La Sénia
Institut d'Educació Secundària
Paiporta

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Algoritmos

Curso 2023 - 2024

1. Introducción a los Algoritmos

Algoritmo: Secuencia **finita** de **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 que sirve para desarrollar un programa usando una herramienta que permite escribir el código (un lenguaje de programación) y otra que sea capaz de “traducirlo” al lenguaje de máquina (interprete o compilador) para ser entendido por un microprocesador.

PROGRAMA = ALGORITMOS + ESTRUCTURAS DE DATOS

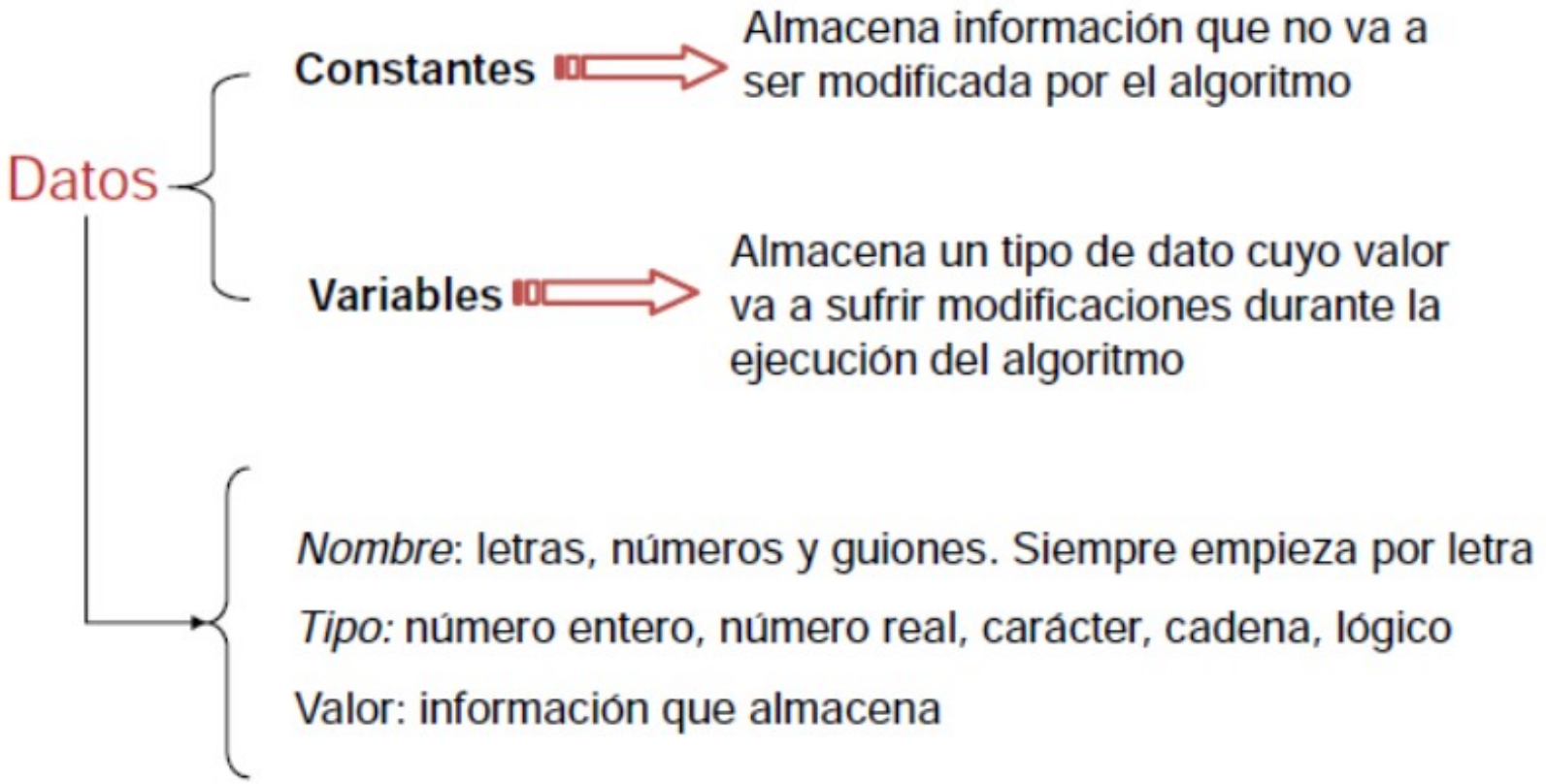
1. Introducción a los Algoritmos

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. Introducción a los Algoritmos



1. Introducción a los Algoritmos

Ejemplo de un algoritmo para la reparación de un PC:

1. Recibir un ordenador estropeado
2. Comenzar a contar el tiempo
3. Comprobar el PC y detectar averías
4. Anotar piezas cambiadas
5. Anotar tiempo empleado
6. Preguntar forma de pago (efectivo o tarjeta [2% recargo])
7. Facturar (Cobrar por el tiempo trabajado y las piezas cambiadas)

1. Introducción a los Algoritmos

En el algoritmo del ejemplo, se necesitan:

1. Una variable para almacenar el tiempo de reparación, tipo numérico entero
2. Una constante con el precio por hora
3. Una variable para almacenar el precio de los componentes, tipo numérico real
4. Una variable para el pago con tarjeta o no, tipo lógico de verdadero o falso
5. Variable para almacenar el importe total, tipo numérico real

1. Introducción a los Algoritmos

Podemos extraer las siguientes características de los algoritmos:

- **Preciso**: debe indicar el ***orden*** en el que se realiza cada uno de los pasos
- **Definido**: si se sigue el algoritmo varias veces, se debe obtener el ***mismo resultado*** cada vez
- **Finito**: debe ***terminar*** en un número finito de pasos

2. Representación de Algoritmos

Los algoritmos se pueden representar de diversas formas:

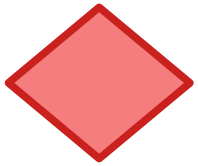
- **Modo gráfico**: Mediante un ***Diagrama de Flujo*** que refleja el proceso que se aplica a los datos de entrada para obtener los datos de salida
- **Modo texto**: Mediante ***Pseudocódigo*** que muestra los pasos del algoritmo en un lenguaje natural pero cercano a la codificación final del algoritmo

2. Representación de Algoritmos

Diagramas de flujo. Representa el flujo de los datos en un proceso mediante los siguientes elementos:

Terminal: indica inicio y fin del programa

Proceso: acciones del programa



Decisión: indica operaciones lógicas o de comparación

Entrada/Salida: permite introducir o mostrar datos

Iteración: grupo de características de un bucle

Entrada Manual: introducir los datos por teclado

Documento: indica la salida de datos de forma impresa

Pantalla: Presentación de resultados por pantalla

Línea de Flujo: indica la dirección en la que circulan los datos y se procesan

Conector: Enlaza dos partes del diagrama en páginas distintas

2. Representación de Algoritmos

Diagramas de flujo. Características:

- Se lee de arriba hacia abajo (vertical) o de izquierda a derecha (horizontal)
- Debe tener un terminador de inicio y otro de fin
- Se utiliza un símbolo por acción
- Debe llegar una flecha a todos los procesos y de ellos salir otra

2. Representación de Algoritmos

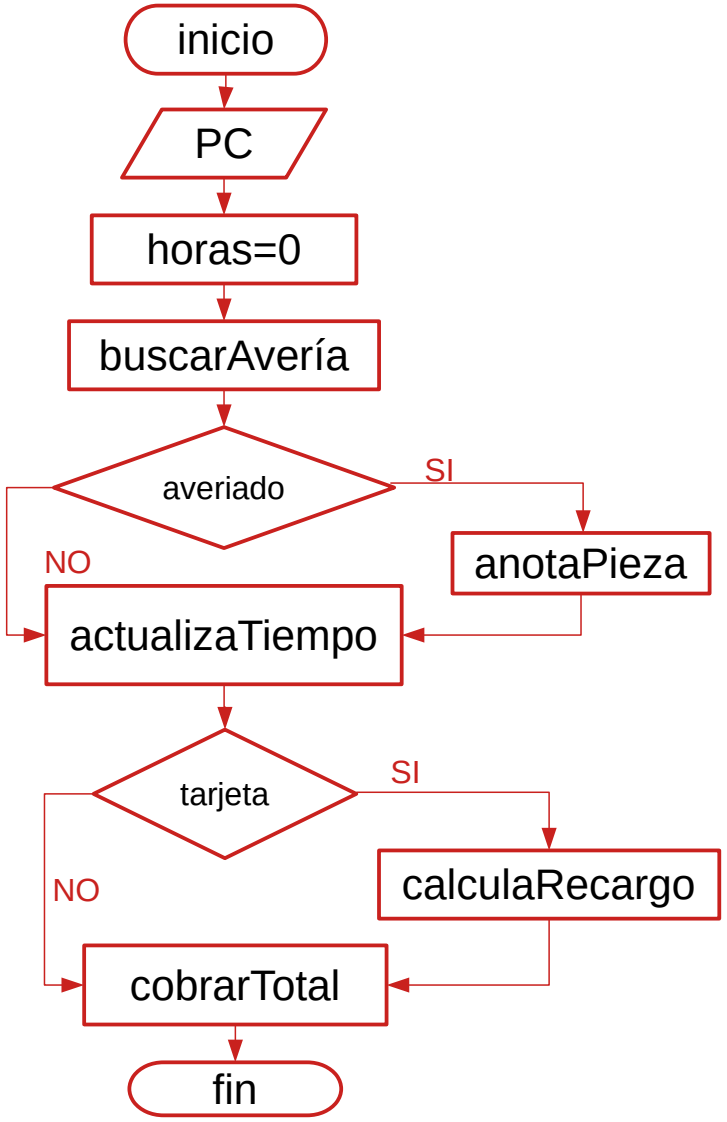
Estrategia para realizar un **Diagrama de Flujo**:

1) Reúne información e identifica qué proceso deseas definir.

- Determina los puntos de inicio y de fin.
- Define las tareas intermedias (para procesar)

2) Utilizar los símbolos anteriores para representar las tareas, datos de entrada, de salida, flujo de información, etc.

2. Representación de Algoritmos (DF)



Reparación de un PC:

1. Recibir un ordenador estropeado
2. Comenzar a contar el tiempo
3. Comprobar el PC y detectar averías
4. Anotar piezas cambiadas
5. Anotar tiempo empleado
6. Preguntar si paga en efectivo o con tarjeta (recargo del 2%)
7. Cobrar el tiempo trabajado por horas y las piezas cambiadas, así como el recargo si lo hay

2. Representación de Algoritmos

Pseudocódigo. Características:

Permite describir un algoritmo en un lenguaje de alto nivel independiente del lenguaje de programación.

- Utiliza un lenguaje natural, entendible incluso por no informáticos
- Usa instrucciones para resolver el problema usando estructuras básicas de programación
- Reglas:
    - Cada instrucción ocupa una línea completa
    - Tiene palabras reservadas: si, entonces, fsi, mientras, fmientras, etc.
    - Referencia a módulos entre <NOMBRE-MODULO>
    - El código debe estar **indentado**

2. Representación de Algoritmos

Pseudocódigo.

- **Programa**: NOMBRE correspondiente al programa o módulo
- **Entorno**:

  Declaración de las estructuras de datos en general (bloque de declaraciones)
- **Algoritmo**:
  - *Inicio del programa*
  - Secuencia de instrucciones que permiten la ejecución del programa (bloque de instrucciones)
  - *Fin del programa*

2. Representación de Algoritmos (PS)

Programa: REPARA_PC

Entorno:

entero horas, tiempo, RECARGO

real PVP_PIEZA, total

Algorimo:

```
inicio
    Recibir PC
    horas=0
    Comprobar_averías
    piezas_cambiadas
    Calcular_tiempo
    si paga_tarjeta entonces
        calcular_recargo
    fsi
    Calcular total
fin
```

Reparación de un PC:

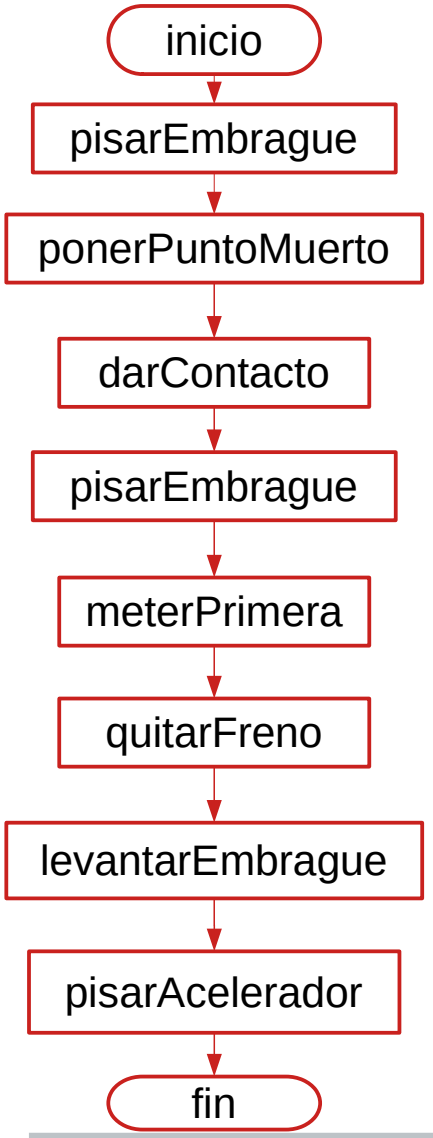
1. Recibir un ordenador estropeado
2. Comenzar a contar el tiempo
3. Comprobar el PC y detectar averías
4. Anotar piezas cambiadas
5. Anotar tiempo empleado
6. Preguntar si paga en efectivo o con tarjeta (recargo del 2%)
7. Cobrar el tiempo trabajado por horas y las piezas cambiadas, así como el recargo si lo hay

2. Ejemplo

Realiza el Diagrama de Flujo y Pseudocódigo de un algoritmo para poner en marcha el coche (arranca coche). Acciones a realizar:

1. Pisar el embrague con el pie izquierdo
2. Poner punto muerto
3. Dar al contacto
4. Pisar embrague
5. Meter la marcha Primera
6. Quitar el freno de mano
7. Levantar el pie del embrague
8. Pisar el acelerador con el pie derecho

2. Ejemplo



Programa: ARRANCA_COCHE
Entorno:

Algorimo:
inicio
  Pisar el embrague con el pie izquierdo
  Poner punto muerto
  Dar al contacto
  Pisar embrague
  Meter la marcha Primera
  Quitar el freno de mano
  Levantar el pie del embrague
  Pisar el acelerador con el pie derecho
fin

3. Estructuras de control

Las **estructuras de control** permiten modificar el flujo de ejecución de las instrucciones.

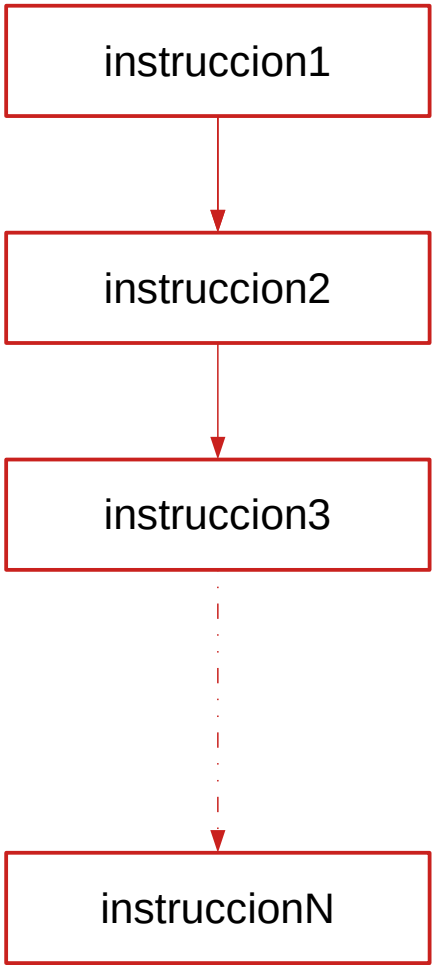
Permiten que no se ejecuten de forma secuencial una tras otra, adaptando el algoritmo a la realidad.

Todas las estructuras de control tienen **un único punto de entrada**.

Se clasifican en:

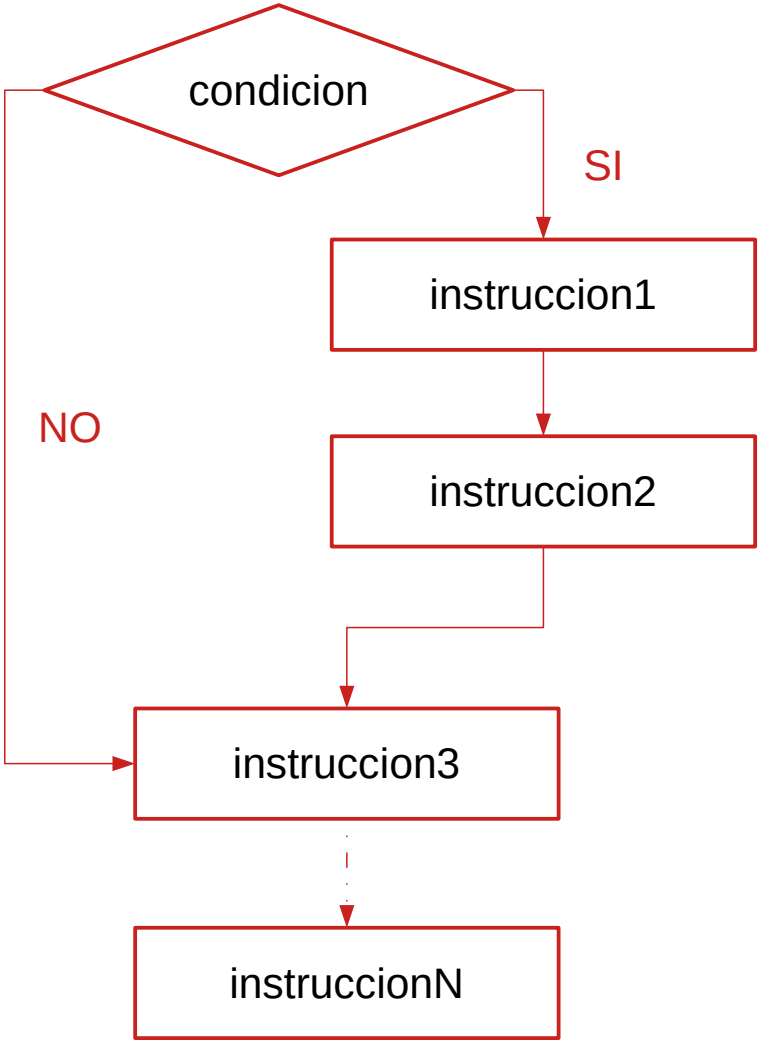
- **Secuenciales**: las instrucciones se ejecutan una tras otra
- **Alternativa o selección**: se ejecutan unas acciones u otras dependiendo del si se cumple o no una **condición**
- **Repetitiva o iterativa**: Se ejecutan las mismas acciones mientras se cumple una condición o un número determinado de veces.

4. Estructura secuencial




```
instruccion1;

instruccion2;

instruccion3;

...

instruccionN;
```

5. Estructuras de selección



si condicion **entonces**

    instruccion1;

    instruccion2;

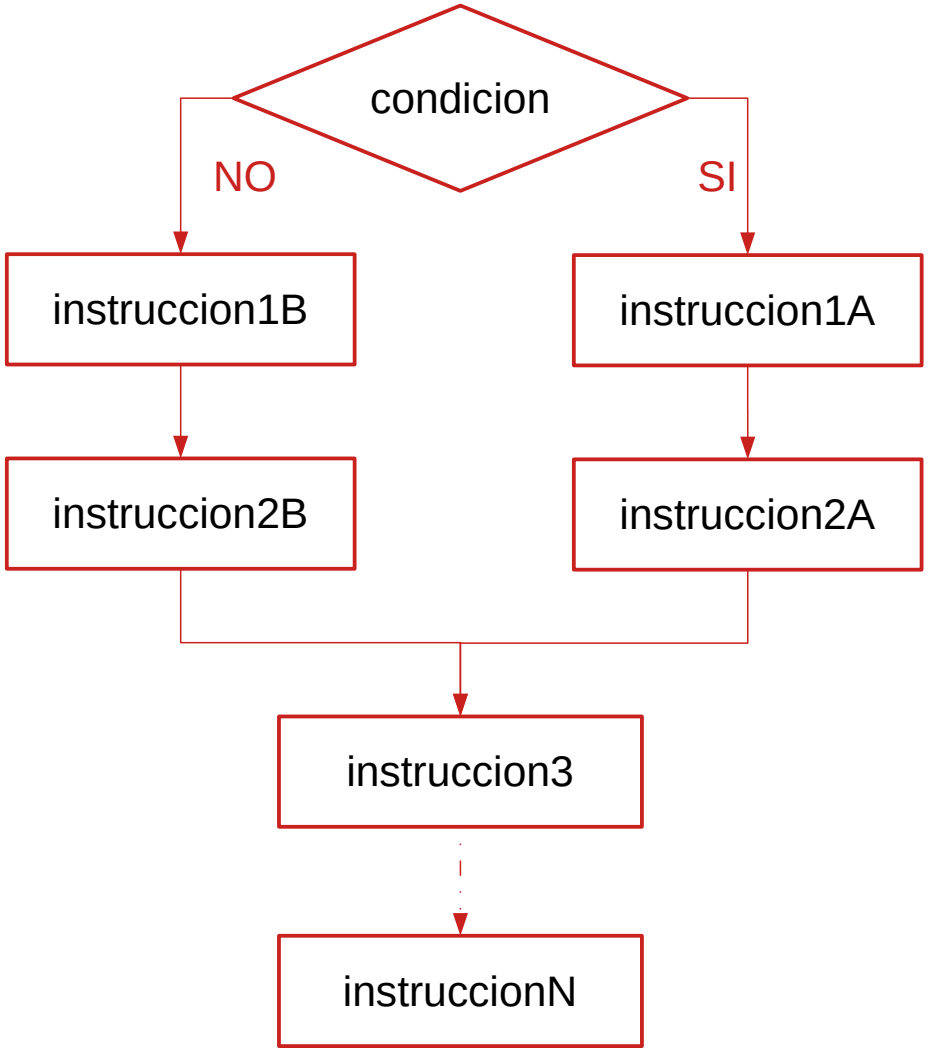
fsi

instruccion3;

...

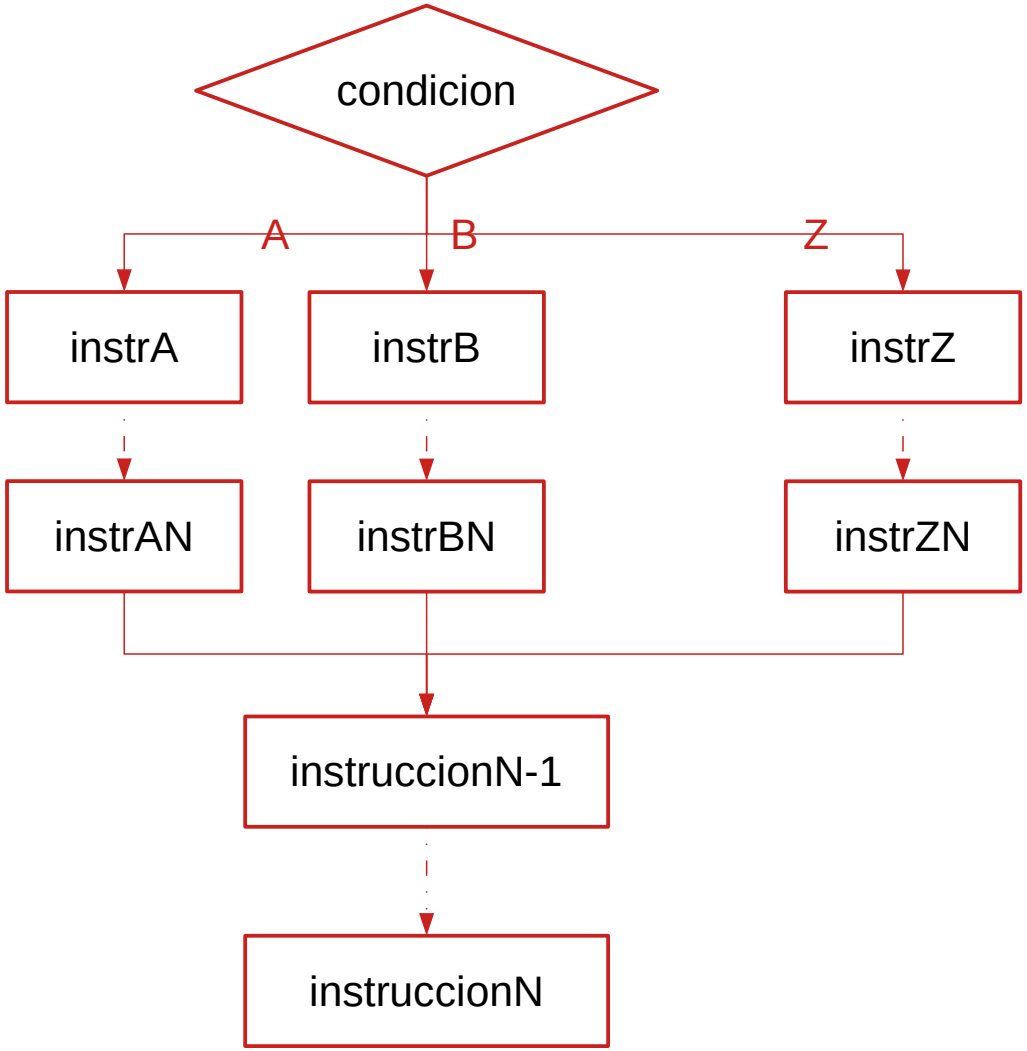
instruccionN;

5. Estructuras de selección



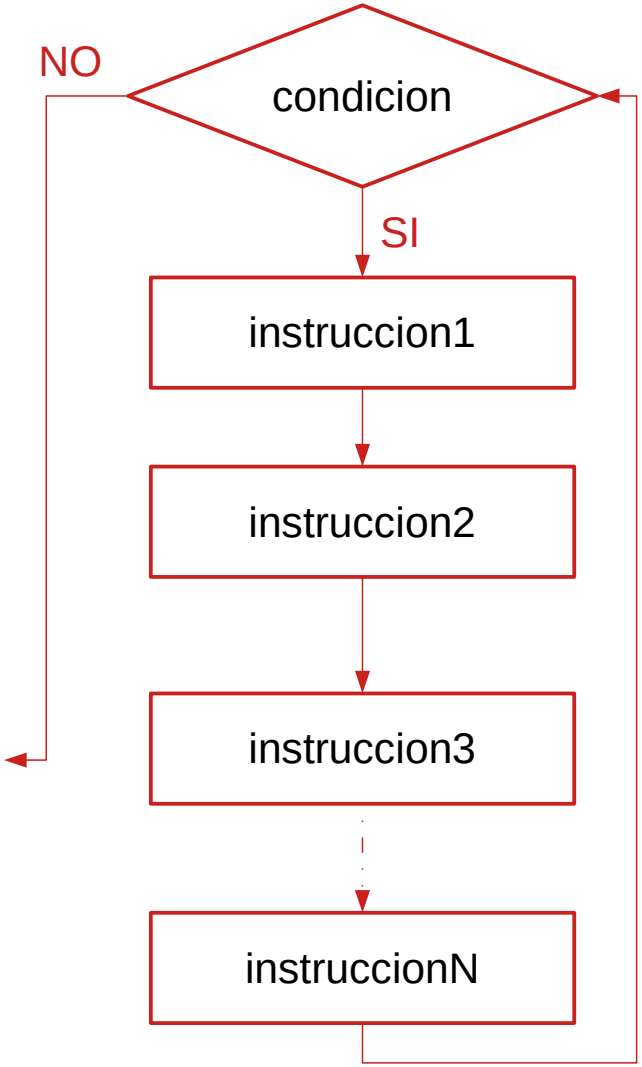

```
si condicion entonces
        instruccion1A;
        instruccion2A;
    sino
        instruccion1B;
        instruccion2B;
fsi
instruccion3;
...
instruccionN;
```

5. Estructuras de selección (multiple)




```
caso condicion de
    A:
        InstrA;
        ...
        instrAN;
    B:
        InstrB;
        ...
        instrBN;
    ...
    Z:
        InstrZ;
        ...
        instrZN;
fcaso
instruccionN-1;
...
instruccionN;
```

6. Estructuras repetitivas (mientras)



mientras condicion **hacer**

instruccion1;

instruccion2;

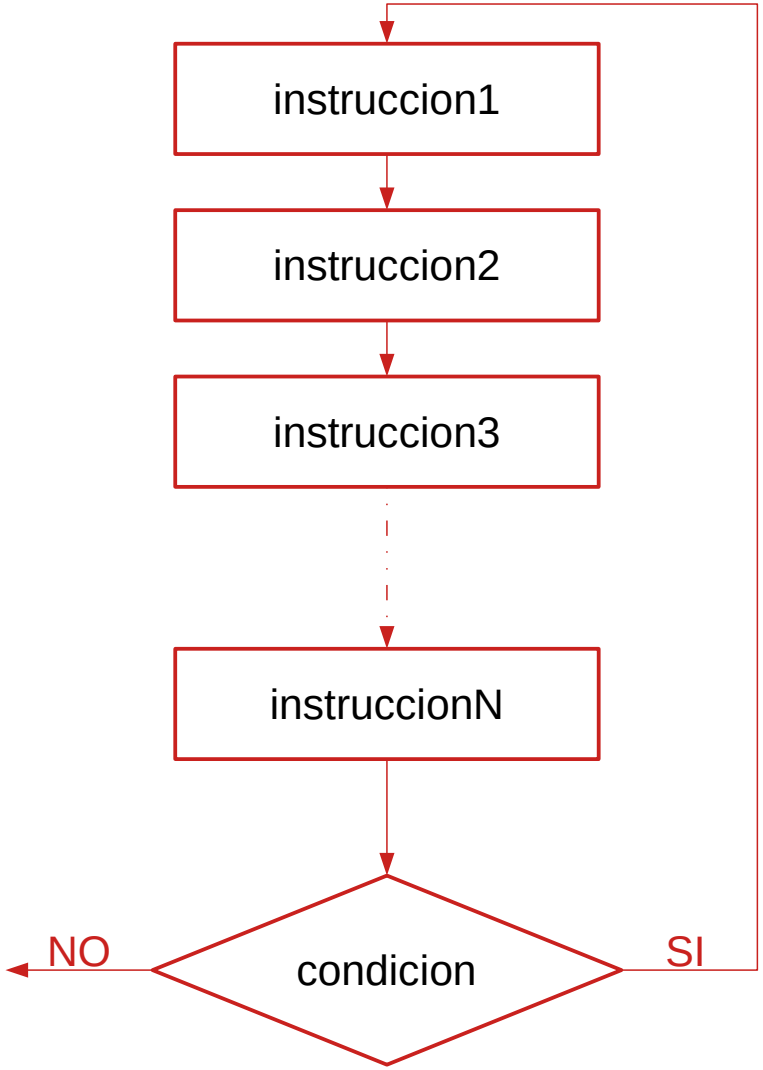
instruccion3;

...

instruccionN;

fmientras

6. Estructuras repetitivas (repetir)



repetir

    instruccion1;

    instruccion2;

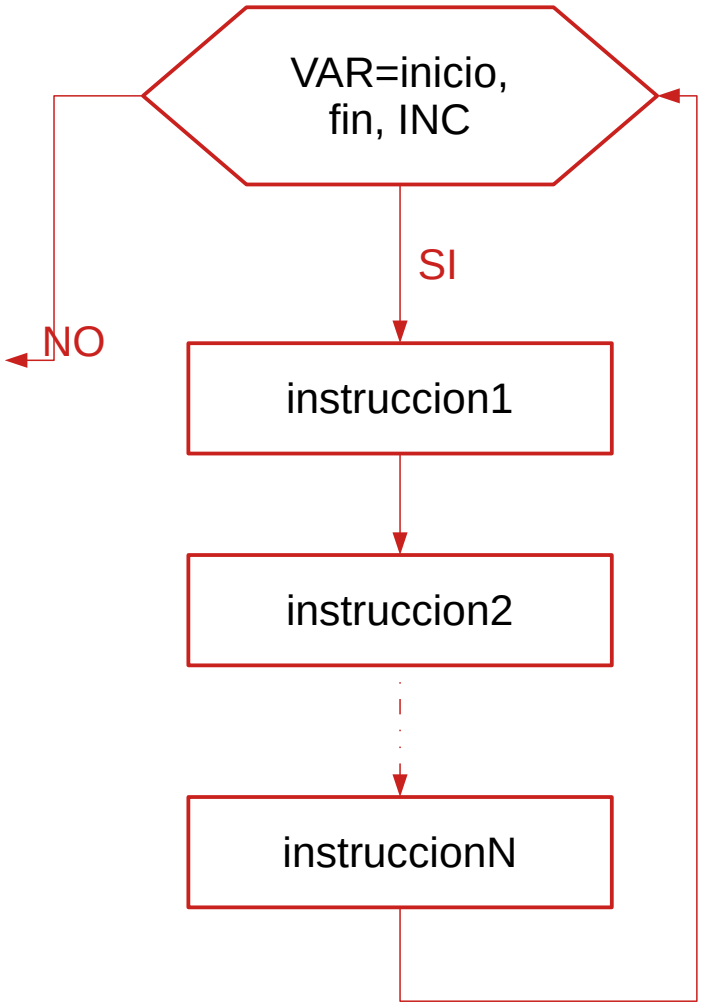
    instruccion3;

    ...

    instruccionN;

mientras condicion

6. Estructuras repetitivas (para)



para VAR **desde** inicio **hasta** fin
[**con incremento** INC] **hacer**

instruccion1;

instruccion2;

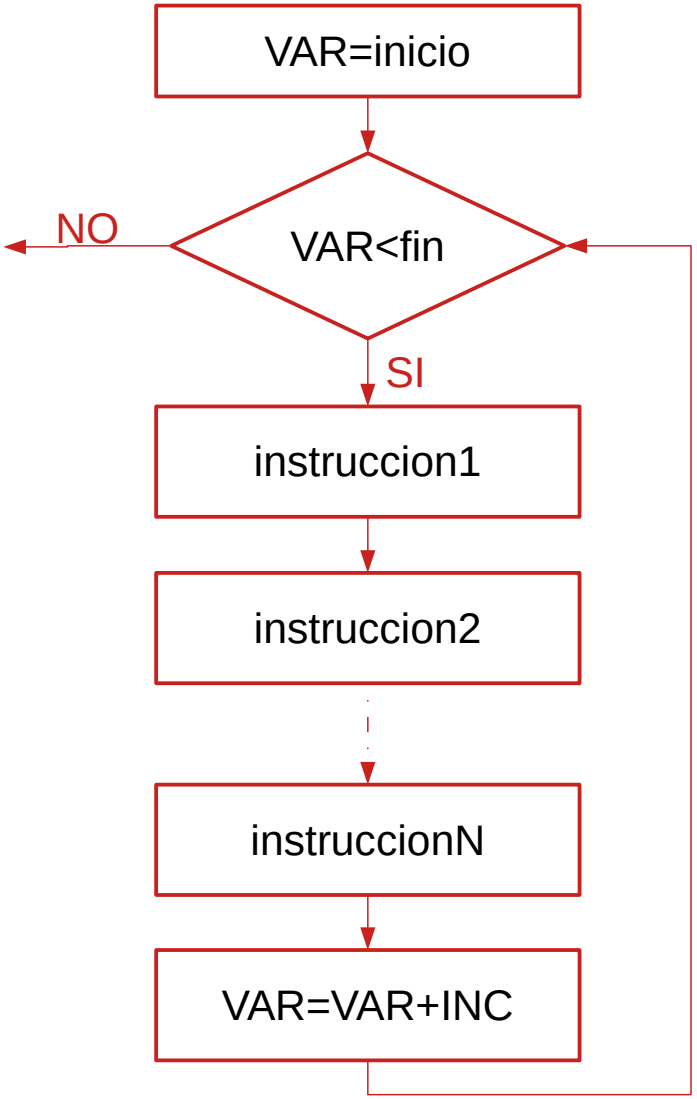
instruccion3;

...

instruccionN;

fpara

6. Estructuras repetitivas (para)



para VAR **desde** inicio **hasta** fin
[**con incremento** INC] **hacer**

instruccion1;

instruccion2;

instruccion3;

...

instruccionN;

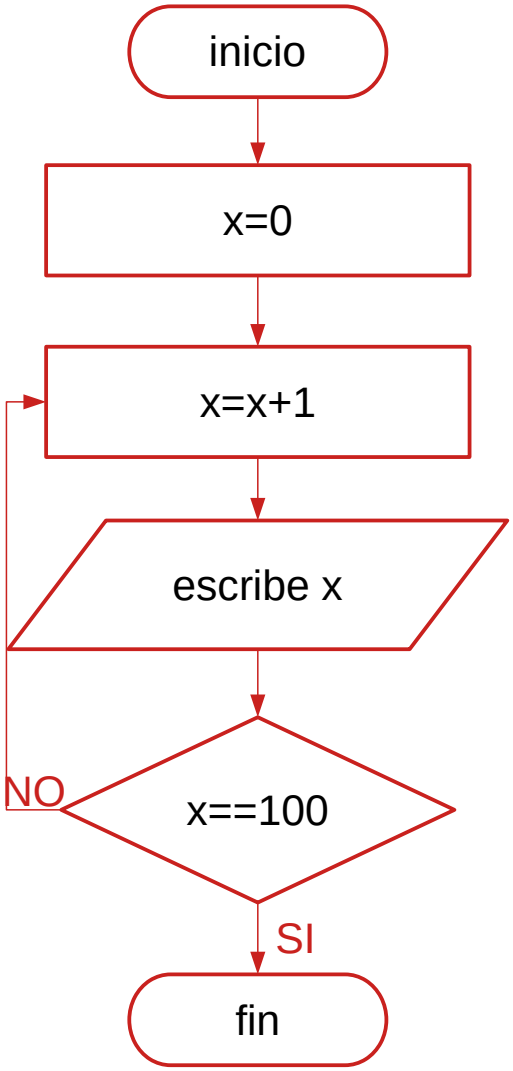
fpara

7. Variables especiales

Las estructuras repetitivas utilizan unas variables especiales con la finalidad de evaluar una condición o ayudar a realizar acciones.

- **Contadores**: una variable ***contador*** se incrementa o decrementa en cada iteración del bucle y permite averiguar su estado.
- **Acumuladores**: Una variable ***acumulador*** almacena cantidades procedentes de acciones de cada iteración; su valor aumenta (o disminuye) con los resultados.
- **Interruptores**: Una variable ***interruptor*** puede tomar los valores verdadero o falso y se utiliza controlar las condiciones en bucles y en estructuras selectivas.

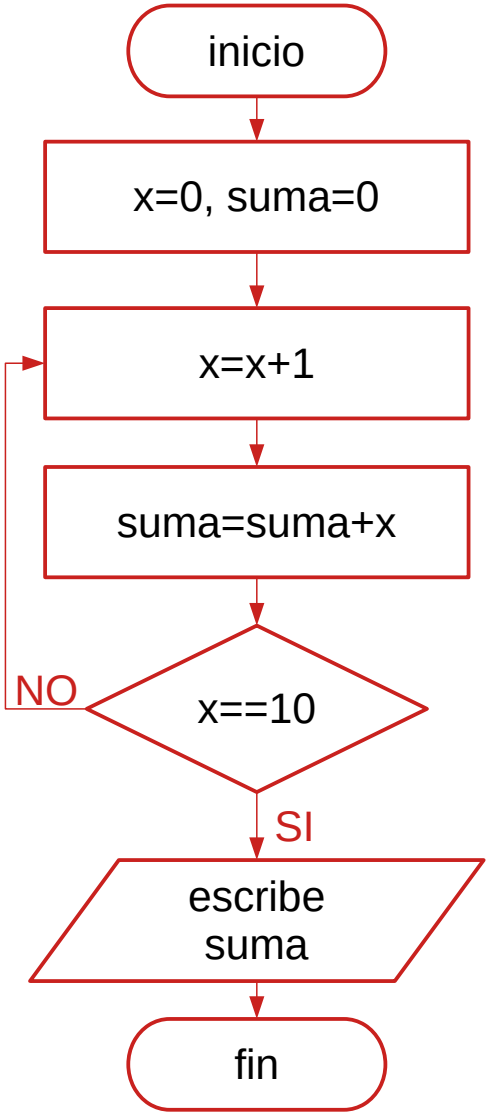
7. Variables especiales (contador)




```
inicio
    x=0;
    repetir
        x=x+1;
        escribe x;
    hasta (x==100)
fin
```

x es una variable ***contador*** que indica el número de veces que hemos ejecutado el bucle

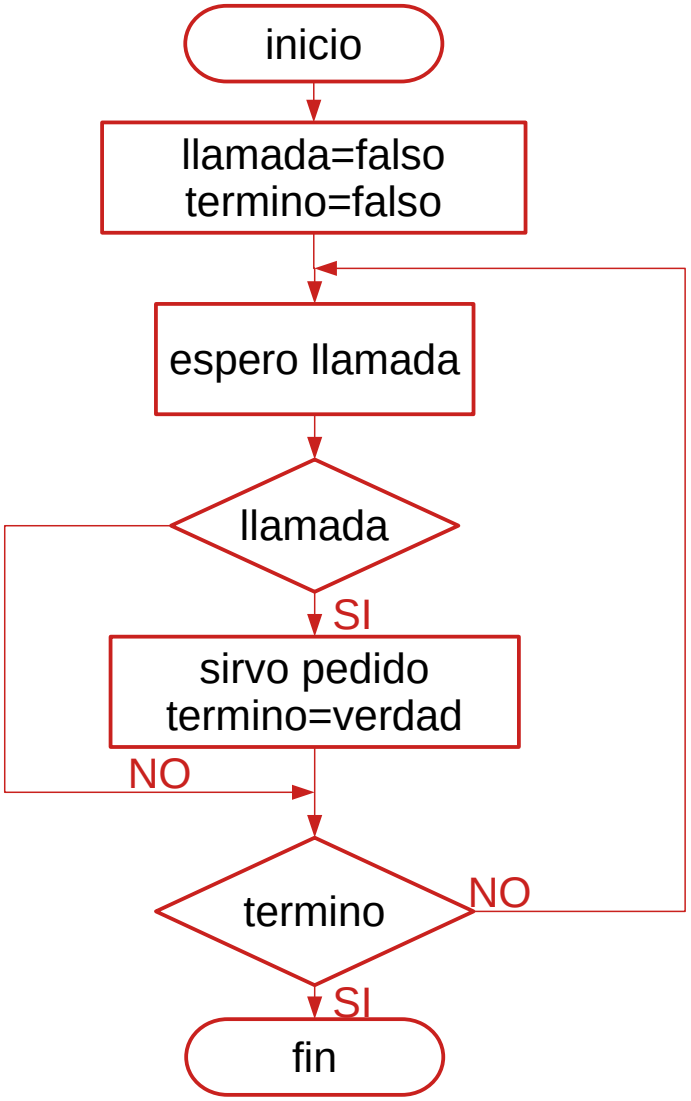
7. Variables especiales (acumulador)




```
inicio
   x=0;
   suma=0;
   repetir
      x=x+1;
      suma=suma+x;
   hasta (x==10)
   escribe x;
fin
```

suma es una variable ***acumulador*** que almacena la suma de los valores del contador x cada vez que ejecutamos el bucle

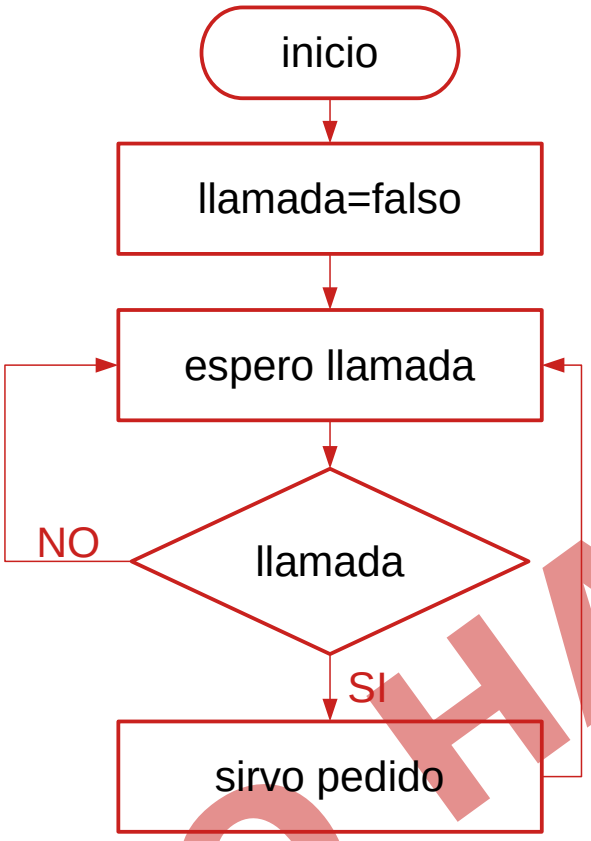
7. Variables especiales (interruptor)




```
inicio
    termino=false;
    llamada=false;
    repetir
        si llamada entonces
            sirvo pedido;
            termino = true;
        fsi;
    hasta termino (condición de fin)
fin
```

La variable ***interruptor*** **termino** indica la condición de fin e interrumpe la ejecución del bucle

7. Variables especiales (interruptor)

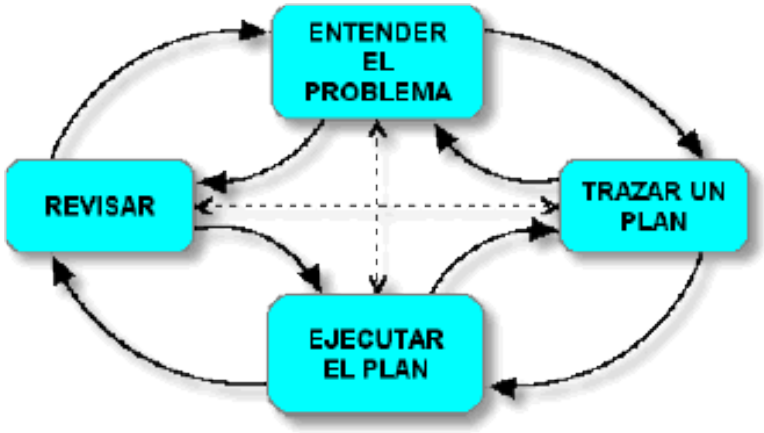


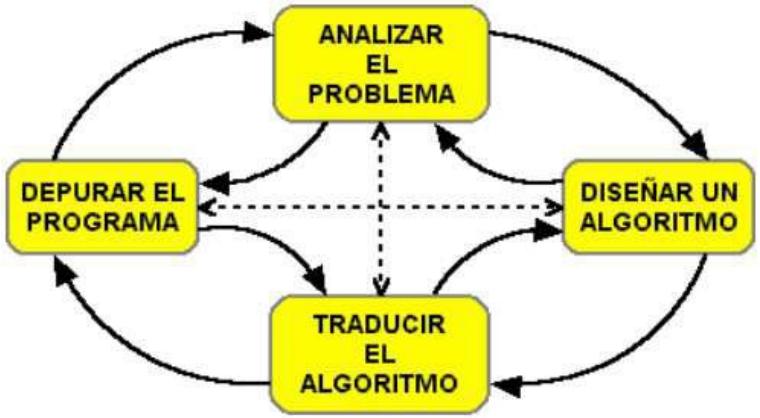

```
inicio
    llamada=false;
    repetir
        si llamada entonces
            sirvo pedido;
        fsi;
    hasta infinito (Queeeeeee!!!!!!!)
fin
```

llamada es una variable ***interruptor*** que define la condición a cumplir para servir pedidos o esperar hasta el infinito

8. Resolución de Algoritmos

Según el método de George Poyla, se aplican 4 etapas o pasos:





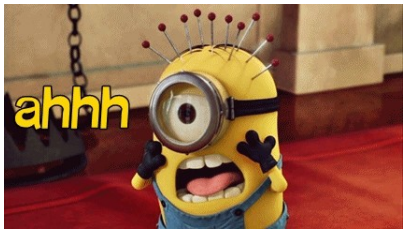
1) **Entender** el problema:
   - ***análisis*** del problema
   - crucial la comprensión lectora y/u oral
2) **Trazar** un plan:
   - ***diseño*** de los procesos, datos y operaciones necesarias
   - Diagrama Flujo
3) **Ejecutar** un plan:
   - **traducir** diseño a una ***codificación*** (pseudocodigo)
4) **Revisar**:
   - ***depurar*** errores
   - comprobar resultados

8. Resolución de Algoritmos (ejemplo)

Disponemos de un juego al que se juega por rondas.

En cada ronda, el ganador obtiene una ficha roja; el segundo, una ficha azul; y el tercero, una amarilla.

Al final de varias rondas, la puntación se calcula de la siguiente manera: Al cubo de la cantidad de fichas rojas, se suma el doble de fichas azules y se descuenta el cuadrado de las fichas amarillas.



Si Andrés llegó 3 veces en primer lugar, 4 veces de último y 6 veces de intermedio, ¿Qué puntuación obtuvo?

8. Resolución de Algoritmos (Análisis)

*En cada ronda, el ganador obtiene una ficha **roja**; el segundo, una ficha **azul**; y el tercero, una **amarilla**.*

*Al final de varias rondas, la **puntación** se calcula de la siguiente manera: Al cubo de la cantidad de fichas rojas, se suma el doble de fichas azules y se descuenta el cuadrado de las fichas amarillas.*

Podemos **analizar** los siguientes datos:

- El ***ganador*** obtiene una ficha **roja**; el ***segundo***, una ficha **azul**; y el ***tercero***, una **amarilla**

La **puntuación**:

- El cubo de fichas **rojas**: R^3
- El doble de fichas **azules**: $2Az$
- El cuadrado de fichas **amarillas**: Am^2

La **puntuación** será $R^3 + 2Az - Am^2$

8. Resolución de Algoritmos (Análisis)

Para calcular la puntación hemos determinado la fórmula:

$$(R^3) + (2Az) - (Am^2)$$

Si Andrés llegó 3 veces en primer lugar, 4 veces de último y 6 veces de intermedio...

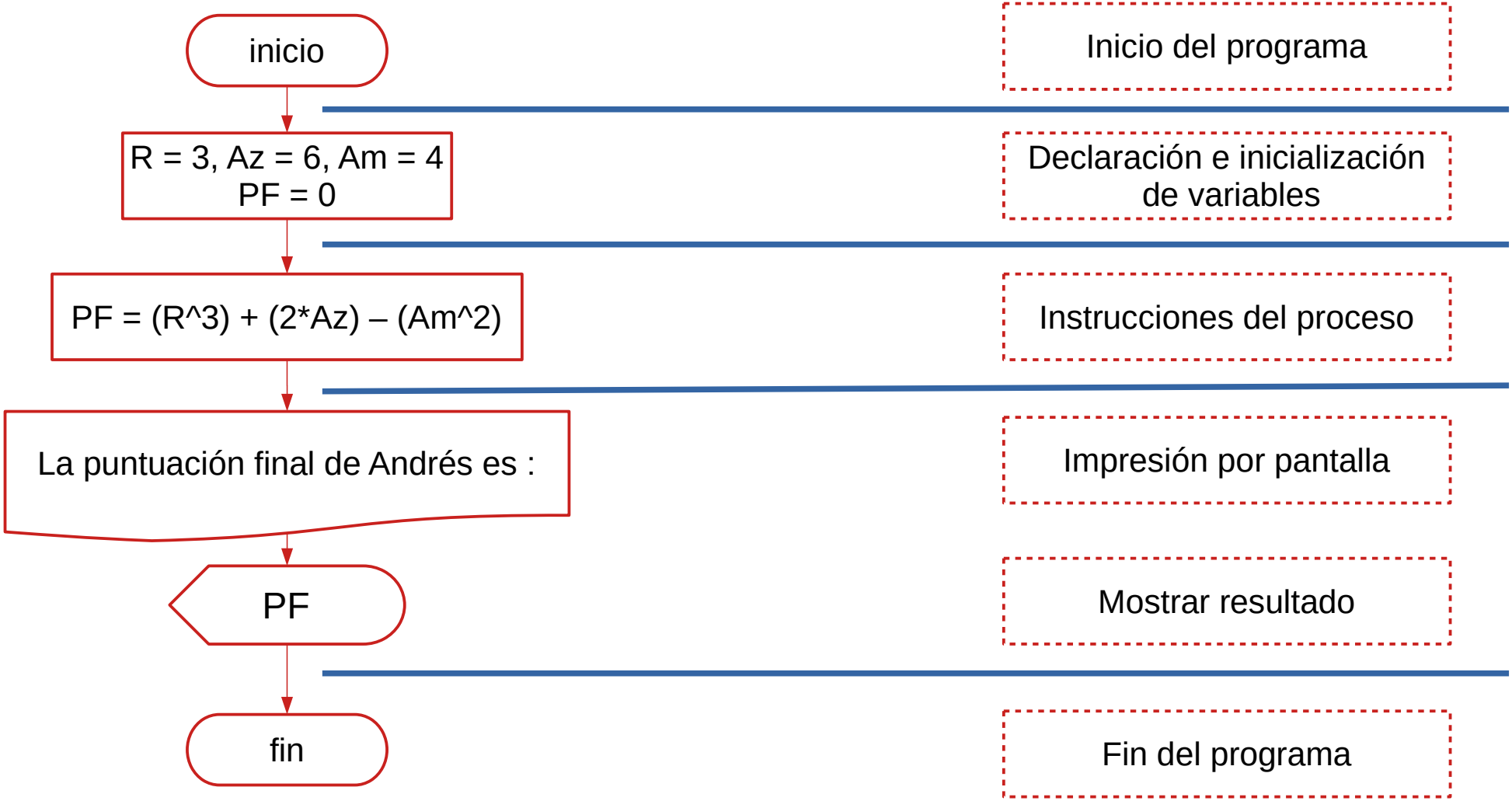
- **Andrés** llega 3 veces el **primero**, 4 veces el **tercero** y 6 veces el **segundo**, por lo que obtiene 3 fichas **rojas**, 6 fichas **azules** y 4 fichas **amarillas**
- Se obtendrá como resultado la Puntación Final (**PF**)

$$PF = (R^3) + (2Az) - (Am^2) = (3^3) + (2*6) - (4^2) = 27 + 12 - 16 = 23$$

- La Puntuación Final obtenida por Andrés es de 23 puntos

8. Resolución de Algoritmos (Diseño)

Diagrama de Flujo para el algoritmo:



8. Resolución de Algoritmos (Traducir)

Del Diagrama de Flujo podemos obtener el Pseudocódigo:

INICIO

```
    //Declaramos las variables y las inicializamos
    ENTERO R=3;
    ENTERO Az=6;
    ENTERO Am=4;
    ENTERO PF=0;
    //Indicamos la operación a realizar con los datos
    PF=(R^3) + (2*Az) – (Am^2)
    //Imprimimos por pantalla el texto y la variable del resultado
    IMPRIMIR “La puntuación final de Andrés es “
    IMPRIMIR PF;
```

FIN

8. Resolución de Algoritmos (Diseño)

Del Diagrama de Flujo podemos obtener el Pseudocódigo:

INICIO

```
    //Declaramos las variables y las inicializamos
    ENTERO R=3;
    ENTERO Az=6;
    ENTERO Am=4;
    ENTERO PF=0;
    //Indicamos la operación a realizar con los datos
    PF=(R^3) + (2*Az) – (Am^2)
    //Imprimimos por pantalla el texto y la variable del resultado
    IMPRIMIR “La puntuación final de Andrés es “
    IMPRIMIR PF;
```

FIN

8. Resolución de Algoritmos (Depurar)

Debemos de comprobar que el programa funciona correctamente y que obtiene el resultado esperado.

Sustituimos las variables y comparamos si el resultado de la fórmula del algoritmo coincide con el esperado:

$$PF = (R\text{^}3) + (2*Az) – (Am^2) =$$
$$= 3\text{^}3+(2*6)+4^2 = 27+12-16 = 23$$

El resultado obtenido es el mismo que se esperaba según nuestro diseño, es decir, una puntuación final de 23 puntos y se obtendrá por pantalla el siguiente mensaje:

La puntuación final de Andrés es 23

8. Resolución de Algoritmos (Ampliar)

- Acabamos de ver cómo calcular la puntuación final de un único jugador: Andrés
- Imagina que queremos obtener la puntuación de más jugadores: 5, 100 o 1500 jugadores. El **número de jugadores** pasa a ser un dato.
- Para calcular la puntuación final de cada jugador, es necesario indicar el **número de fichas** de cada color que ha conseguido cada uno
- Pero la solución no pasa por copiar el mismo programa N veces si no modificarlo para que se ejecute esas veces.

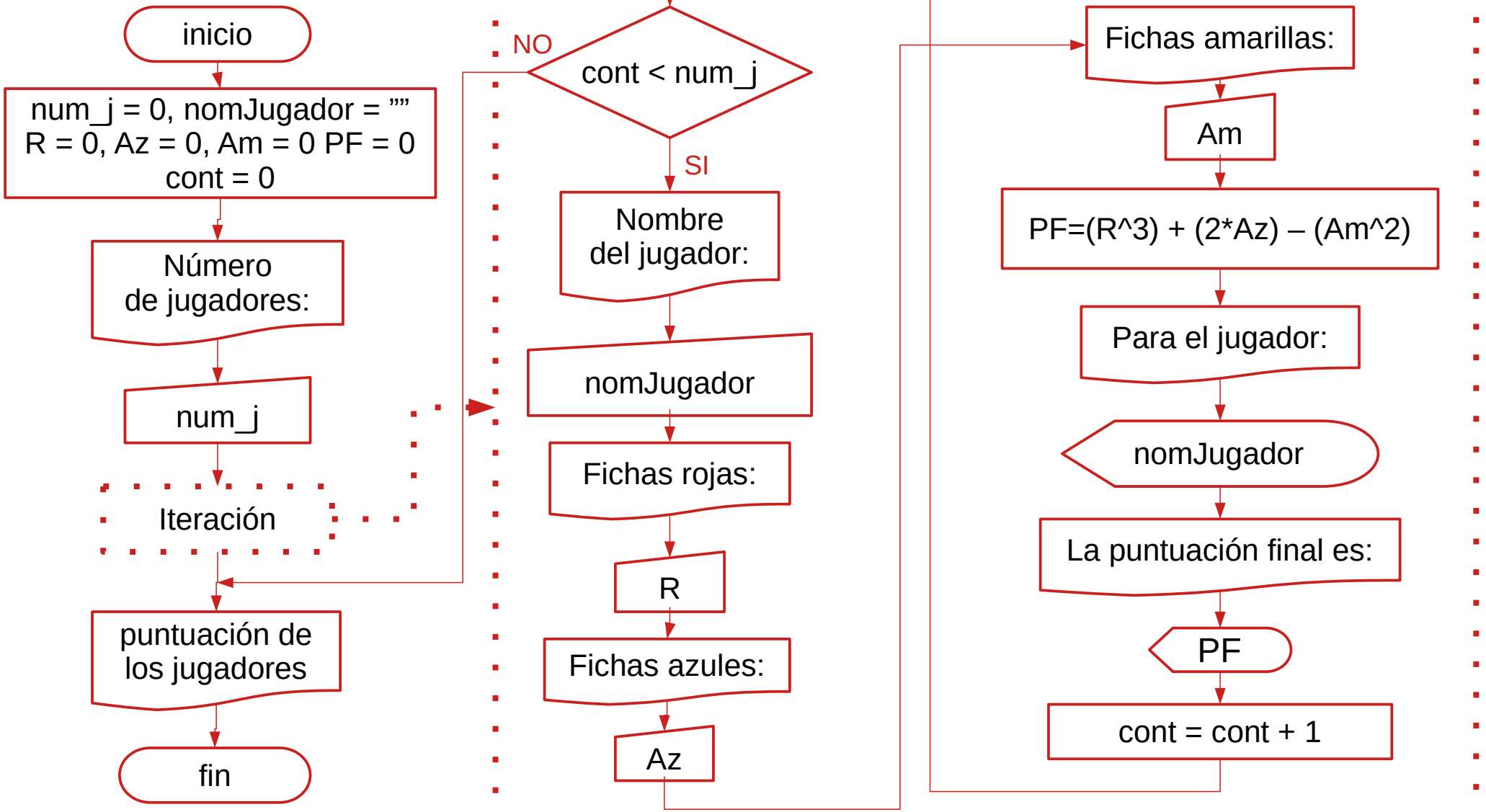
8. Resolución de Algoritmos (Ampliar)

Análisis del problema

- Hemos resuelto el problema con un jugador y unas fichas determinadas para ese jugador.

Varios jugadores (Hay que usar bucles)

- Pedir los datos de cada jugador:
  - Nombre
  - Fichas de colores
- Aplicar la fórmula a cada jugador.


inicio
num_j = 0, nomJugador = ””
R = 0, Az = 0, Am = 0 PF = 0
cont = 0
Número
de jugadores:
num_j
Iteración
puntuación de
los jugadores
fin
NO
cont < num_j
SI
Nombre
del jugador:
nomJugador
Fichas rojas:
R
Fichas azules:
Az
Fichas amarillas:
Am
PF=(R^3) + (2*Az) – (Am^2)
Para el jugador:
nomJugador
La puntuación final es:
PF
cont = cont + 1

Tarea resuelta - Traducir

```
INICIO
    TEXTO nomJugador="";
    ENTERO R=0,Az=0, Am=0, PF=0;
    ENTERO num_j=0, contador=0;
    IMPRIMIR "Número de jugadores: ";
    LEER numJ;
    MIENTRAS (contador < num_j)
        IMPRIMIR "Nombre del jugador";
        LEER nomJugador;
        IMPRIMIR "Fichas rojas: ";
        LEER R;
        IMPRIMIR "Fichas azules: ";
        LEER Az;
        IMPRIMIR "Fichas amarillas: ";
        LEER Am;
        PF=(R^3) + (2*Az) – (Am^2);
        IMPRIMIR "Para el jugador ";
        IMPRIMIR nomJugador;
        IMPRIMIR "La puntuación final es: ";
        IMPRIMIR PF;
        contador=contador +1;
    FMIENTRAS
    IMPRIMIR "se ha calculado la puntuación de los jugadores";
FIN
```

8. Resolución de Algoritmos (Ampliar)

Depurar el algoritmo: Pruebas

- Para poder comprobar que funciona correctamente deberemos realizar pruebas con unos datos de ejemplo.
- Los resultados obtenidos se muestran en una tabla simulando la pantalla.
- Por ejemplo, suponemos que se quiere calcular la puntuación de 4 jugadores.

Variables / n.º iteración	Valor inicial
num_j	4
contador	0
nomJugador	“”
R	0
Az	0
Am	0
PF	0

8. Resolución de Algoritmos (Traza)

Variables / n.º iteración	Valor inicial	Iteración1	Iteración 2	Iteración3	Iteración4	Iteración5
num_j	4	4	4	4	4	4
contador	0	0	1	2	3	4
nomJugador	“”	Pablo	Ana	Luis	Lola	-
R	0	10	8	7	6	-
Az	0	5	4	3	2	-
Am	0	0	3	5	7	-
PF	0	1010	511	324	171	-

Iteración 5

- contador < num_j con los valores 4<4, es **false**
- Ya no se ejecuta el código del mientras con lo que se imprime la instrucción posterior con el mensaje final y finaliza el programa

9. Ejercicios

1) Dado tres números enteros introducidos por teclado mostrar por pantalla los números ordenados de mayor a menor, en caso de ser alguno igual mostrará también un mensaje “Hay dos iguales” o “Hay tres iguales”.

2) Algoritmo que nos diga si una persona puede acceder a cursar un ciclo formativo de grado superior o no. (Averigua que se necesita para acceder a un GS)

3) Calcular la media de "n" números ingresados por teclado, mostrar por pantalla el resultado.

4) Acepta el reto nº247