Elementos básicos del lenguaje

VISIÓN GENERAL Y ELEMENTOS BÁSICOS DEL LENGUAJE.

ÍNDICE.

Estructuras de repetición

Sentencias break y continue

Estas estructuras de salto **no deben ser utilizadas en programación estructurada**.

La sentencia break solo será utilizada en la estructura de alternativa multiple switch.

Estructuras de control.

Las estructuras de control son construcciones hechas a partir de palabras reservadas del lenguaje que permiten modificar el flujo de ejecución de un programa. De este modo, pueden crearse construcciones alternativas y bucles de repetición de bloques de instrucciones. Hay que señalar que un bloque de instrucciones se encontrará encerrado mediante llaves {.....} si existe más de una instrucción.

Los bucles son estructuras de repetición, bloques de instrucciones que se repiten un número de veces, mientras se cumpla una condición o hasta que se cumpla una condición.
Existen tres construcciones para estas estructuras de repetición:

- Bucle for.
- Bucle while.
- Bucle do-while.

Como regla general puede decirse que se utilizará:
El bucle **for**:

- **cuando se conozca de antemano el número exacto de veces que ha de repetirse** un determinado bloque de instrucciones.

El bucle **while-do**:

- **cuando NO se conoce exactamente el número de veces** que se ejecutará el bucle, pero cuando es posible que no deba ejecutarse **ninguna vez**.

El bucle **do-while**:

- cuando NO se conoce exactamente el número de veces que se ejecutará el bucle pero se sabe que por lo menos se ha de ejecutar una vez.

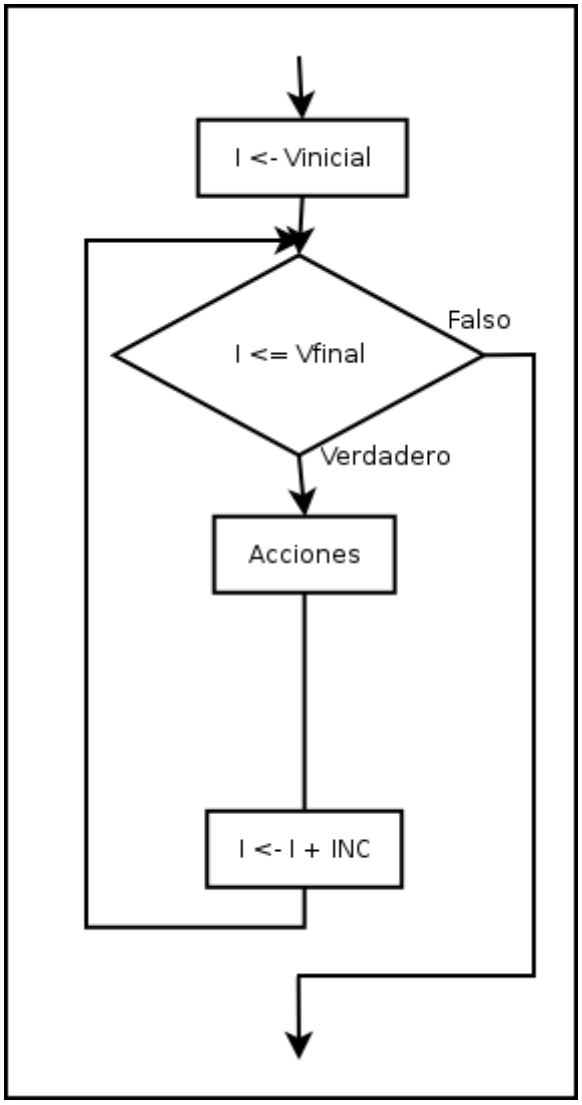
Estas reglas son generales y algunos programadores se sienten más cómodos utilizando principalmente una de ellas.

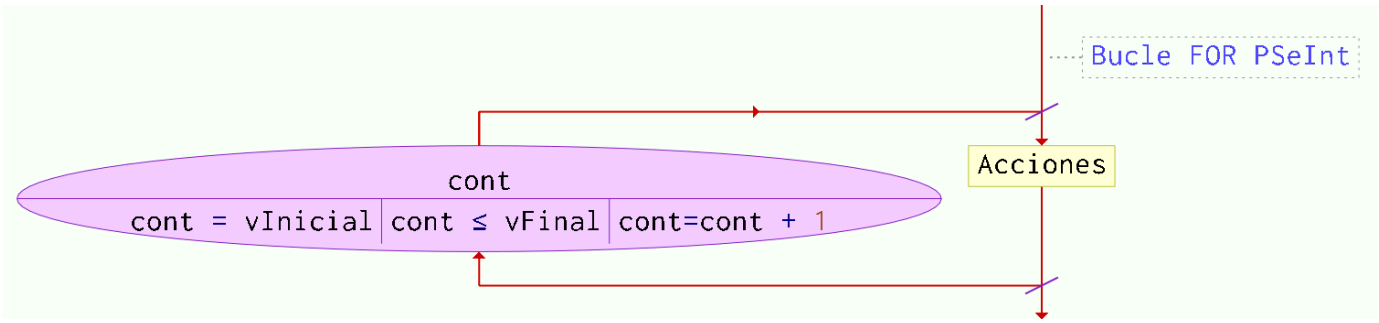
Bucle PARA (for).

Sintaxis en JAVA

```
for (inicialización ; condición ; incremento){
        bloque instrucciones;
        }
```

- La cláusula **inicialización** es una instrucción que **se ejecuta una sola vez al inicio del bucle**, normalmente para inicializar un contador.
- La cláusula **condición** es una expresión lógica, que **se evalúa al inicio de cada nueva iteración del bucle**. En el momento en que dicha expresión se evalúe a false, se dejará de ejecutar el bucle y el control del programa pasará a la siguiente instrucción (a continuación del bucle for).
- La cláusula **incremento** es una instrucción que **se ejecuta en cada iteración del bucle como si fuera la última instrucción dentro del bloque de instrucciones**. Generalmente se trata de una instrucción de incremento o decremento de alguna variable.





AUNQUE NO ES CONVENIENTE, por claridad del código, cualquiera de estas tres cláusulas puede estar vacía, aunque SIEMPRE hay que poner los puntos y coma (;).

Ejemplo: Programa que muestra los números naturales (1,2,3,4,5,6,...) hasta un número introducido por teclado.

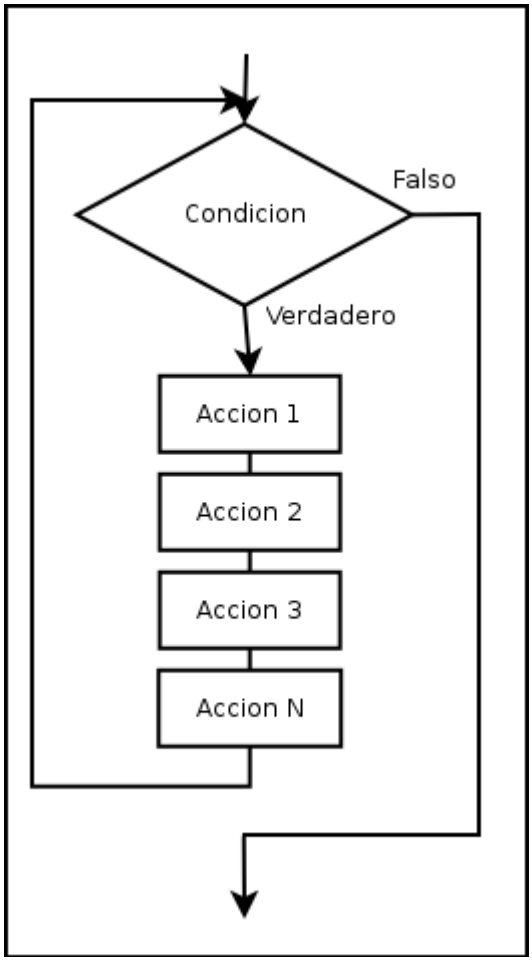
```
class Numeros {
  public static void main(String arg[ ]) {

    BufferedReader stdin;
```

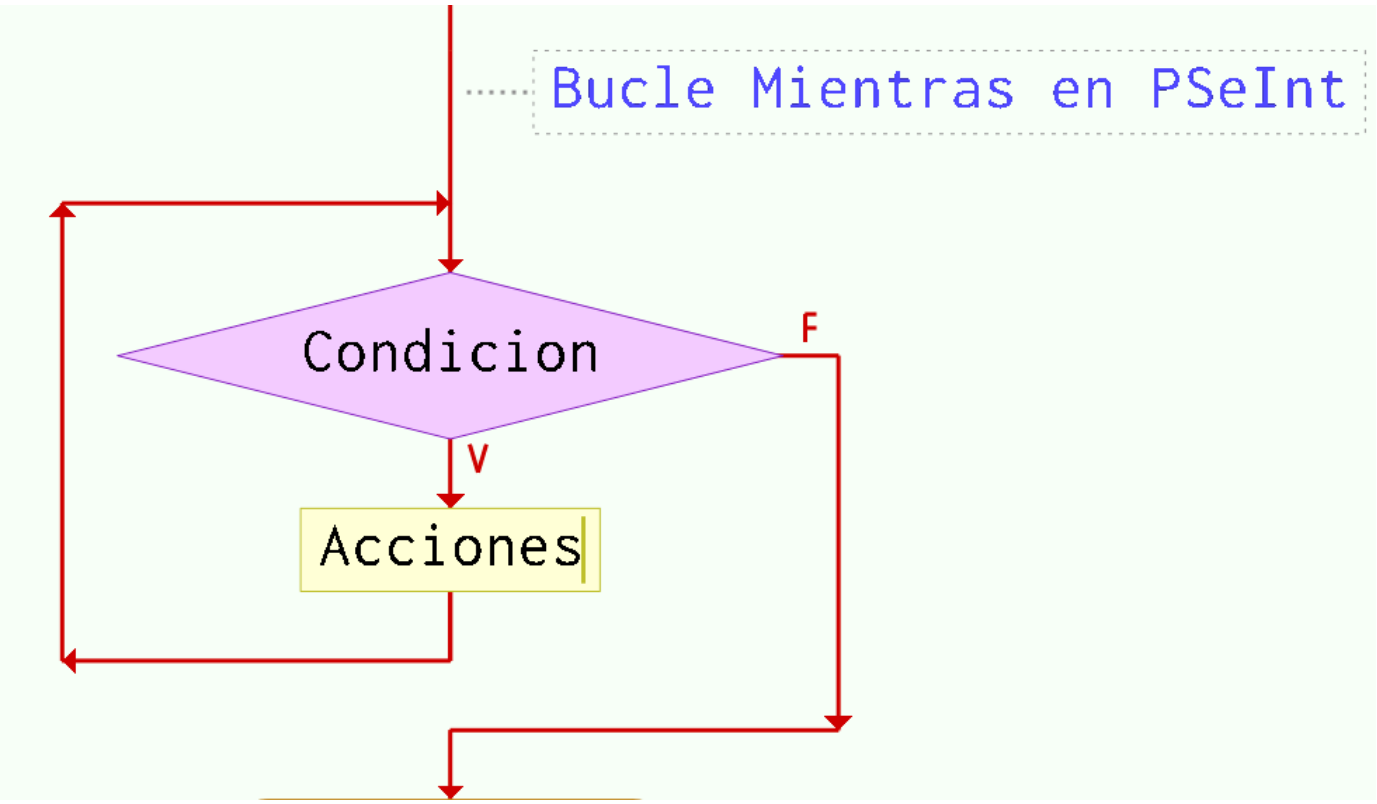
```
stdin=new BufferedReader(new InputStreamReader(System.in));
int num, cont;
System.out.println("Dame un número tope :");
num =Integer.parseInt(stdin.readLine());
for (cont=1; cont<=num; cont++){

        System.out.println("Número :" + cont);

}

}

}
```

Bucle MIENTRAS (While do).

Sintaxis en JAVA

```
while (Expresión) {
                    bloque instrucciones;
                }
```





El bloque de instrucciones **se ejecuta mientras se cumple una condición** (mientras Condición se evalúe a true), la condición se comprueba ANTES de empezar a ejecutar por primera vez el bucle, por lo que si la Condición se evalúa a false en la primera vez, el bloque de instrucciones no se ejecutará ninguna vez.

Ejemplo: Programa que muesta los números naturales (1,2,3,4,5,6,...) hasta un número introducido por teclado.

```
class Numeros {
public static void main(String arg[ ]) {
BufferedReader stdin;
stdin=new BufferedReader(new InputStreamReader(System.in));
int num, cont;
System.out.println("Dame un número tope :");
num =Integer.parseInt(stdin.readLine());
cont = 1;
while (cont<=num)
        {

       System.out.println("Número :" + cont);
       cont = cont +1;

}
```

```
}
}
```

Bucle HASTA (do While).

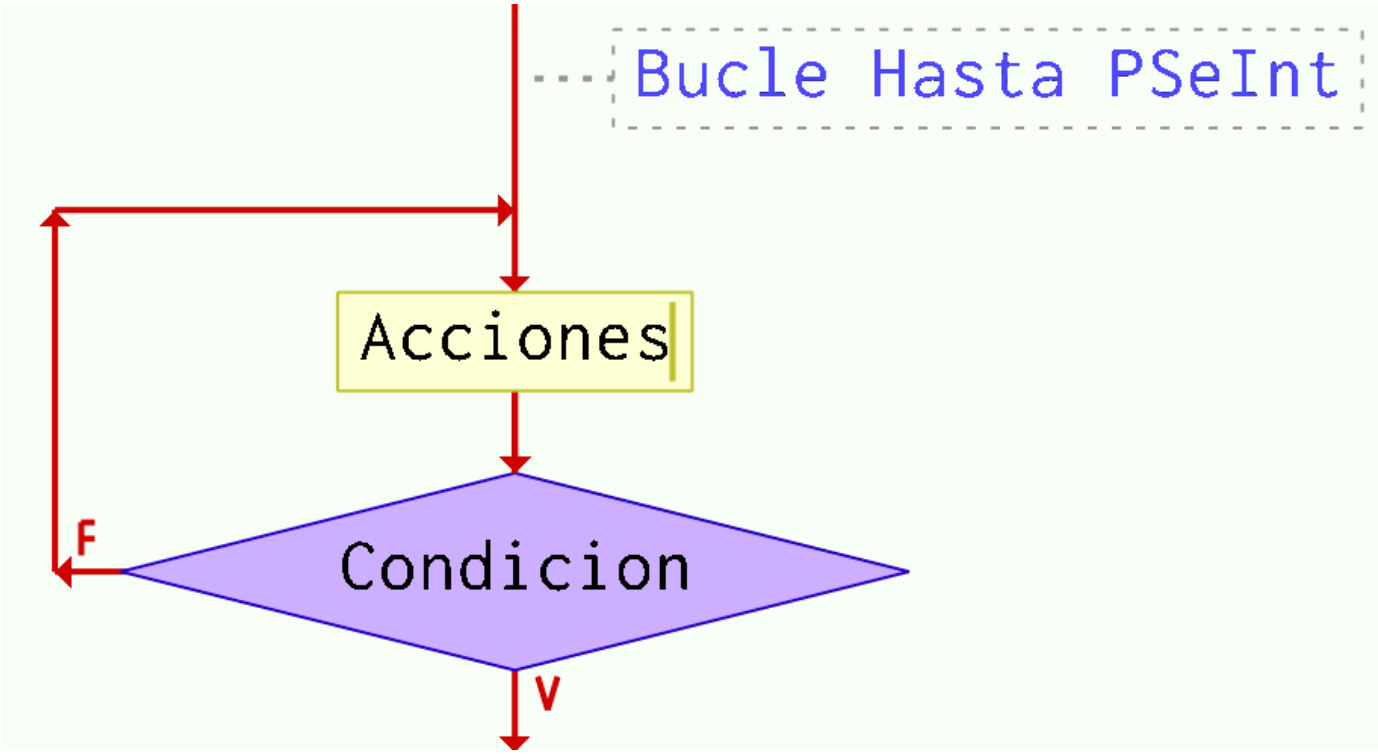
Sintaxis en JAVA:

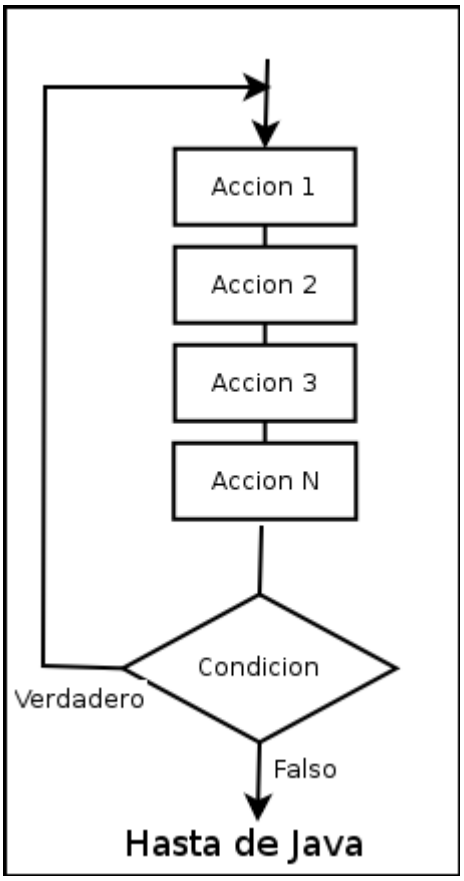
```
do {
        bloque instrucciones;
        }
while (Expresión);
```

IMPORTANTE: Java no ha cumplido con las especificaciones básicas de construcción de ordinogramas, y a convertido la estructura HASTA en un MIENTRAS que se evalua al final.

Podríais pensar que todos los ejercicios que hemos hecho son incorrectos, pero no es cierto, en realidad solo tenemos que cambiar la condición del hasta para adecuarnos a JAVA.

En este tipo de bucle, bloque instrucciones se ejecuta siempre al menos una vez, y ese bloque de instrucciones se ejecutará mientras condición se evalúe a true. Por lo tanto, entre las instrucciones que se repiten deberá existir alguna que, en algún momento, haga que la condición se evalúe a false, de lo contrario el bucle sería infinito.





Ejemplo: Programa que muesta los números naturales (1,2,3,4,5,6,...) hasta un número introducido por teclado.

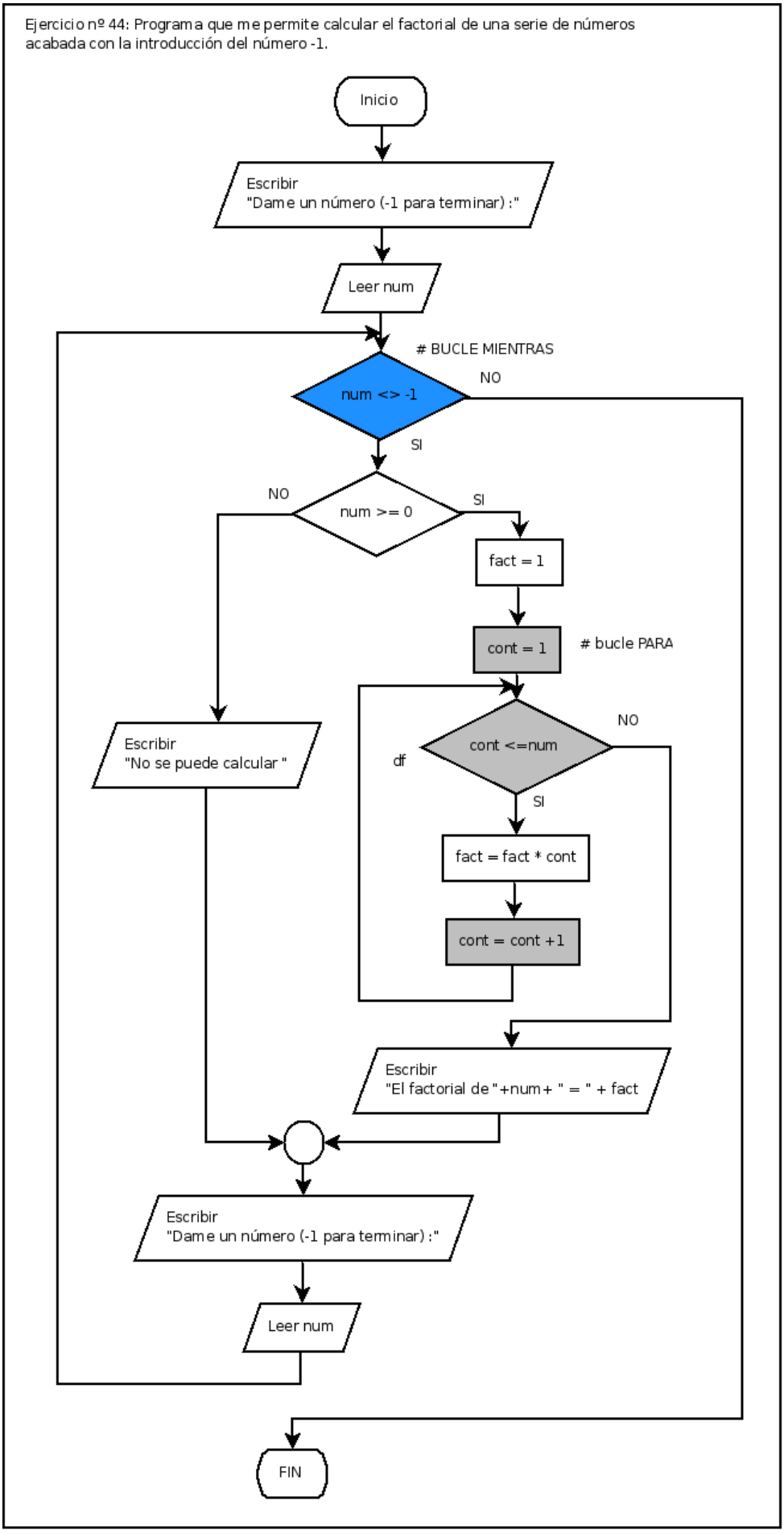
```
class Numeros {
public static void main(String arg[ ]) {
BufferedReader stdin;
stdin=new BufferedReader(new InputStreamReader(System.in));
int num, cont;
System.out.println("Dame un número tope :");
num =Integer.parseInt(stdin.readLine());
cont = 1;
do {

        System.out.println("Número :" + cont);
        cont = cont +1;

}while (cont<=num);

}
}
```

Bucles anidados

Dibuja los siguientes ordinogramas

Ejercicios repeticiones anidadas.

Ejercicio nº 44: Programa que me permite calcular el factorial de una serie de números acabada con la introducción del número -1.



Ejercicio común. Ejercicio nº 45: Crear un programa que pida un número entero y muestre la tabla de multiplicar correspondiente a dicho número. El programa terminará cuando pulsemos el

valor -1.

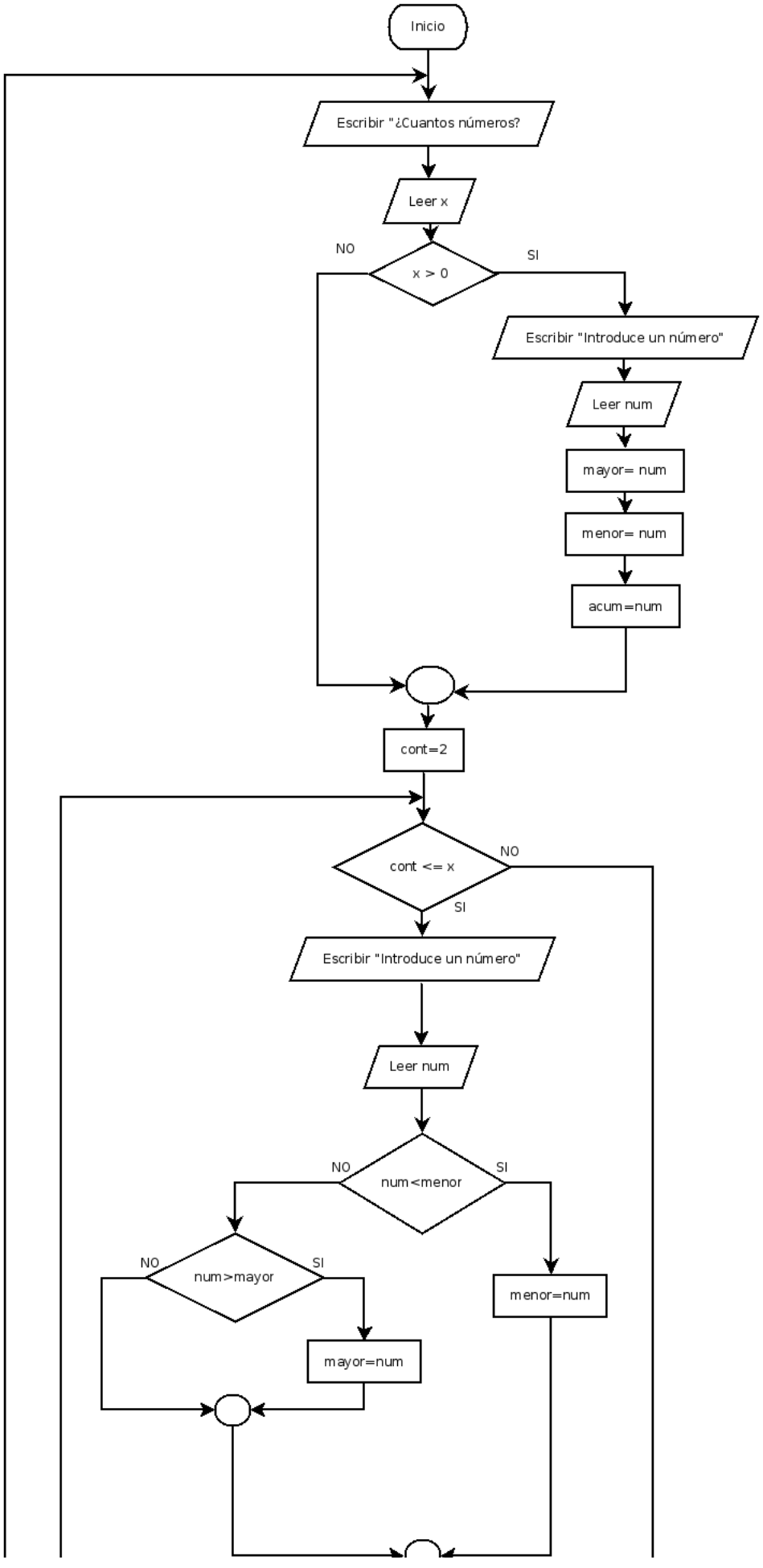
Ejercicio común. Ejercicio nº 46: Escribir un programa que me ayude a aprender las tablas de multiplicar.

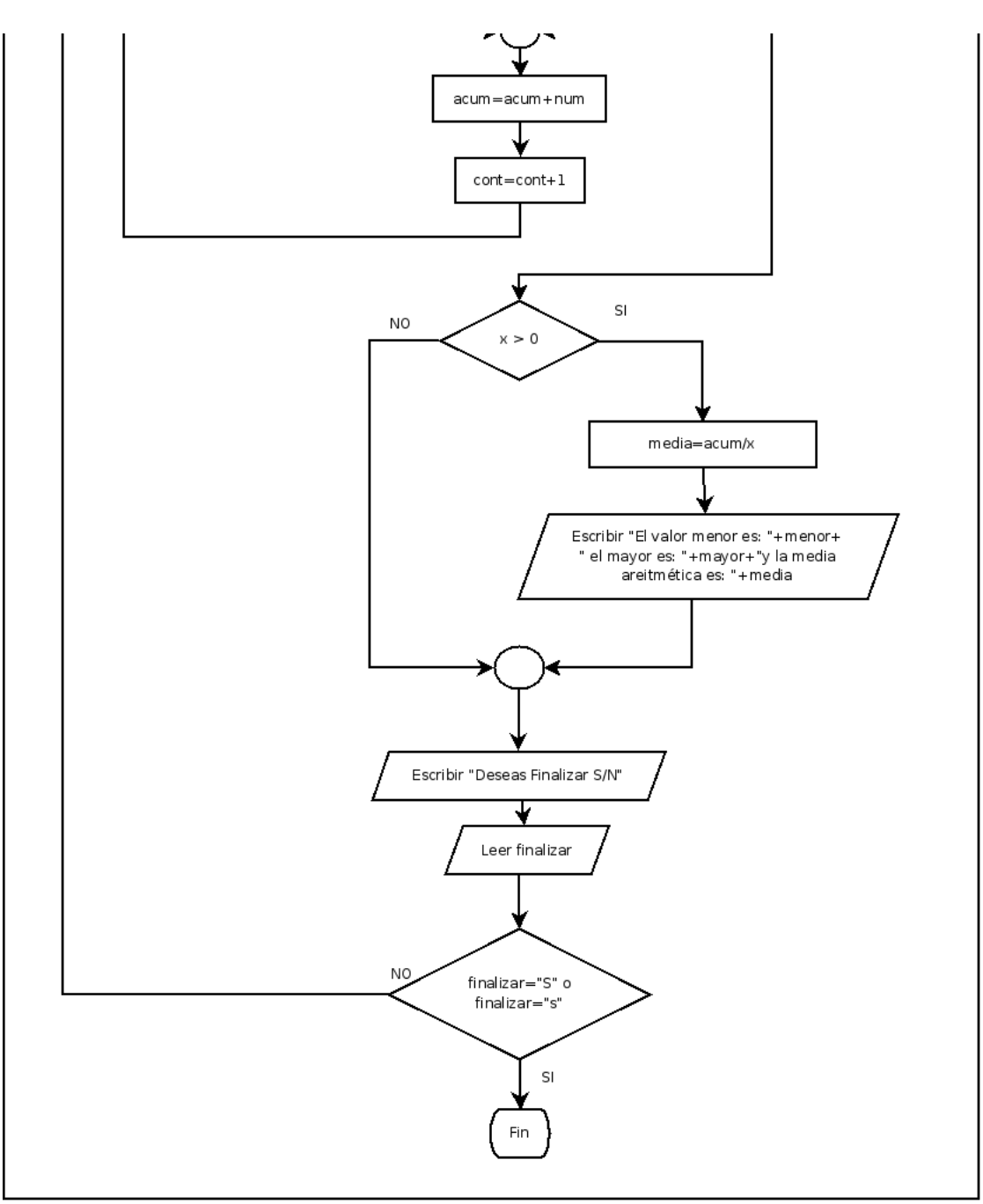
Para ello se irá pidiendo la tabla de multiplicar de un número (pedido por teclado con anterioridad) y comprobando que los valores introducidos son correctos. Si es así el programa escribirá "CORRECTO" y en caso contrario deberá escribir "LO SIENTO, SE HA EQUIVOCADO. LA RESPUESTA CORRECTA ES número".

La última línea mostrará el número de aciertos.

Ejercicio nº 47: Escribir un programa que primero pida por pantalla con cuántos números se va a trabajar (por ejemplo x) y luego pida x números por pantalla. Después de introducir los x números se mostrará por pantalla , el mayor, el menor y la media aritmética.

Ejercicio nº 47: Escribir un programa que primero pida por pantalla con cuántos números se va a trabajar (por ejemplo x) y luego pida x números por pantalla. Después de introducir los x números se mostrará por pantalla , el mayor, el menor y la media aritmética.





Ejercicio común. Ejercicio nº 48: Programa que escriba en pantalla una tabla con cuadrados y cubos a partir del 1 hasta otro número tope (este último introducido por teclado).

Ejercicio común. Ejercicio nº 49: Igual que el anterior pero introduciendo por teclado el tope de la potencia.

Ejercicio común. Ejercicio nº 50: Programa que lee un numero entero y positivo y nos indica si es primo o no.

Ejercicios con bucles

Contar.java

Pediremos al usuario un numero, y contaremos desde el uno hasta el numero que nos haya introducido

Contar2.java

Ahora pediremos dos numeros, y contaremos desde el menor hasta el mayor.

Tabla.java

Programa que escribe la tabla de multiplicar de un número introducido por teclado.

ParImpar15.java

Programa que solicita 15 números y nos va indicando si son pares o impares. Al finalizar nos mostrará cuantos son pares y cuantos impares.

Doble.java

Pediremos numeros, y diremos cual es el doble del numero introducido.Pararemos cuando el numero introducido sea cero.

Atras.java

Programa que permite realizar una cuenta atras, desde el numero que introduzca el usuario hasta cero. Lo realizará de tres formas distintas.

- for
- do-while.
- while.

Adivina.java

El usuario "piensa" un número del 1 al 100 y el ordenador lo adivina. (el usurio debe indicarle al ordenador si es mayor, menor o igual):

Menu.java

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.

Factorial.java

Programa que me permite calcular el factorial de una serie de números acabada con la introducción del número -1.

Primo.java

Programa que lee un numero entero y positivo y nos indica si es primo o no.

Tablas.java

Escribir un programa que me ayude a aprender las tablas de multiplicar.

Para ello se irá pidiendo la tabla de multiplicar de un número (pedido por teclado con anterioridad) y comprobando que los valores introducidos son correctos. Si es así el programa escribirá "CORRECTO" y en caso contrario deberá escribir "LO SIENTO, SE HA EQUIVOCADO. LA RESPUESTA CORRECTA ES número".

La última línea mostrará el número de aciertos.