Fundamentos de Programación

Estudio y confección de ordinogramas y pseudocódigo.

- Introducción. Diagramas de flujo.
- Diagramas de flujo del sistema: Organigramas.
- Diagramas de flujo del programa: Ordinogramas y pseudocódigo.
- Representación de estructuras básicas.
- Ejemplos prácticos.
- Comentarios.
- Elementos auxiliares de un programa.
- Ejemplos prácticos.
- Ejercicios.

Instrucciones repetitivas

Instrucciones repetitivas: son aquellas que permiten variar o alterar la secuencia normal de ejecución de un programa haciendo posible que un grupo de operaciones (acciones) se repite un número determinado o indeterminado de veces, dependiendo en este caso del cumplimiento de una condición.
Veremos tres tipos, aunque una es una variante de otra:

- Bucle Mientras.
- Bucle Hasta.
- Bucle Para (es una restricción del bucle mientras).

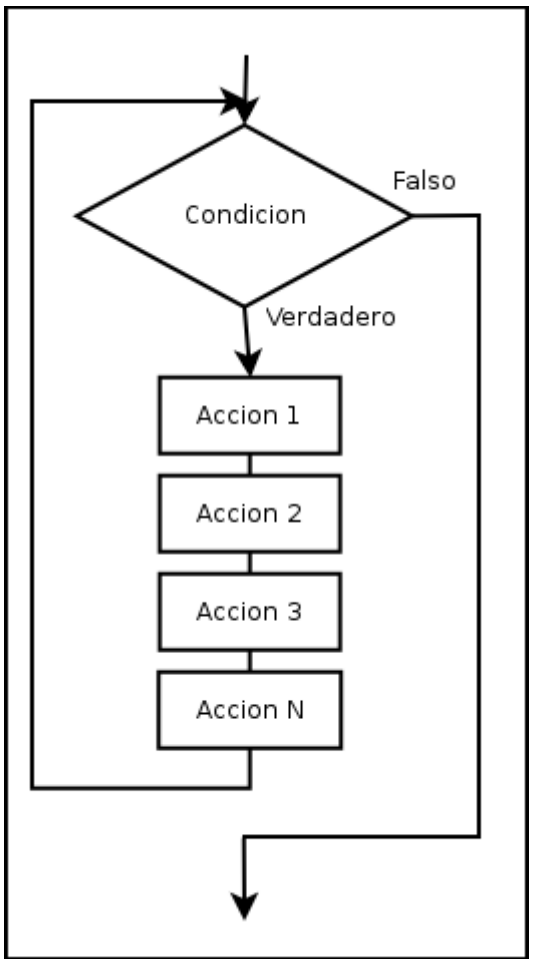
Estructura Mientras o WHILE.

el bloque de acciones se repite **mientras que** la condición sea cierta, evaluándose siempre la condición antes de entrar en el bucle, por ello es posible que las acciones no se ejecuten nunca.

Pseudocódigo

```
Mientras Condición hacer
    Instrucción 1
    Instrucción 2
    ...
    Instrucción N
FinMientras
```

Ordinograma



Ejemplo de funcionamiento de un buble Mientras.

Estructura Hasta o Until.

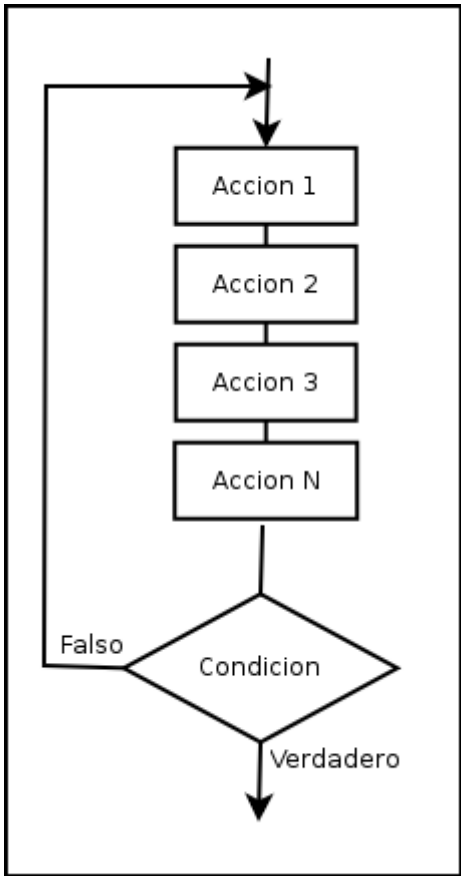
El bloque de instrucciones se repite **hasta** que la condición sea cierta, y la condición se evalúa al final del bloque por lo que **siempre** se ejecutarán al menos una vez el bloque de instrucciones.

Pseudocódigo **Ordinograma**

Repetir

Instrucción 1
Instrucción 2
...
Instrucción N

Hasta Condición.



Estructura Para o for: se conoce de antemano el número de veces que se ejecutará el bloque de instrucciones.
El bloque de acciones se repite **mientras que** la condición sea cierta, evaluándose siempre la condición antes de entrar en el bucle, por ello es posible que las acciones no se ejecuten nunca. Esta explicación es identica a un bucle mientras, pero en un bucle para debe tener las siguientes características:

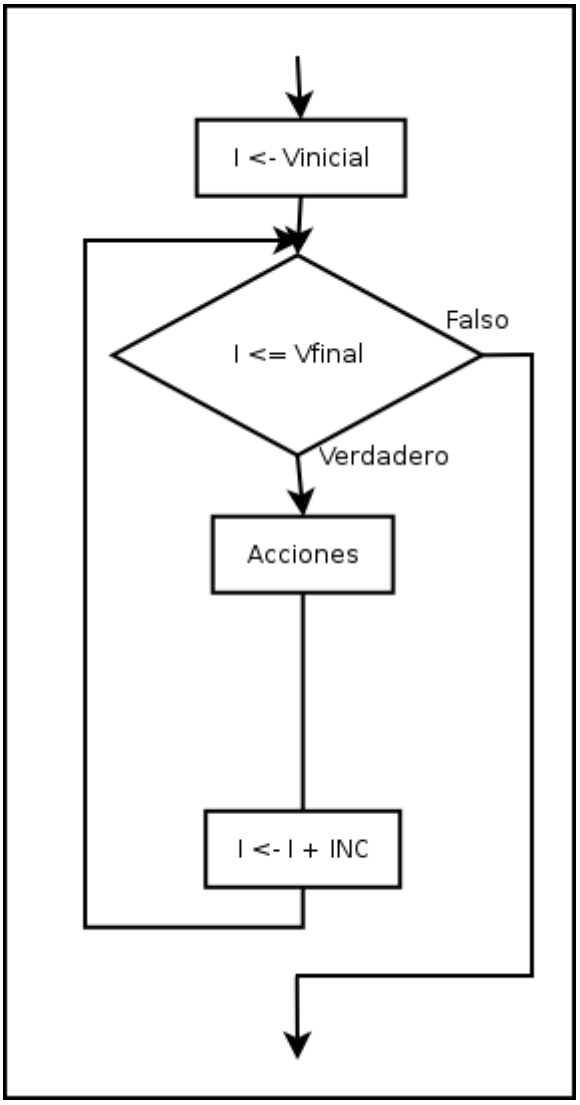
1. Una variable contador se inicializa con valor inicial.
2. La condición siempre debe ser: variable_contador <= valor_final
3. En cada interacción la variable contador se incrementa en un determinado valor.

Pseudocódigo **Ordinograma**

Para Var_Cont de ValorInicial a ValorFinal con Incremento = n

    Instrucción 1
    Instrucción 2
    ...
    Instrucción N

FinPara



Ejemplo de funcionamiento de un **bucle Para,** que se ejecuta 3 veces, desde 1 hatsa 3

Formas de terminar un bucle

Objetivos

Uno de los peligros que se corren cuando se escribe un bucle es que éste no acabe nunca, lo que se denomina **bucle infinito**, para no cometer este error grave debemos recordar que las condiciones de los bucles deben poder cambiar dentro del bucle, es decir que si por ejemplo utilizamos una variable comparada con una contante dicha variable debe poder cambiar de valor dentro del bucle.

Las estructuras repetitivas deben incluir un mecanismo para que éstas se acaben. Algunos de los métodos más usados para esa labor son los siguientes:

1. Cuando sabemos el número de veces que se va a repetir la estructura, mediante un contador.

Por ejemplo, en un enunciado del tipo: “imprimir la tabla del 7”, sabemos que el proceso va desde 1 a 10, por lo tanto, usaremos un contador.

```
. . .
cont = 1
Mientas cont <=10
      . . .

      cont = cont + 1
FinMientras
. . .
```

2. Preguntando si queremos seguir en el bucle.

Por ejemplo, en un ejercicio del tipo “introducir N alumnos y hallar su media”, debemos preguntar si queremos introducir más alumnos:

```
. . .
Mientras ((seguir=”s”) o (seguir=”S”))
      . . .

         escribir “Introducir más alumnos?”
         leer seguir

FinMientras
. . .
```

3. Usando un valor centinela.

Así, en el caso del siguiente problema: “Introducir N notas hasta introducir un 10”:

```
. . .

Leer nota
Mientras (nota <> 10)

         . . .

         Leer nota

   FinMientras
. . .
```

4. Usando un interruptor que tomará el valor lógico true o false.

En un ejemplo como “Repetir ciertas instrucciones mientras la condición sea cierta”:

```
. . .
Mientras (SW = Verdadero)

        . . .

FinMientras
. . .
```

Elementos auxiliares

Objetivos

Son variables que realizan funciones especificas dentro de un programa, y por su gran utilidad, frecuencia de uso y peculiaridades, conviene hacer un estudio separado de las mismas.

CONTADORES.

Si vamos a repetir una acción un número determinado de veces y esa variable se va a incrementar siempre en una cantidad constante, se denomina **contador**. Deberíamos llamarla algo así como CONT, CONTA, CONTADOR. O CONT1, CONT2, ... CONTN si tuviéramos varios contadores dentro de un programa.

Se utilizan en los siguientes casos:

1. para contabilizar el número de veces que es necesario repetir una acción (variable de control de un bucle).
2. para contar un suceso particular solicitado por el enunciado del problema.

Un contador debe inicializarse a un valor inicial (normalmente a cero) e incrementarse cada vez que ocurra un suceso.

ACUMULADORES.

Si por el contrario, dicho objeto se va **incrementando de forma variable** se denomina **acumulador**. Deberemos llamarla ACU, ACUM, ACUMULA, ACUMULADOR, SUMA, ... u otra palabra significativa.

Se utiliaza en aquellos casos en que se desea obtener el total acumulado de un conjunto de cantidades, siendo inicializado con un valor cero.

También en ocasiones hay que obtener el total acumulado como producto de distintas cantidades, en este caso se inicializará a uno.

Por ejemplo: imprimir la suma de N edades.

INTERRUPTORES.

Por último, tenemos ciertas variables que **pueden tomar dos valores: cierto o falso**. Se les denomina **interruptores o switches** y su función es que ciertas instrucciones se ejecuten mientras tenga un valor determinado. A las variables de este tipo las denominaremos SW.

Se utiliza para:

- recordar que un determinado suceso a ocurrido o no en un punto determinado del programa, y poder así realizar las decisiones oportunas.
- para hacer que dos acciones diferentes se ejecuten alternativamente dentro de un bucle.

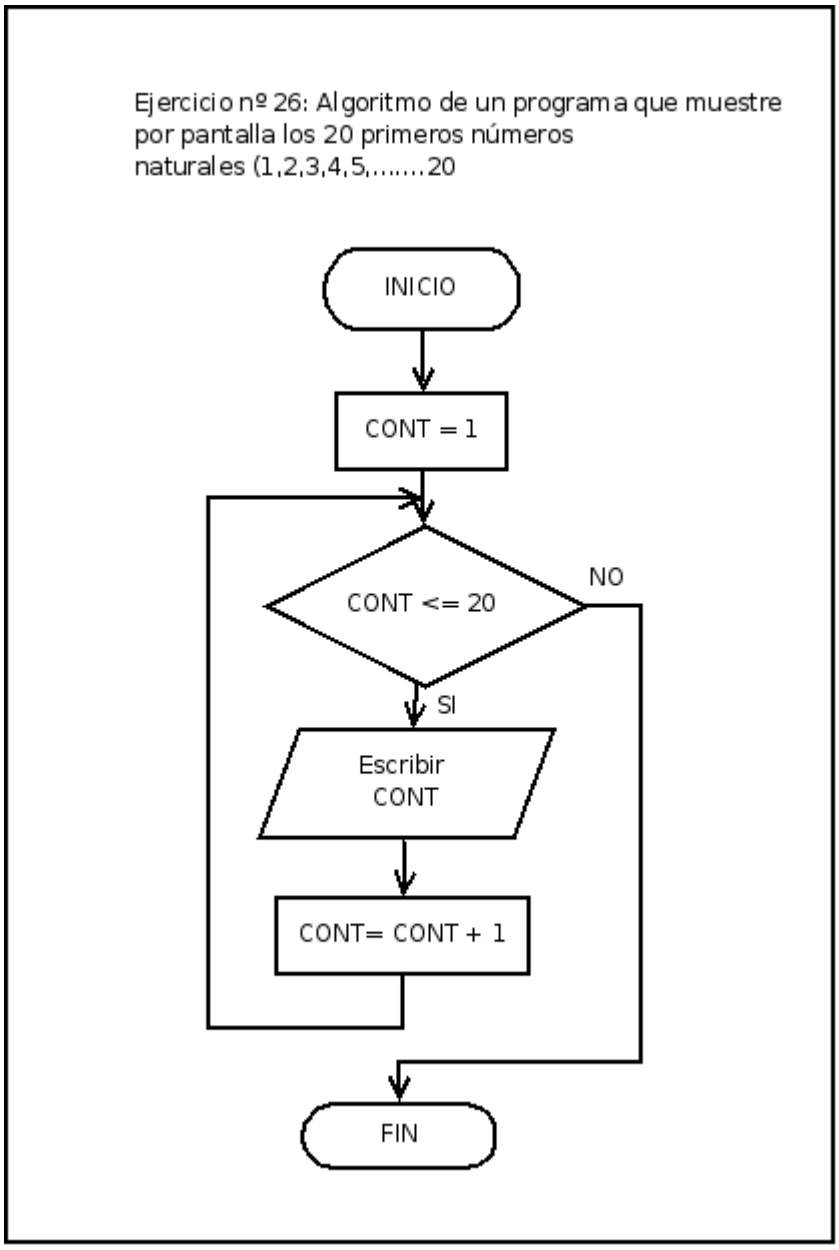
Por ejemplo: introducir N edades y acabar al introducir un 99.

Ejercicios repetición

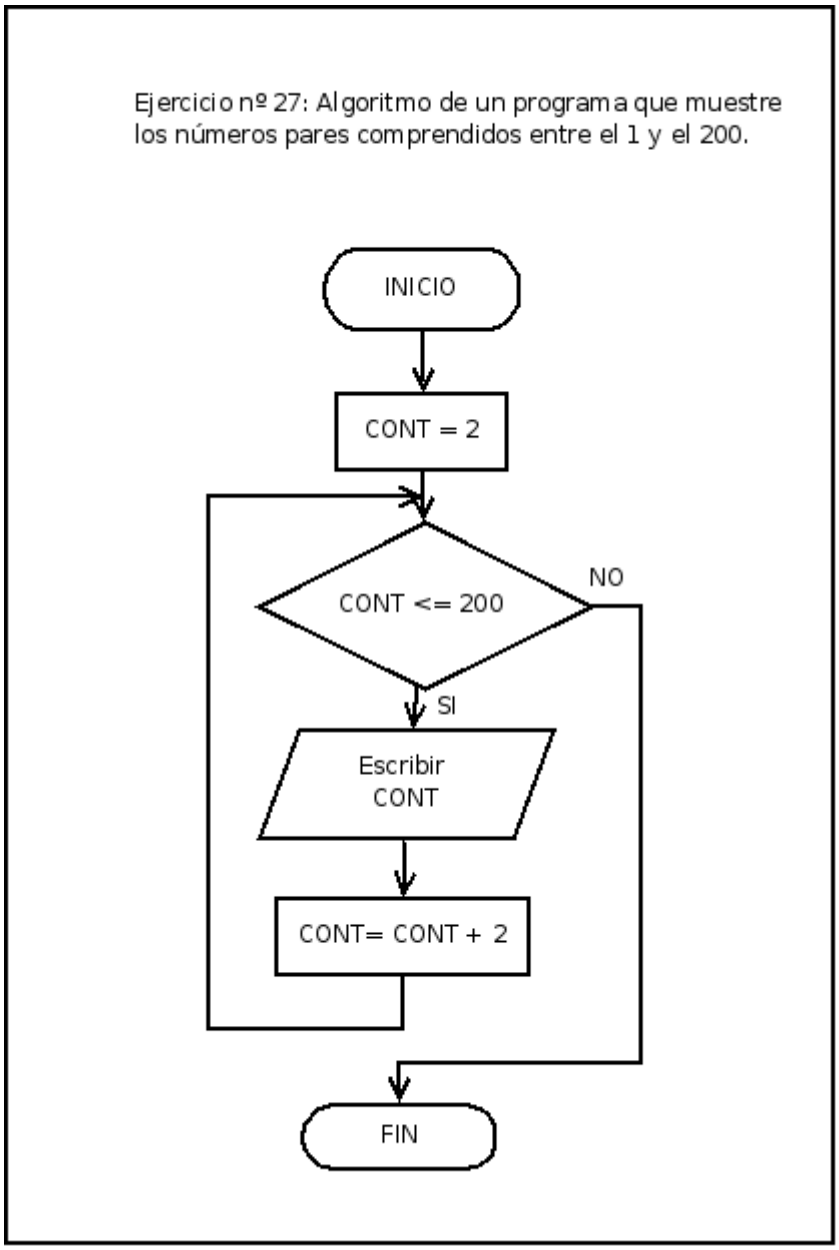
Dibuja los siguientes ordinogramas

Ejercicios con Mientras (WHILE) y Para (FOR).

Ejercicio nº 26: Algoritmo de un programa que muestre por pantalla los 20 primeros números naturales (1,2,3,4,5,…….20).

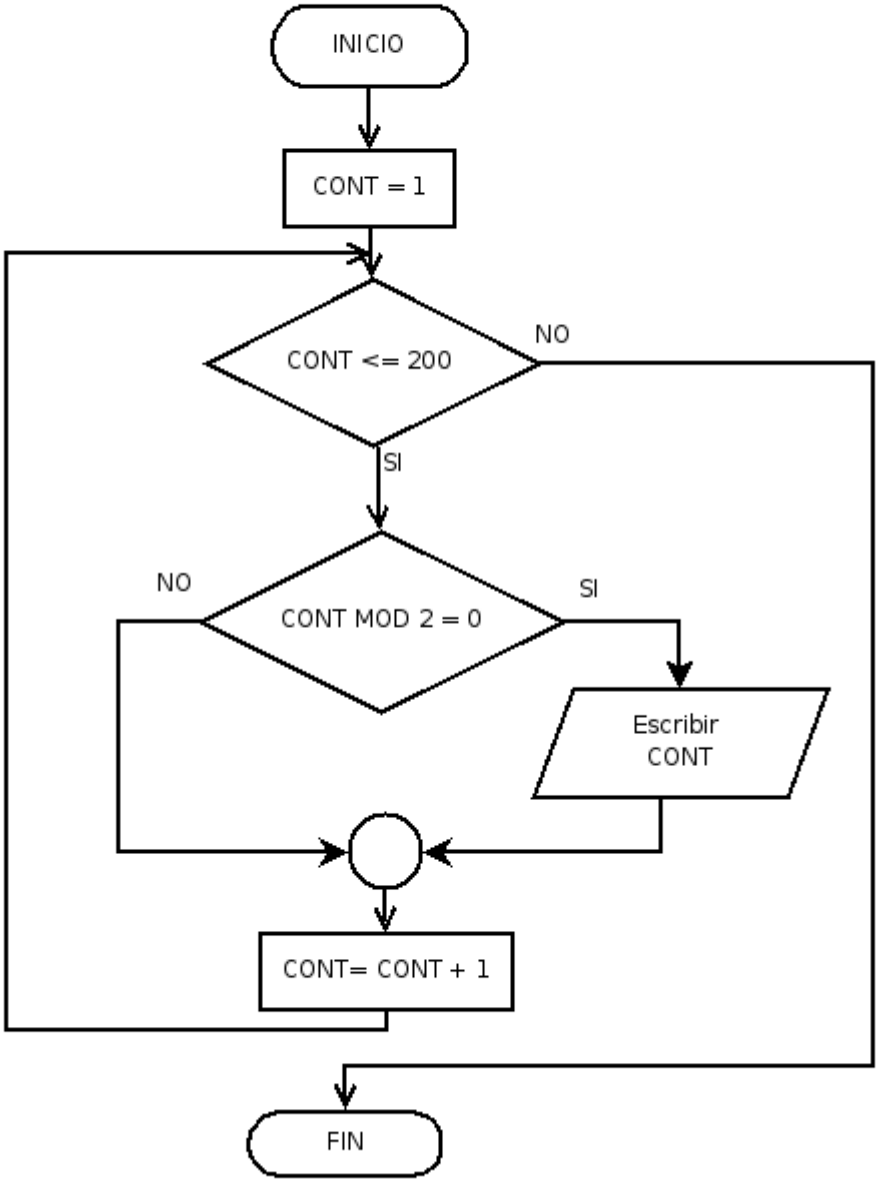


Ejercicio nº 27: Algoritmo de un programa que muestre los números pares comprendidos entre el 1 y el 200.



Ejercicio nº 28: Igual pero sumando de 1 en 1.

Ejercicio nº 28: Algoritmo de un programa que muestre los números pares comprendidos entre el 1 y el 200 pero sumando de 1 en 1.



Ejercicio Común. Ejercicio nº 29: Algoritmo que muestra los números desde el 1 hasta un número N, que se introducirá por teclado.

Ejercicio Común. Ejercicio nº 30: Algoritmo que lee un número entero positivo N y calcula y visualiza su factorial N!

0! = 1

1! = 1

2! = 2 * 1

3! = 3 * 2* 1

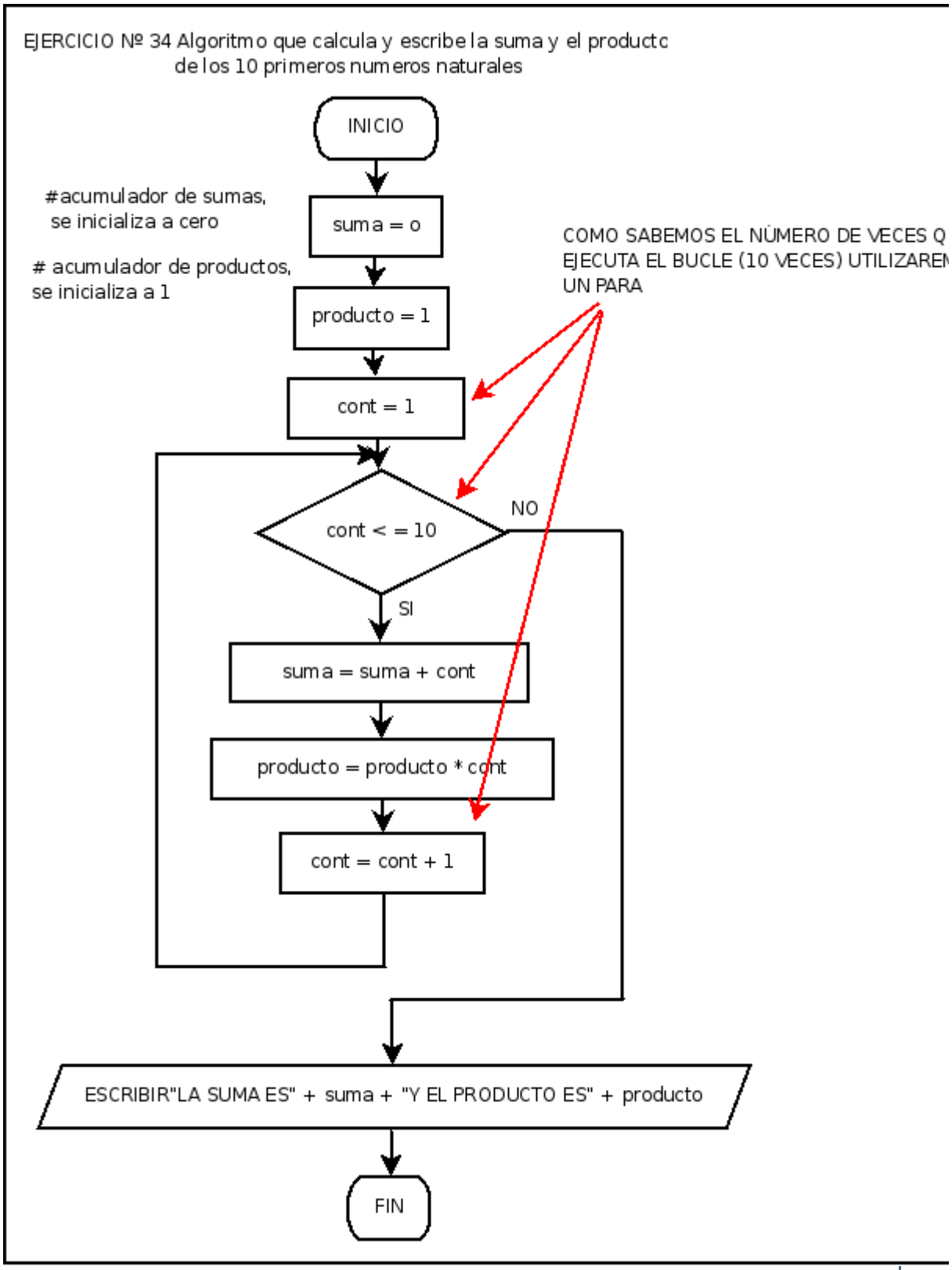
N! = N * (N-1) * (N-2)……..* 3*2*1

Ejercicio Común. Ejercicio nº 31: Algoritmo que lee 100 números no nulos y visualiza un mensaje de si ha leído algún número negativo o no.

Ejercicio Común. Ejercicio nº 32: Algoritmo que lee 100 números no nulos, y obtiene y visualiza un mensaje indicando cuantos son positivos y cuantos negativos.

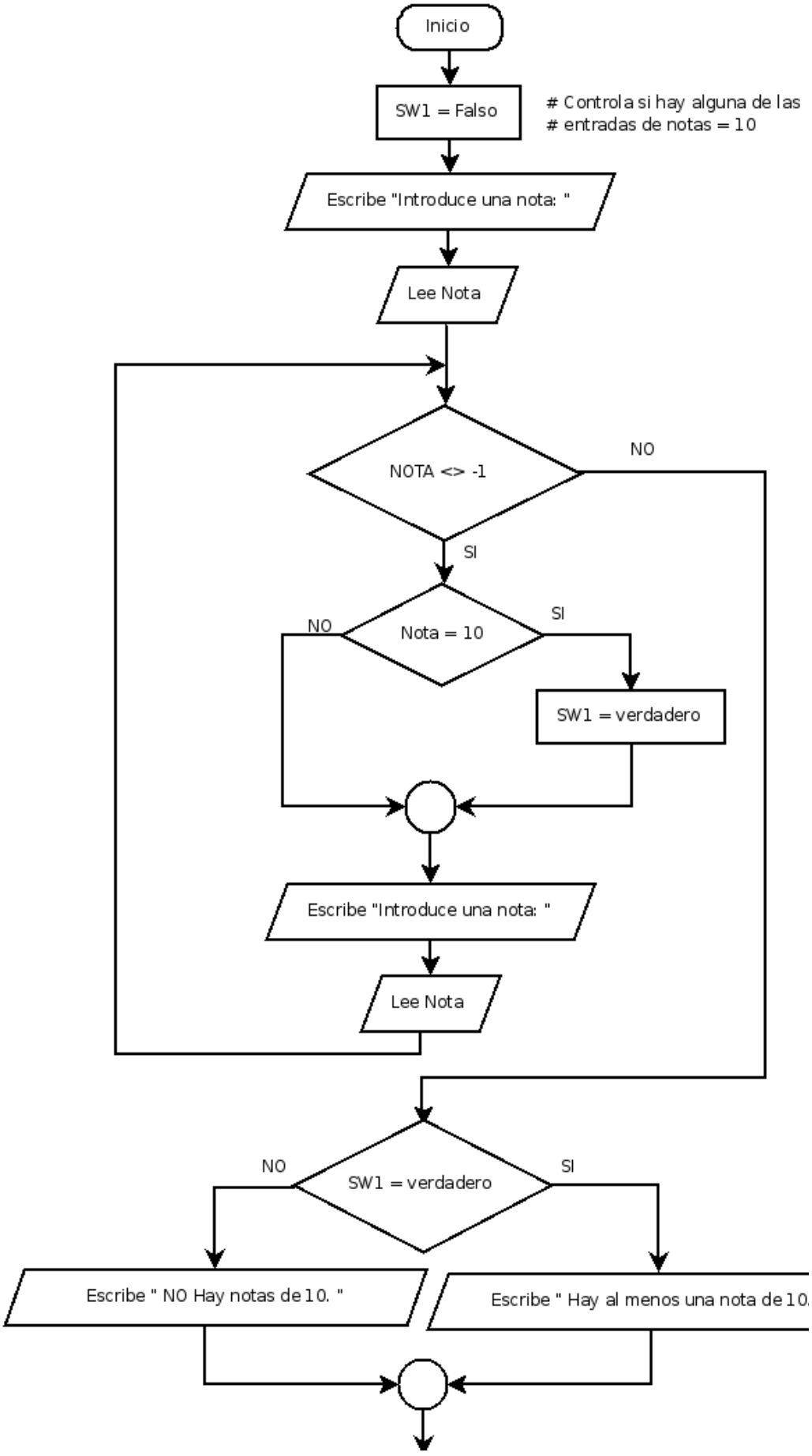
Ejercicio Común. Ejercicio nº 33: Algoritmo que lee una secuencia de números no nulos, terminada con la introducción de un 0, y obtiene e imprime si ha leído algún número negativo, cuantos positivos y cuantos negativos.

Ejercicio nº 34: Algoritmo que calcula y escribe la suma y el producto de los 10 primeros números naturales.



Ejercicio nº 35: Algoritmo que lee una secuencia de notas (con valores que van de 0 a 10) que termina con el valor -1 y nos dice si hubo o no alguna nota con valor 10.

Ejercicio 35: Algoritmo que lee una secuencia de notas (con valores que van desde 0 a
que termina con el valor -1 y nos dice si hubo o no alguna nota con valor 10



Ejercicio Común. Ejercicio nº 36: Algoritmo que suma independientemente los pares y los impares de los números comprendidos entre 100 y 200.

Ejercicio Común. Ejercicio nº 37: Algoritmo que calcule el valor de elevar un número real, A, a un número exponente entero, B.

Ejercicio Común. Ejercicio nº 38: Dado un número y guardado en una variable, el usuario debe adivinar el número almacenado, el ordenador nos irá diciendo si es mayor, menor o igual.

Ejercicio Común. Ejercicio nº 39: El usuario "piensa" un número del 1 al 100 y el ordenador lo adivina. (el usuario debe indicarle al ordenador si es mayor, menor o igual):

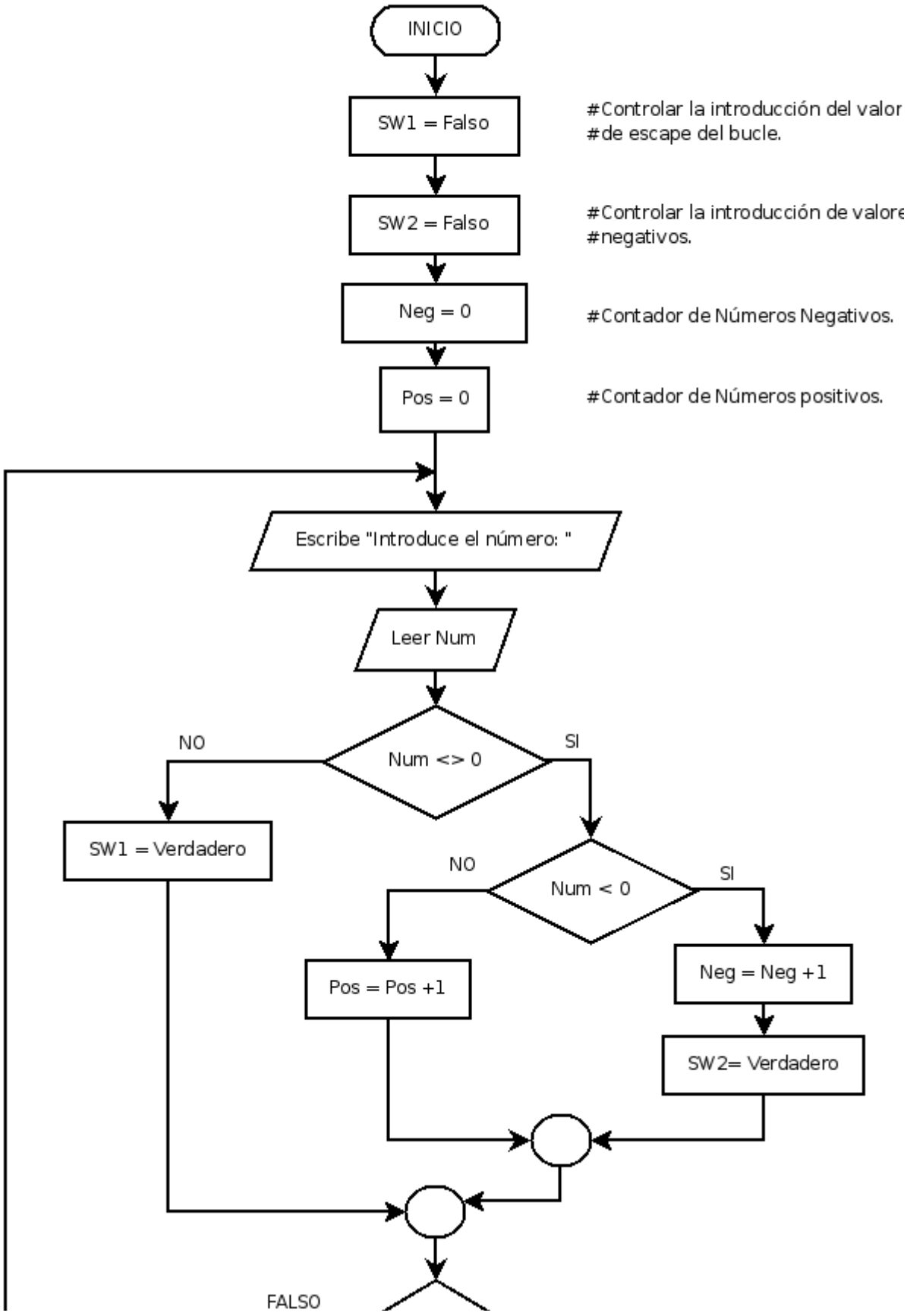
Ejercicio Común. Ejercicio nº 40: Dada una cantidad de euros (múltiplo de 5) realizar el desglose de billetes (500,200,100,50,20,10,5), siempre intentando dar el mínimo de billetes posible.

Dibuja los siguientes ordinogramas

Ejercicios con HASTA (UNTIL).

Ejercicio nº 41: Algoritmo que lee una secuencia de números no nulos, terminada con la introducción de un 0, y obtiene e imprime si ha leído algún número negativo, cuantos positivos y cuantos negativos.

Ejercicio 41 : Algoritmo que lee una secuencia de números no nulos, terminada con la introducción de un 0, y obtiene e imprime si ha leido algún número negativo, cuantos son positivos y cuantos negativos.



Ejercicio Común. Ejercicio nº 42: Algoritmo que lee una secuencia de notas (con valores que van de 0 a 10) que termina con el valor -1 y nos dice si hubo o no alguna nota con valor 10.

Ejercicio Común. Ejercicio nº 43: Algoritmo de un programa que muestra un menú con las siguientes opciones:

1. Suma.
2. Resta.
3. Multiplicación.
4. División.
5. Salir.

Y permite hacer las operaciones con dos numeros hasta que pulsemos la opción de salir.