ESTRUCTURAS REPETITIVAS

Programación

[Escuela]
[Título del curso]

Estructura repetitiva

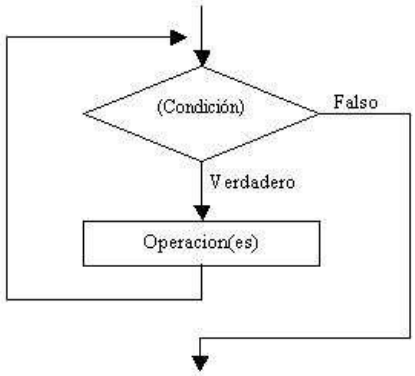
Hasta ahora hemos empleado estructuras SECUENCIALES y CONDICIONALES. Existe otro tipo de estructuras tan importantes como las anteriores que son las estructuras REPETITIVAS.

Una estructura repetitiva permite ejecutar una instrucción o un conjunto de instrucciones varias veces.

Una ejecución repetitiva de sentencias se caracteriza por la o las sentencias que se repiten. El test o prueba de condición antes de cada repetición, que motivará que se repitan o no las sentencias.

Estructura repetitiva "while".

Representación gráfica de la estructura "**while**":



No debemos confundir la representación gráfica de la estructura repetitiva "**while**" (Mientras) con la estructura condicional "if" (Si)

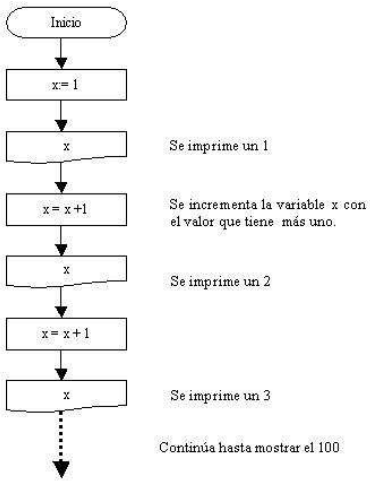
```
while (condition) {
  // code block to be executed
}
```

Funcionamiento: En primer lugar, se verifica la condición, si la misma resulta verdadera se ejecutan las operaciones que indicamos por la rama del Verdadero.
A la rama del verdadero la graficamos en la parte inferior de la condición. Una línea al final del bloque de repetición la conecta con la parte superior de la estructura repetitiva.
En caso que la condición sea Falsa continúa por la rama del Falso y sale de la estructura repetitiva para continuar con la ejecución del algoritmo.

El bloque se repite **MIENTRAS** la condición sea Verdadera.

Importante: Si la condición siempre retorna verdadero estamos en presencia de un ciclo repetitivo infinito. Dicha situación es un error de programación, nunca finalizará el programa.

```
int i = 0;

while (i < 5) {
  System.out.println(i);
  i++;
}
```



Ejemplo 1:

Realizar un programa que imprima en pantalla los números del 1 al 100.

Sin conocer las estructuras repetitivas podemos resolver el problema empleando una estructura secuencial. Inicializamos una variable con el valor 1, luego imprimimos la variable, incrementamos nuevamente la variable y así sucesivamente.

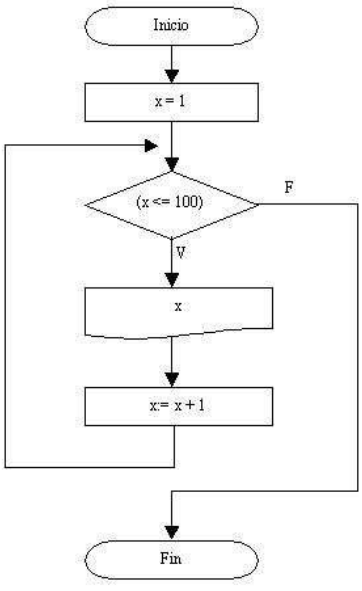


Diagrama de flujo sin "while": Si continuamos con el diagrama no nos alcanzarían las próximas 5 páginas para finalizarlo. Emplear una estructura secuencial para resolver este problema produce un diagrama de flujo y un programa en Java muy largo.

Diagrama de flujo con "while": Ahora veamos la solución empleando una estructura repetitiva "**while**":

Es muy importante analizar este diagrama:

La primera operación inicializa la variable x en 1, seguidamente comienza la estructura repetitiva "**while**" y disponemos la siguiente condición (x <= 100), se lee **MIENTRAS** la variable x sea menor o igual a 100.

Al ejecutarse la condición retorna VERDADERO porque el contenido de x (1) es menor o igual a 100. Al ser la condición verdadera se ejecuta el bloque de instrucciones que contiene la estructura "**while**". El bloque de instrucciones contiene una salida y una operación.
Se imprime el contenido de x, y seguidamente se incrementa la variable x en uno.

La operación x=x + 1 se lee como "en la variable x se guarda el contenido de x más 1". Es decir, si x contiene 1 luego de ejecutarse esta operación se almacenará en x un 2.

Al finalizar el bloque de instrucciones que contiene la estructura repetitiva se verifica nuevamente la condición de la estructura repetitiva y se repite el proceso explicado anteriormente.

Mientras la condición retorne verdadero se ejecuta el bloque de instrucciones; al retornar falso la verificación de la condición se sale de la estructura repetitiva y continua el algoritmo, en este caso finaliza el programa.

Lo más difícil es la definición de la condición de la estructura "**while**" y qué bloque de instrucciones se van a repetir. Observar que si, por ejemplo, disponemos la condición x >=100 (si x es mayor o igual a 100) no provoca ningún error sintáctico pero estamos en presencia de un error lógico porque al evaluarse por primera vez la condición retorna falso y no se ejecuta el bloque de instrucciones que queríamos repetir 100 veces.

No existe una RECETA para definir una condición de una estructura repetitiva, sino que se logra con una práctica continúa solucionando problemas.

Una vez planteado el diagrama debemos verificar si el mismo es una solución válida al problema (en este caso se debe imprimir los números del 1 al 100 en pantalla), para ello podemos hacer un seguimiento del flujo del diagrama y los valores que toman las variables a lo largo de la ejecución:

```
x 1
2
3
4
.
.
100
101
```

Cuando x vale 101 la condición de la estructura repetitiva retorna falso, en este caso finaliza el diagrama.

Importante: Podemos observar que el bloque repetitivo puede no ejecutarse ninguna vez si la condición retorna falso la primera vez.

La variable x debe estar inicializada con algún valor antes que se ejecute la operación x=x + 1 en caso de no estar inicializada aparece un error de compilación.

Programa:

```
public class EstructuraRepetitivaWhile1 {
	public static void main(String[] ar) {
		//Declaraciones
		int x; x=1;

		//Inicio
		while (x<=100) {
			System.out.print(x);
			System.out.print(" - ");
			x = x + 1;
		}
	}//main
}//class
```

Importante: Como podemos observar no hemos creado un objeto de la clase Scanner. Esto debido a que

en este programa no hay que ingresar datos por teclado. Para las salidas utilizamos la función print, que se encuentra creada por defecto en cualquier programa que codifiquemos en Java.

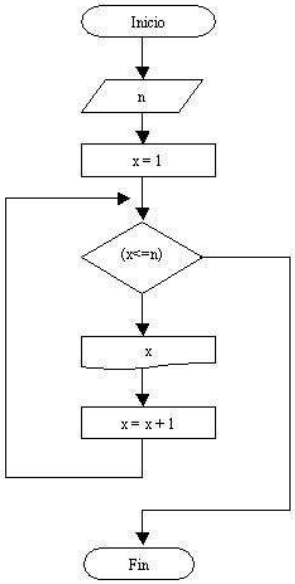
Recordemos que un problema no estará 100% solucionado si no hacemos el programa en Java que muestre los resultados buscados.

Probemos algunas modificaciones de este programa y veamos qué cambios se deberían hacer para:

```
- Imprimir los números del 1 al 500.
- Imprimir los números del 50 al 100.
- Imprimir los números del -50 al 0.
- Imprimir los números del 2 al 100 pero de 2 en 2 (2,4,6,8....100).
```

Respuestas:

```
-  Debemos cambiar la condición del while con x<=500.
-  Debemos inicializar x con el valor 50.
-  Inicializar x con el valor -50 y fijar la condición x<=0.
- Inicializar a x con el valor 2 y dentro del bloque repetitivo incrementar a x
en 2 ( x = x + 2 ).
```



Ejemplo 2:

Escribir un programa que solicite la carga de un valor positivo y nos muestre desde 1 hasta el valor ingresado de uno en uno.

Ejemplo: Si ingresamos 30 se debe mostrar en pantalla los números del 1 al 30.

Es de FUNDAMENTAL importancia analizar los diagramas de flujo y la posterior codificación en Java de los siguientes problemas, en varios problemas se presentan otras situaciones no vistas en el ejercicio anterior.

Diagrama de flujo: Podemos observar que se ingresa por teclado la variable n. El operador puede cargar cualquier valor. Si el operador carga 10 el bloque repetitivo se ejecutará 10 veces, ya que la condición es "si/if" Mientras x<=n?, es decir, mientras x sea menor o igual a 10, pues x comienza en uno y se incrementa en uno cada vez que se ejecuta el bloque repetitivo.

A la prueba del diagrama la podemos realizar dándole valores a las variables; por ejemplo, si ingresamos 5 el seguimiento es el siguiente:

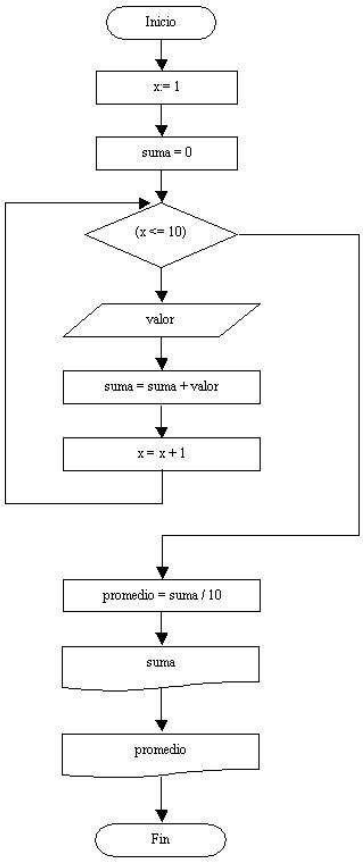
```
n       x
1      (Se imprime el contenido de x)
2            "      "
3            "      "
4            "      "
5            "      "

(Sale del while porque 6 no es menor o igual a 5)
```

Programa:

```
import java.util.Scanner;

public class EstructuraRepetitivaWhile2 {
	public static void main(String[] ar) {
		//Declaraciones
		Scanner teclado=new Scanner(System.in);
		int n,x;

		//Inicio
		System.out.print("Ingrese el valor final:");
		n=teclado.nextInt();
		x=1;
		while (x<=n) {
			System.out.print(x);
			System.out.print(" - ");
			x = x + 1;
		}
	}//main
}//class
```

Los nombres de las variables n y x pueden ser palabras o letras (como en este caso)

La variable x recibe el nombre de CONTADOR. Un contador es un tipo especial de variable que se incrementa o decrementa con valores constantes durante la ejecución del programa.
El contador x nos indica en cada momento la cantidad de valores impresos en pantalla.



Ejemplo 3:

Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio.

Diagrama de flujo: En este problema, a semejanza de los anteriores, llevamos un CONTADOR llamado x que nos sirve para contar las vueltas que debe repetir el "**while**".
También aparece el concepto de ACUMULADOR (un acumulador es un tipo especial de variable que se incrementa o decrementa con valores variables durante la ejecución del programa)

Hemos dado el nombre de suma a nuestro acumulador. Cada ciclo que se repita la estructura repetitiva, la variable suma se incrementa con el contenido ingresado en la variable valor.

La prueba del diagrama se realiza dándole valores a las variables:

```
valor  suma   x      promedio 0    0
(Antes de entrar a la estructura repetitiva estos son los
valores).
       5      5      1
       16     21     2
       7      28     3
       10     38     4
       2      40     5
       20     60     6
       5      65     7
       5      70     8
       10     80     9
       2      82     10
       8      90     11
       9
```

Este es un seguimiento del diagrama planteado. Los números que toma la variable valor dependerá de qué cifras cargue el operador durante la ejecución del programa.
El promedio se calcula al salir de la estructura repetitiva (es decir primero sumamos los 10 valores ingresados y luego los dividimos por 10)

Hay que tener en cuenta que cuando en la variable valor se carga el primer valor (en este ejemplo 5) al cargarse el segundo valor (16) el valor anterior 5 se pierde, por ello la necesidad de ir almacenando en la variable suma los valores ingresados.

Programa:

```
import java.util.Scanner;

public class EstructuraRepetitivaWhile3 {
	public static void main(String[] ar) {
		//Declaraciones
		Scanner teclado=new Scanner(System.in);
		int x,suma,valor,promedio;
		x=1;
		suma=0;

		//Inicio
		while (x<=10) {
			System.out.print("Ingrese un valor:");
			valor=teclado.nextInt();
			suma=suma+valor;
			x=x+1;
		}
		promedio=suma/10;
		System.out.print("La suma de los 10 valores es:");
		System.out.println(suma);
		System.out.print("El promedio es:");
		System.out.print(promedio);
	}//main
}//class
```

Ejemplo 4:

Una planta que fabrica perfiles de hierro posee un lote de n piezas.
Confeccionar un programa que pida ingresar por teclado la cantidad de piezas a procesar y luego ingrese la longitud de cada perfil; sabiendo que la pieza cuya longitud esté comprendida en el rango de 1,20 y 1,30 son aptas. Imprimir por pantalla la cantidad de piezas aptas que hay en el lote.

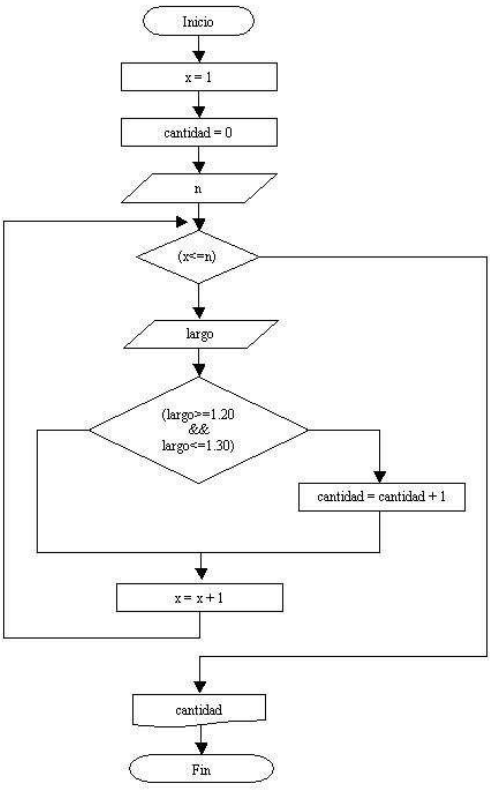


Diagrama de flujo: Podemos observar que dentro de una estructura repetitiva puede haber estructuras condicionales (inclusive puede haber otras estructuras repetitivas que veremos más adelante)

En este problema hay que cargar inicialmente la cantidad de piezas a ingresar (n), seguidamente se cargan n valores de largos de piezas.
Cada vez que ingresamos un largo de pieza (largo) verificamos si es una medida correcta (debe estar entre
1.20 y 1.30 el largo para que sea correcta), en caso de ser correcta la CONTAMOS (incrementamos la variable cantidad en 1)

Al contador cantidad lo inicializamos en cero porque inicialmente no se ha cargado ningún largo de medida.

Cuando salimos de la estructura repetitiva porque se han cargado n largos de piezas mostramos por pantalla el contador cantidad (que representa la cantidad de piezas aptas)

En este problema tenemos dos CONTADORES:

```
x             (Cuenta la cantidad de piezas cargadas hasta el momento)
cantidad      (Cuenta los perfiles de hierro aptos)
```

Programa:

```
import java.util.Scanner;

public class EstructuraRepetitivaWhile4 {
	public static void main(String[] ar) {
		//Declaraciones
		Scanner teclado=new Scanner(System.in);
		int x,cantidad,n;
		float largo;
		x=1;
		cantidad=0;

		//Inicio
		System.out.print("Cuantas piezar procesará:");
		n=teclado.nextInt();
		while (x<=n) {
			System.out.print("Ingrese la medida de la pieza:");
			largo=teclado.nextFloat();
			if (largo>=1.20 && largo<=1.30) {
				cantidad = cantidad +1;
			}
			x=x + 1;
		}//while
		System.out.print("La cantidad de piezas aptas son:");
		System.out.print(cantidad);
	}//main
}//class
```

Problemas propuestos

Ha llegado la parte fundamental, que es el momento donde uno desarrolla individualmente un algoritmo para la resolución de problemas.

El tiempo a dedicar a esta sección EJERCICIOS PROPUESTOS debe ser mucho mayor que el empleado a la sección de EJERCICIOS RESUELTOS. Es de vital importancia para llegar a ser un buen PROGRAMADOR poder resolver problemas en forma individual.

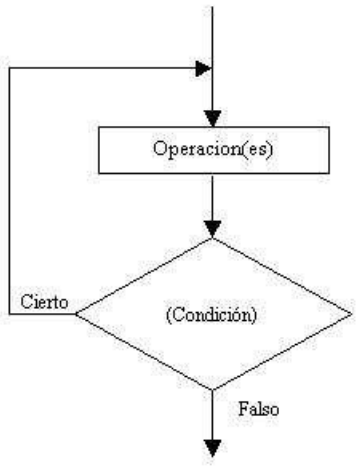
1. Escribir un programa que solicite ingresar 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.
2. Se ingresan un conjunto de n alturas de personas por teclado. Mostrar la altura promedio de las personas.
3. En una empresa trabajan n empleados cuyos sueldos oscilan entre $100 y $500, realizar un programa que lea los sueldos que cobra cada empleado e informe cuántos empleados cobran entre $100 y $300 y cuántos cobran más de $300. Además, el programa deberá informar el importe que gasta la empresa en sueldos al personal.
4. Realizar un programa que imprima 25 términos de la serie 11 - 22 - 33 - 44, etc. (No se ingresan valores por teclado)
5. Mostrar los múltiplos de 8 hasta el valor 500. Debe aparecer en pantalla 8 - 16 - 24, etc.
6. Realizar un programa que permita cargar dos listas de 15 valores cada una. Informar con un mensaje cuál de las dos listas tiene un valor acumulado mayor (mensajes "Lista 1 mayor", "Lista 2 mayor", "Listas iguales")

Tener en cuenta que puede haber dos o más estructuras repetitivas en un algoritmo.

7. Desarrollar un programa que permita cargar n números enteros y luego nos informe cuántos valores fueron pares y cuántos impares. Emplear el operador ?%? en la condición de la estructura condicional:

```
if (valor%2==0)      //Si el if da verdadero luego es par.
```

Estructura repetitiva "do while"



La estructura "**do while**" es otra estructura repetitiva, la cual ejecuta al menos una vez su bloque repetitivo, a diferencia del "while" o del "for" que podían no ejecutar el bloque.

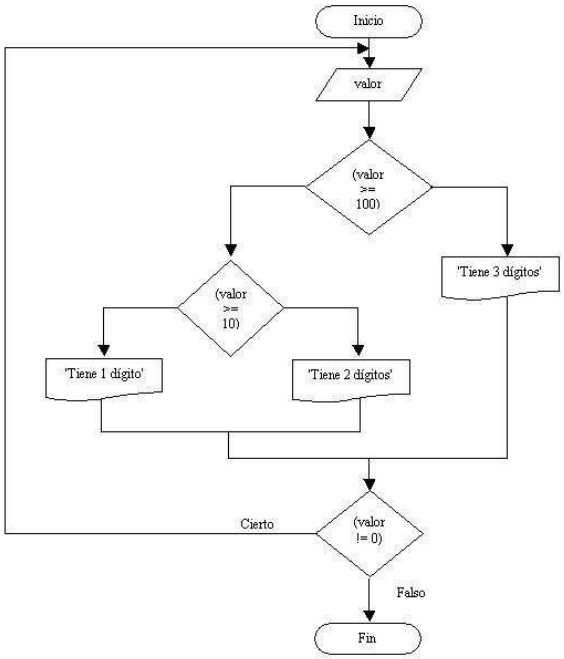
```
do {
  // code block to be executed
}
while (condition);
```

Esta estructura repetitiva se utiliza cuando conocemos de antemano que por lo menos una vez se ejecutará el bloque repetitivo.

La condición de la estructura está abajo del bloque a repetir, a diferencia del while que está en la parte superior.

```
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

Representación gráfica:
El bloque de operaciones se repite MIENTRAS que la condición sea Verdadera.
Si la condición retorna Falso el ciclo se detiene. En Java, todos los ciclos repiten por verdadero y cortan por falso.
Es importante analizar y ver que las operaciones se ejecutan como mínimo una vez.



Ejemplo 5:
Escribir un programa que solicite la carga de un número entre 0 y 999, y nos muestre un mensaje de cuántos dígitos tiene el mismo. Finalizar el programa cuando se cargue el valor 0.

Diagrama de flujo: No hay que confundir los rombos de las estructuras condicionales con los de las estructuras repetitivas do while. En este problema por lo menos se carga un valor. Si se carga un valor mayor o igual a 100 se trata de un número de tres cifras, si es mayor o igual a 10 se trata de un valor de dos dígitos, en caso contrario se trata de un valor de un dígito. Este bloque se repite hasta que se ingresa en la variable valor el número 0 con lo que la condición de la estructura do while retorna falso y sale del bloque repetitivo finalizando el programa.

Programa:

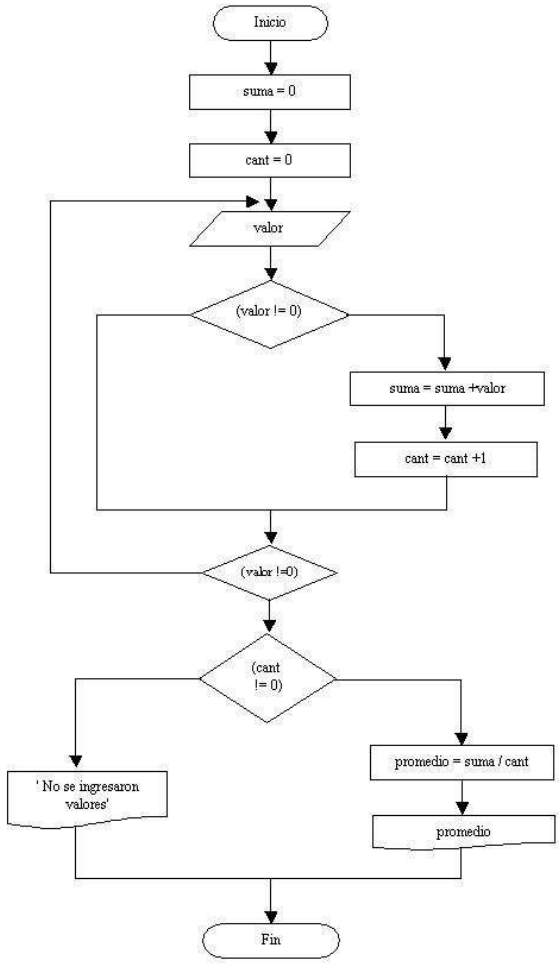
```
import java.util.Scanner;

public  class  EstructuraRepetitivaDoWhile1  {
      public static void main(String[] ar) {
            //Declaraciones
            Scanner                    teclado=new
         Scanner(System.in);
            int valor;

            //Inicio
            do {
                  System.out.print("Ingrese valor entre 0 y 999 (0
            finaliza):");
                  valor=teclado.nextInt();
                  if (valor>=100) {
                        System.out.println("Tiene 3 dígitos.");
                  } else {
                        if (valor>=10) {
                              System.out.println("Tiene 2 dígitos.");
                        } else {
```

```
                                    System.out.println("Tiene 1 dígito.");
                                }
                            }
                        } while (valor!=0);
    }//main
}//class
```

Ejemplo 6:
Escribir un programa que solicite la carga de números por teclado, obtener su promedio. Finalizar la carga de valores cuando se cargue el valor 0.

Cuando la finalización depende de algún valor ingresado por el operador conviene el empleo de la estructura "**do while**", por lo menos se cargará un valor (en el caso más extremo se carga 0, que indica la finalización de la carga de valores)

Diagrama de flujo: Es importante analizar este diagrama de flujo.
Definimos un contador cant que cuenta la cantidad de valores ingresados por el operador (no lo incrementa si ingresamos 0)



El valor 0 no es parte de la serie de valores que se deben sumar.
Definimos el acumulador suma que almacena todos los valores ingresados por teclado.
La estructura repetitiva "**do while**" se repite hasta que ingresamos el valor 0. Con dicho valor la condición del ciclo retorna falso y continúa con el flujo del diagrama.
Disponemos por último una estructura condicional para el caso que el operador cargue únicamente un 0 y por lo tanto no podemos calcular el promedio ya que no existe la división por 0.
En caso que el contador cant tenga un valor distinto a 0 el promedio se obtiene dividiendo el acumulador suma por el contador cant que tiene la cantidad de valores ingresados antes de introducir el 0.

Programa:

```
import java.util.Scanner;

public class EstructuraRepetitivaDoWhile2 {
	public static void main(String[] ar) {
		//Declaaciones
		Scanner teclado=new Scanner(System.in);
		int suma,cant,valor,promedio;
		suma=0; cant=0;

		do {
			System.out.print("Ingrese un valor (0 para finalizar):");
			valor=teclado.nextInt();
			if (valor!=0) { suma=suma+valor;
				cant++;

			}
		} while (valor!=0);

		if (cant!=0) {
			promedio=suma/cant;
			System.out.print("El promedio de los valores ingresados
		es:");
			System.out.print(promedio);
		} else {
			System.out.print("No se ingresaron valores.");}
	}//main
}//class
```

El contador cant DEBE inicializarse antes del ciclo, lo mismo que el acumulador suma. El promedio se calcula siempre y cuando el contador cant sea distinto a 0.

Ejemplo 7:
Realizar un programa que permita ingresar el peso (en kilogramos) de piezas. El proceso termina cuando ingresamos el valor 0. Se debe informar:

a) Cuántas piezas tienen un peso entre 9.8 Kg. y 10.2 Kg.?, cuántas con más de 10.2 Kg.? y cuántas con menos de 9.8 Kg.?

b) La cantidad total de piezas procesadas.

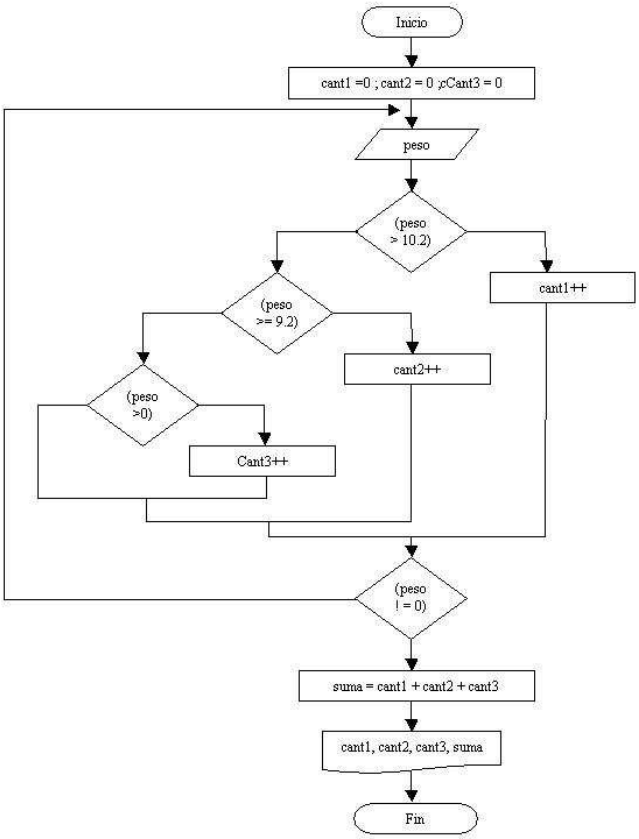


Diagrama de flujo: Los tres contadores cont1, cont2, y cont3 se inicializan en 0 antes de entrar a la estructura repetitiva. A la variable suma no se la inicializa en 0 porque no es un acumulador, sino que guarda la suma del contenido de las variables cont1, cont2 y cont3.

La estructura se repite hasta que se ingresa el valor 0 en la variable peso. Este valor no se lo considera un peso menor a 9.8 Kg., sino que indica que ha finalizado la carga de valores por teclado.

Programa:

```
import java.util.Scanner;

public class EstructuraRepetitivaDoWhile3 {
     public static void main(String[] ar) {
          //Declaraciones
          Scanner teclado=new Scanner(System.in);
          int cant1=0,cant2=0,cant3=0,suma=0;
          float
          peso;


          //Inicio
          do {
               System.out.print("Ingrese el peso de la pieza (0 para
                    finalizar): ");
               peso=teclado.nextFloat();
               if (peso>10.2) {
                    cant1++;
               } else {
                    if (peso>=9.8)
                         { cant2++;
                    } else {
                         if (peso>0) {
                              cant3++
                              ;
                         }
                    }
               }
          } while(peso!=0);

          //Imprimimos estadisticas
          suma=cant1+cant2+cant3;
          System.out.print("Piezas aptas:");
          System.out.println(cant2);
          System.out.print("Piezas con un peso superior a 10.2:");
          System.out.println(cant1);
          System.out.print("Piezas con un peso inferior a 9.8:");
          System.out.println(cant3);

     }//main
}//class
```

Problemas propuestos

8. Realizar un programa que acumule (sume) valores ingresados por teclado hasta ingresar el 9999 (no sumar dicho valor, indica que ha finalizado la carga). Imprimir el valor acumulado e informar si dicho valor es cero, mayor a cero o menor a cero.

9. En un banco se procesan datos de las cuentas corrientes de sus clientes. De cada cuenta corriente se conoce: Nombre, Apellidos y CCC (número de cuenta), datos que introducirá el usuario. El salado parte de 0 y se va calculado mientras se van introduciendo ingresos y gastos con un bucle, el cual finalizar al ingresar un valor de importe 0, lo que provocará la impresión de los daros de la cuenta, (Nombre, Apellidos, CCC e importe. Una vez mostrada la información, preguntará al usuario si quiere calcular el estado de otra cuenta contestando este con s/n.

```
Introduce tu nombre:      --------
Introduce tus apellidos:  --------
Introduce tu CCC: ES 62 9999 1234 96 1234567890


n° cuenta:  ES 62 9999 1234 96 1234567890


Nombre:     --------
Apellidos:  --------
CCC:        ES 62 9999 1234 96 1234567890
Importe:    21.345€
```

10. Realizar un programa que imprima por pantalla la se secuencia Fibonacci. Este programa solicitará por pantalla la cantidad de Fibonacci.

El número Fibonacci se calcula con la suma de los dos números Fibonacci anteriores partiendo de 0 y 1, los siguientes son 1, 2, 3, 5, 8, 13, 21 ...

Estructura repetitiva “for”

Cualquier problema que requiera una estructura repetitiva se puede resolver empleando la estructura “while”. Pero hay otra estructura repetitiva cuyo planteo es más sencillo en ciertas situaciones.

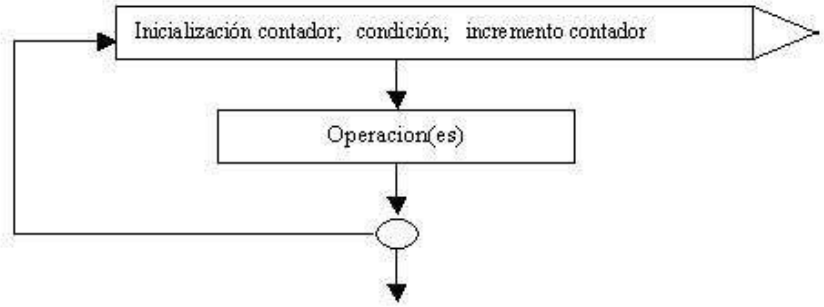
En general, la estructura “for” se usa en aquellas situaciones en las cuales CONOCEMOS la cantidad de veces que queremos que se ejecute el bloque de instrucciones.

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

```
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}
```

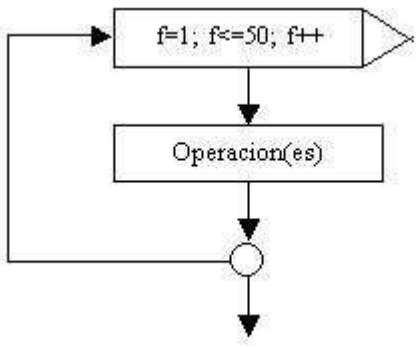
Ejemplo: cargar 10 números, ingresar 5 notas de alumnos, etc. Conocemos de antemano la cantidad de veces que queremos que el bloque se repita. Veremos, sin embargo, que en el lenguaje Java la estructura “for” puede usarse en cualquier situación repetitiva, porque en última instancia no es otra cosa que una estructura “while” generalizada.

Representación gráfica:



En su forma más típica y básica, esta estructura requiere una variable entera que cumple la función de un CONTADOR de vueltas. En la sección indicada como "inicialización contadora", se suele colocar el nombre de la variable que hará de contador, asignándole a dicha variable un valor inicial. En la sección de "condición" se coloca la condición que deberá ser verdadera para que el ciclo continúe (en caso de un falso, el ciclo se detendrá). Y finalmente, en la sección de "incremento contador" se coloca una instrucción que permite modificar el valor de la variable que hace de contador (para permitir que alguna vez la condición sea falsa)

Cuando el ciclo comienza, antes de dar la primera vuelta, la variable del “for” toma el valor indicado en la sección de de "inicialización contadora". Inmediatamente se verifica, en forma automática, si la condición es verdadera. En caso de serlo se ejecuta el bloque de operaciones del ciclo, y al finalizar el mismo se ejecuta la instrucción que se haya colocado en la tercera sección.
Seguidamente, se vuelve a controlar el valor de la condición, y así prosigue hasta que dicha condición entregue un falso.



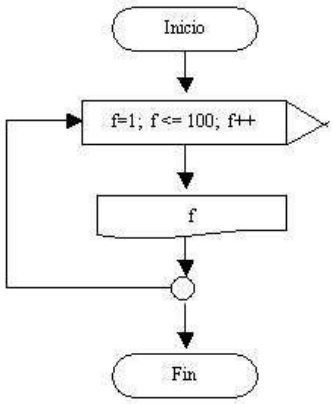
Si conocemos la cantidad de veces que se repite el bloque es muy sencillo emplear un “for”, por ejemplo si queremos que se repita 50 veces el bloque de instrucciones puede hacerse así:

La variable del “for” puede tener cualquier nombre. En este ejemplo se la ha definido con el nombre f. Analicemos el ejemplo:

- La variable f toma inicialmente el valor 1.
- Se controla automáticamente el valor de la condición: como f vale 1 y esto es menor
que 50, la condición da verdadero.
- Como la condición fue verdadera, se ejecutan la/s operación/es.

- Al finalizar de ejecutarlas, se retorna a la instrucción f++, por lo que la variable f se incrementa en uno.
- Se vuelve a controlar (automáticamente) si f es menor o igual a 50. Como ahora su valor es 2, se ejecuta nuevamente el bloque de instrucciones e incrementa nuevamente la variable del for al terminar el mismo.
- El proceso se repetirá hasta que la variable f sea incrementada al valor 51.
En este momento la condición será falsa, y el ciclo se detendrá.

La variable f PUEDE ser modificada dentro del bloque de operaciones del for, aunque esto podría causar problemas de lógica si el programador es inexperto.
La variable f puede ser inicializada en cualquier valor y finalizar en cualquier valor. Además, no es obligatorio que la instrucción de modificación sea un incremento del tipo contador (f++).
Cualquier instrucción que modifique el valor de la variable es válida. Si por ejemplo se escribe f=f+2 en lugar de f++, el valor de f será incrementado de a 2 en cada vuelta, y no de a 1. En este caso, esto significará que el ciclo no efectuará las 50 vueltas sino sólo 25.



Ejemplo 1:
Realizar un programa que imprima en pantalla los números del 1 al 100.

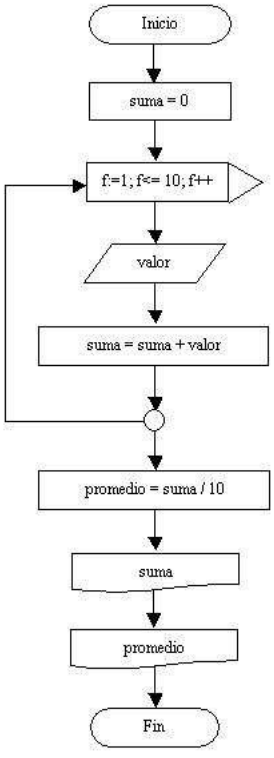
Diagrama de flujo: Podemos observar y comparar con el problema realizado con el while. Con la estructura while el CONTADOR x sirve para contar las vueltas. Con el for el CONTADOR f cumple dicha función. Inicialmente f vale 1 y como no es superior a 100 se ejecuta el bloque, imprimimos el contenido de f, al finalizar el bloque repetitivo se incrementa la variable f en 1, como 2 no es superior a 100 se repite el bloque de instrucciones.

Cuando la variable del "for" llega a 101 sale de la estructura repetitiva y continúa la ejecución del algoritmo que se indica después del círculo.

La variable f (o como sea que se decida llamarla) debe estar definida como una variable más.

Programa:

```
public class EstructuraRepetitivaFor1 { public
     static void main(String[] ar) {
          int f;

          for(f=1;f<=100;f++) {
               System.out.print(f);
               System.out.print("-");
          }

     }//main
}//class
```



Ejemplo 2:
Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio. Este problema ya lo desarrollamos , lo resolveremos empleando la estructura for.

Diagrama de flujo:

En este caso, a la variable del for (f) sólo se la requiere para que se repita el bloque de instrucciones 10 veces.

Programa:

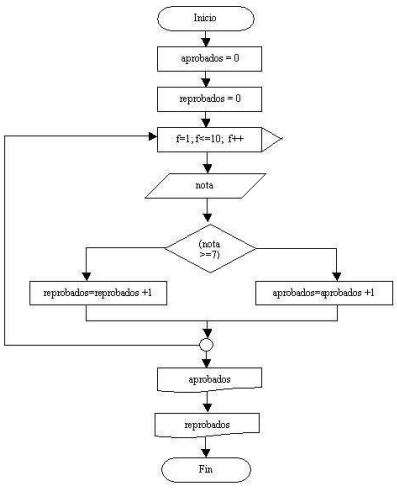
```
import java.util.Scanner;

public class EstructuraRepetitivaFor2 { public
     static void main(String[] ar) {
          Scanner teclado=new Scanner(System.in);
          int suma,f,valor,promedio;



          suma=0;
```

```
        for(f=1;f<=10;f++) {
            System.out.print("Ingrese    valor:");
            valor=teclado.nextInt();
            suma=suma+valor;
        }
        System.out.print("La  suma  es:");
        System.out.println(suma);

        promedio=suma/10;
        System.out.print("El  promedio  es:");
        System.out.print(promedio);
    }
}
```

El problema requiere que se carguen 10 valores y se sumen los mismos.
Tener en cuenta encerrar entre llaves bloque de instrucciones a repetir dentro del for. El promedio se calcula fuera del for luego de haber cargado los 10 valores.

Ejemplo 3:

Escribir un programa que lea 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.



Para resolver este problema se requieren tres contadores:

```
aprobados   (Cuenta   la   cantidad   de   alumnos
aprobados)  reprobados  (Cuenta  la  cantidad  de
reprobados)
f (es el contador del for)
```

Dentro de la estructura repetitiva debemos hacer la carga de la variable nota y verificar con una estructura condicional si el contenido de la variable nota es mayor o igual a 7 para incrementar el contador aprobados, en caso de que la condición retorne falso debemos incrementar la variable reprobados.

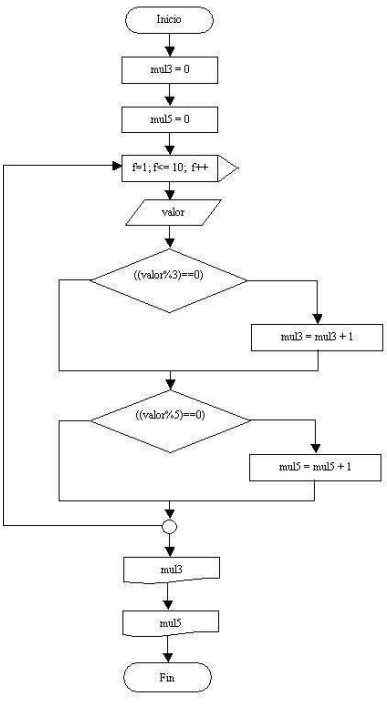
Diagrama de flujo: Los contadores aprobados y reprobados deben imprimirse FUERA de la estructura repetitiva.

Es fundamental inicializar los contadores aprobados y reprobados en cero antes de entrar a la estructura for.

Importante: Un error común es inicializar los contadores dentro de la estructura repetitiva. En caso de hacer esto los contadores se fijan en cero en cada ciclo del for, por lo que al finalizar el for como máximo el contador puede tener el valor 1.

Programa:

```
import java.util.Scanner;

public   class   EstructuraRepetitivaFor3   {
    public static void main(String[] ar) {
        Scanner teclado=new Scanner(System.in);
        int aprobados,reprobados,f,nota;
        aprobados=0;
        reprobados=0;

        for(f=1;f<=10;f++) { System.out.print("Ingrese la
            nota:"); nota=teclado.nextInt();
            if (nota>=7) {
                aprobados=aprobados+1;
            } else {
                reprobados=reprobados+1;
            }
        }
        System.out.print("Cantidad   de   aprobados:");
        System.out.println(aprobados);
        System.out.print("Cantidad   de   reprobados:");
        System.out.print(reprobados);
    }
}
```



Ejemplo 4:

Escribir un programa que lea 10 números enteros y luego muestre cuántos

valores ingresados fueron múltiplos de 3 y cuántos de 5. Debemos tener en cuenta que hay números que son múltiplos de 3 y de 5 a la vez.

Diagrama de flujo: Tengamos en cuenta que el operador matemático % retorna el resto de dividir un valor por otro, en este caso: valor%3 retorna el resto de dividir el valor que ingresamos por teclado, por tres.

Veamos: si ingresamos 6 el resto de dividirlo por 3 es 0, si ingresamos 12 el resto de dividirlo por 3 es 0. Generalizando: cuando el resto de dividir por 3 al valor que ingresamos por teclado es cero, se trata de un múltiplo de dicho valor.

Ahora bien ¿por qué no hemos dispuesto una estructura if anidada? Porque hay valores que son múltiplos de 3 y de 5 a la vez. Por lo tanto, con "if" anidados no podríamos analizar los dos casos.
Es importante darse cuenta cuando conviene emplear if anidados y cuando no debe emplearse.

Programa:

```
import java.util.Scanner;

public   class   EstructuraRepetitivaFor4   {
     public static void main(String[] ar) {
          Scanner teclado=new Scanner(System.in);
          int mul3, mul5, valor, f;

          mul3=0;
          mul5=0;

          for(f=1;f<=10;f++) { System.out.print("Ingrese un valor:");
               valor=teclado.nextInt();
               if  (valor%3==0)  {
                    mul3=mul3+1;
               }
               if  (valor%5==0)  {
                    mul5=mul5+1;
               }
          }
          System.out.print("Cantidad de valores ingresados múltiplos de
  3:");
          System.out.print(mul3);
          System.out.print("Cantidad de valores ingresados múltiplos de
  5:");
          System.out.print(mul3);

      }//main
 }//Class
```

Problema 5:

Escribir un programa que lea n números enteros y calcule la cantidad de valores mayores o iguales a 1000.

Este tipo de problemas también se puede resolver empleando la estructura repetitiva "for". Lo primero que se hace es cargar una variable que indique la cantidad de valores a ingresar. Dicha variable se carga antes de entrar a la estructura repetitiva "for".

La estructura "for" permite que el valor inicial o final dependa de una variable cargada previamente por teclado.

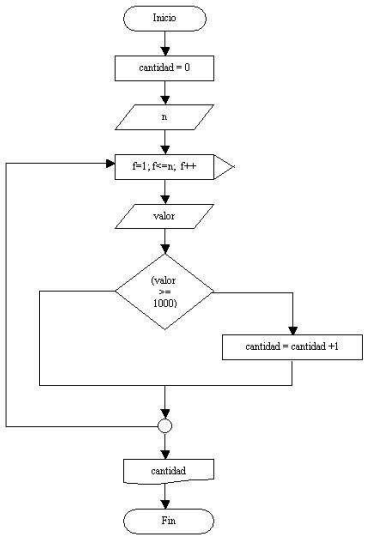


Diagrama de flujo:

Tenemos un contador llamado cantidad y f que es el contador del "for".
La variable entera n se carga previo al inicio del "for", por lo que podemos fijar el valor final del "for" con la variable n.

Por ejemplo si el operador carga 5 en n la estructura repetitiva "for" se ejecutará 5 veces.

La variable valor se ingresa dentro de la estructura repetitiva, y se verifica si el valor de la misma es mayor o igual a 1000, en dicho caso se incrementa en uno el contador cantidad.

Fuera de la estructura repetitiva imprimimos el contador cantidad que tiene almacenado la cantidad de valores ingresados mayores o iguales a 1000.

Programa:

```
import java.util.Scanner;

public class EstructuraRepetitivaFor5 { public
     static void main(String[] ar) {

          Scanner teclado=new Scanner(System.in);
          int cantidad,n,f,valor;
          cantidad=0;
          System.out.print("Cuantos   valores   ingresará:");
          n=teclado.nextInt();
          for(f=1;f<=n;f++)  {  System.out.print("Ingrese
               el valor:"); valor=teclado.nextInt();
               if (valor>=1000) {
                    cantidad=cantidad+1;
               }
          }
          System.out.print("La  cantidad  de  valores  ingresados  mayores  o
iguales a 1000 son:");
          System.out.print(cantidad);
     }
}
```

Mas ejemplos:

```
for (int i = 0; i < 5; i++) {
     System.out.println(i);
}
```

```
for (int i = 0; i <= 10; i = i + 2) {
     System.out.println(i);
}
```

```
// Outer loop
for (int i = 1; i <= 2; i++) {
     System.out.println("Outer: " + i); // Executes 2 times

     // Inner loop
     for (int j = 1; j <= 3; j++) {
          System.out.println(" Inner: " + j);  // Executes 6 times (2 * 3)
     }
}
```

Problemas propuestos

Ha llegado nuevamente la parte fundamental, que es el momento donde uno desarrolla individualmente un algoritmo para la resolución de un problema.

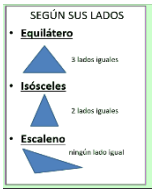
11. Confeccionar un programa que lea n pares de datos, cada par de datos corresponde a la medida de la base y la altura de un triángulo. El programa deberá informar:
    a. De cada triángulo la medida de su base, su altura y su superficie.
    b. La cantidad de triángulos cuya superficie es mayor a 12.

12. Desarrollar un programa que solicite la carga de 10 números e imprima la suma de los últimos 5 valores ingresados.

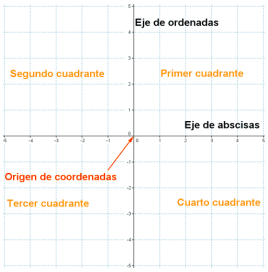
13. Desarrollar un programa que muestre la tabla de multiplicar del 5 (del 5 al 50)

14. Confeccionar un programa que permita ingresar un valor del 1 al 10 y nos muestre la tabla de multiplicar del mismo (los primeros 12 términos)
Ejemplo: Si ingreso 3 deberá aparecer en pantalla los valores 3, 6, 9, hasta el 36.

15. Realizar un programa que lea los lados de n triángulos, e informar:
    a. De cada uno de ellos, qué tipo de triángulo es: equilátero (tres lados iguales), isósceles (dos lados iguales), o escaleno (ningún lado igual)
    b. Cantidad de triángulos de cada tipo.
    c. Tipo de triángulo que posee menor cantidad.



16. Escribir un programa que pida ingresar coordenadas (x,y) que representan puntos en el plano. Informar cuántos puntos se han ingresado en el primer, segundo, tercer y cuarto cuadrante. Al comenzar el programa se pide que se ingrese la cantidad de puntos a procesar.



17. Realiza la carga de 10 valores enteros por teclado. Se desea conocer:
    a. La cantidad de valores ingresados negativos.
    b. La cantidad de valores ingresados positivos.
    c. La cantidad de múltiplos de 15.
    d. El valor acumulado de los números ingresados que son pares

18. Haz un duplicado del ejercicio 15 y modifícalo para que termine si uno de los lados es negativo.
19. Haz un duplicado del ejercicio 16 y modifícalo para que termine si el punto está en el origen.
20. Haz un duplicado del ejercicio 17 y modifícalo para que termine si se introduce el cero.