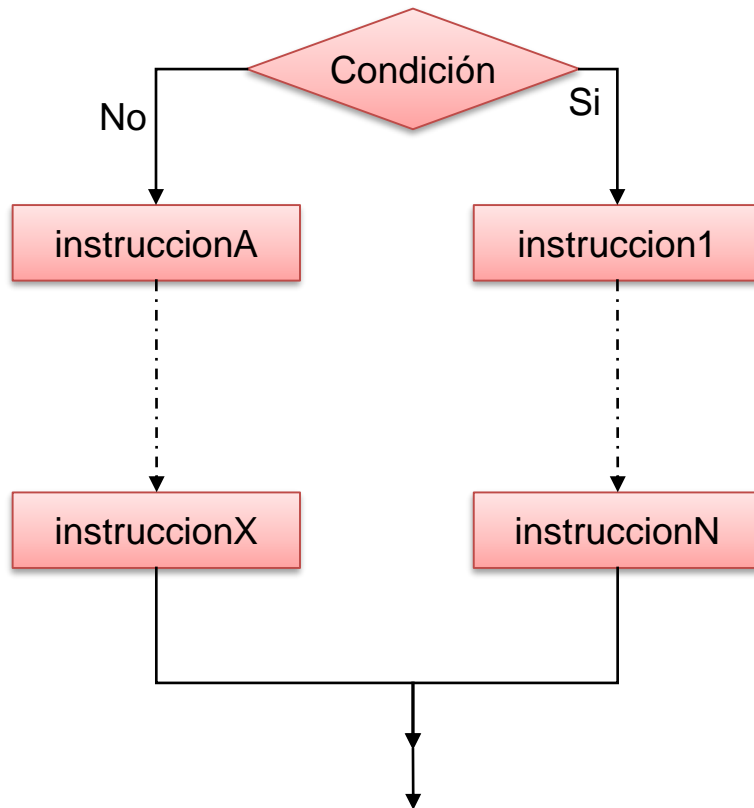
....

instruccionN;

fsi

4.- Representación de algoritmos

Estructura Condicional Compuesta



si condición **entonces**

instruccion1;

....

instruccionN;

sino

instruccionA;

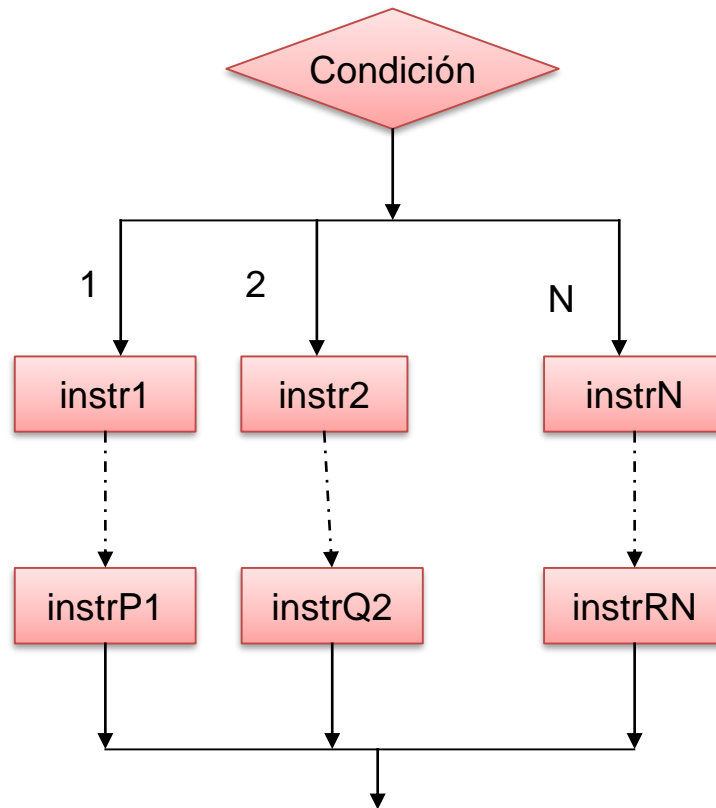
....

instruccionX;

fsi

4.- Representación de algoritmos

Estructura Condicional Múltiple



caso condición **de**

1: instr1;

....

instrP1;

2: instr2;

....

instrQ2;

....

N: instrN;

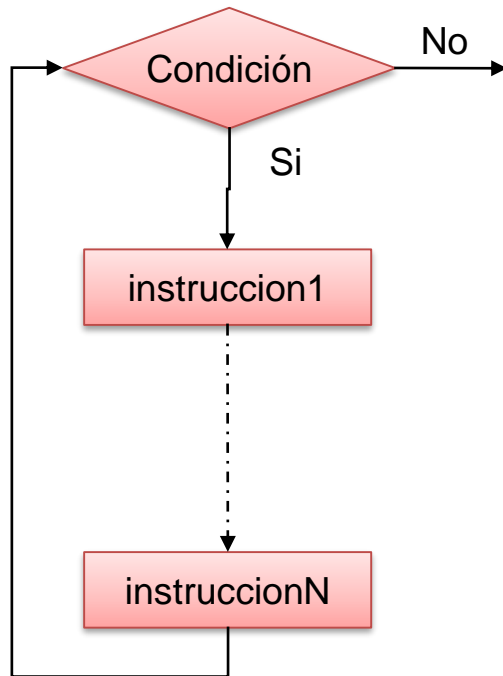
....

instrRN;

fcaso

4.- Representación de algoritmos

Estructura repetitiva Mientras



mientras condición **hacer**

instruccion1;

instruccion2;

instruccion3;

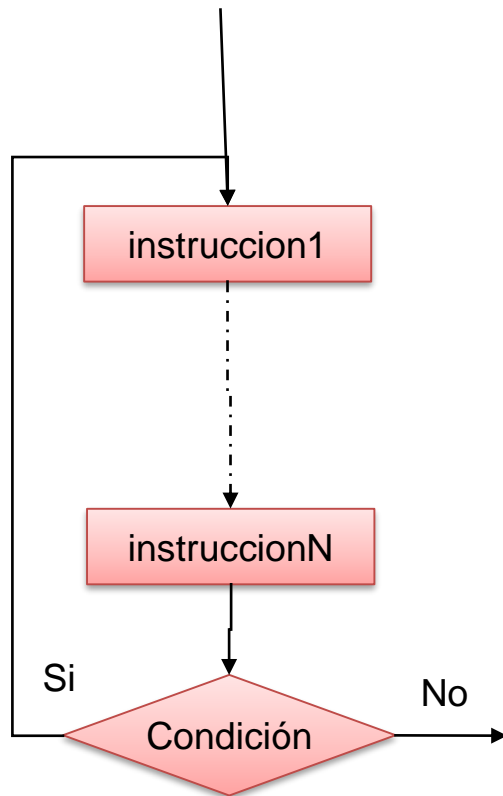
....

instruccionN;

fmientras

4.- Representación de algoritmos

Estructura repetitiva Repite... mientras



repetir

instruccion1;

instruccion2;

instruccion3;

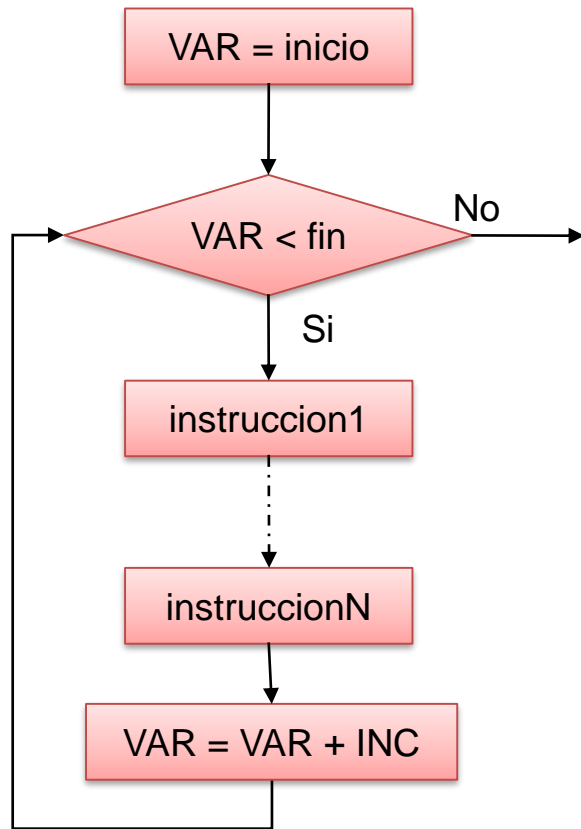
....

instruccionN;

mientras condición;

4.- Representación de algoritmos

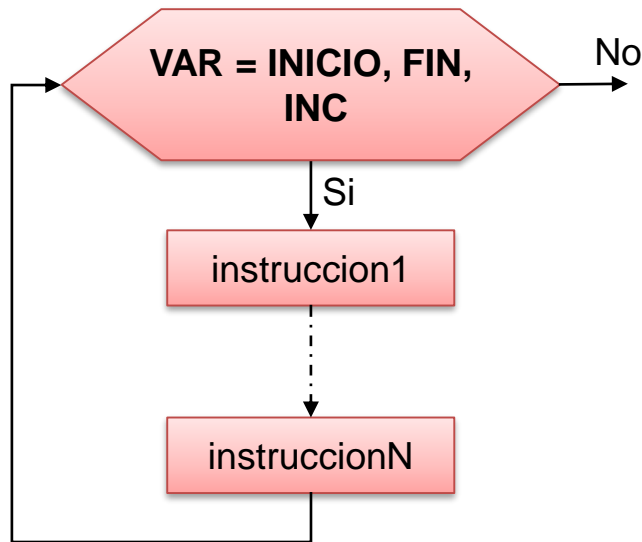
Estructura repetitiva Para (1)



para VAR **desde** inicio **hasta** fin
[con incremento inc] hacer
 instruccion1;
 instruccion2;
 instruccion3;
 ...
 instruccionN;
fpara

4.- Representación de algoritmos

Estructura repetitiva Para (2)



para VAR **desde** inicio **hasta** fin
[con incremento inc] hacer
instruccion1;
instruccion2;
instruccion3;
....
instruccionN;
fpara

4.- Representación de algoritmos

Contadores, acumuladores e interruptores

Vamos a ver una serie de variables especiales usadas en estructuras de tipo repetitivo bien para evaluar una condición o bien de ayuda para realizar acciones.

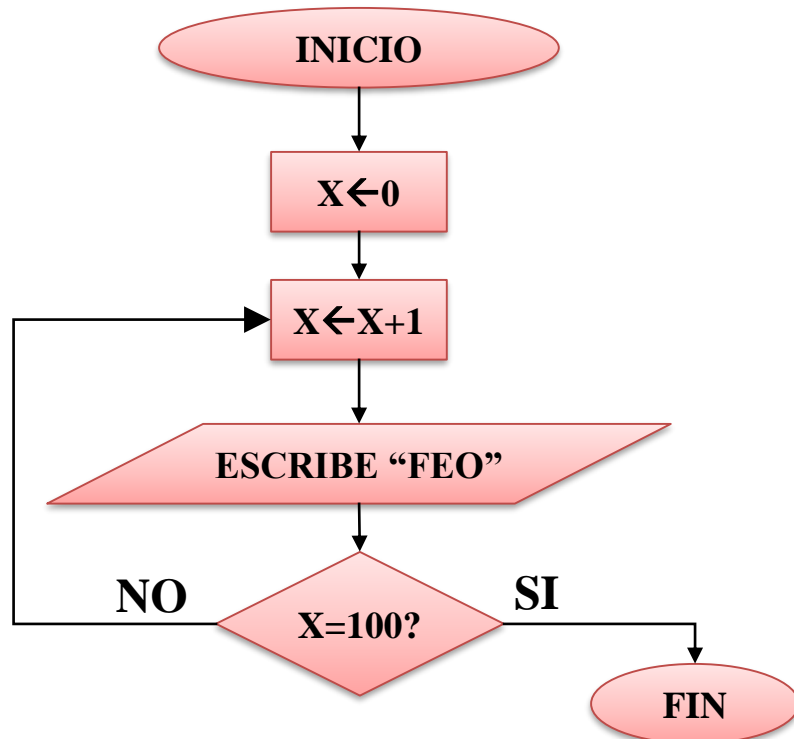
Dichas variables son las siguientes:

- Contadores
- Acumuladores
- Interruptores

4.- Representación de algoritmos

Contadores

Un contador es una variable cuyo valor se incrementa o decrementa en cada iteración de un bucle, sirve para averiguar cual es el estado de un bucle.



PSEUDOCÓDIGO:

INICIO

X ← 0

REPETIR

X ← X + 1

ESCRIBE("FEO")

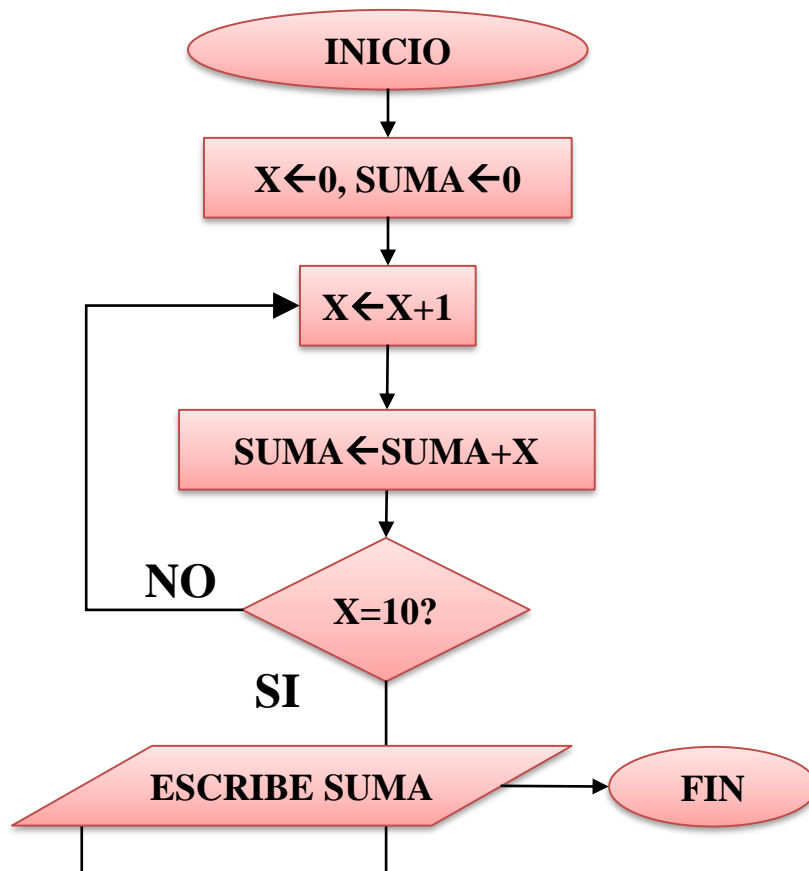
HASTA (X=100)

FIN

4.- Representación de algoritmos

Acumuladores

Parecido a un contador, pero su uso es almacenar cantidades procedentes de acciones en cada iteración.



PSEUDOCÓDIGO:

INICIO

X ← 0

SUMA ← 0

REPETIR

X ← X + 1

SUMA ← SUMA + X

HASTA (X = 10)

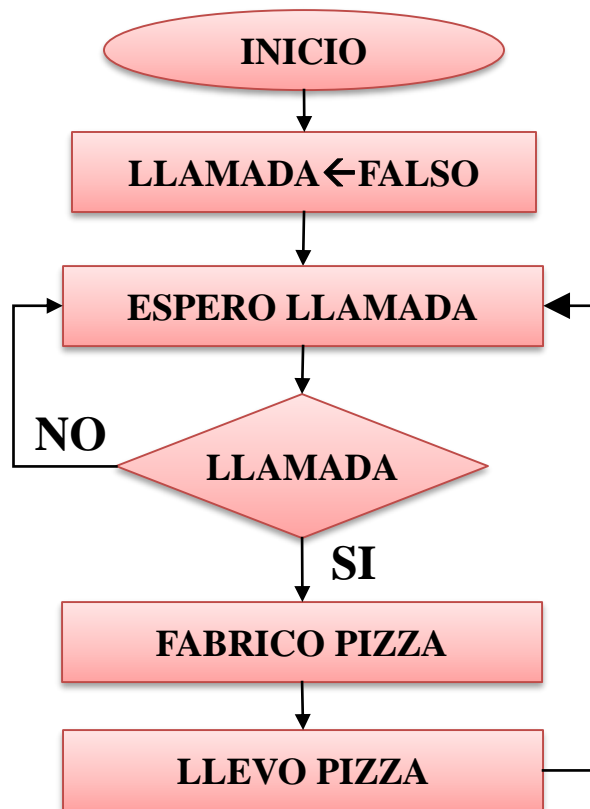
ESCRIBE(SUMA)

FIN

4.- Representación de algoritmos

Interruptores

Un interruptor es una variable cuyo valor puede tomar los valores verdadero o falso. Se usa para control de bucles y estructuras selectivas.



PSEUDOCÓDIGO:

```
INICIO
LLAMADA ← FALSO
REPETIR
    ESPERO LLAMADA
    SI LLAMADA ENTONCES
        FABRICO PIZZA
        LLEVO PIZZA
    FIN_SI
HASTA EL JUICIO FINAL
```

Bibliografía

Bibliografía

- ✓ Aprende JAVA con ejercicios. Edición 2018. Luis José Sánchez.
- ✓ Empezar a programar usando Java. 2ª edición. Universitat Politècnica de València
- ✓ Apuntes de la asignatura Ingeniería del Software de la Universitat Politècnica de València.
- ✓ <https://github.com/statickidz/TemarioDAW>
- ✓ <https://es.khanacademy.org/computing/computer-science/algorithms>
- ✓ https://es.wikipedia.org/wiki/Lenguaje_de_programación
- ✓ https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado