

Estructuras de Condicional y Bucles



Encarnación Sánchez Gallego

Índice



1. Introducción
2. **Programación estructurada .**
3. Estructuras de selección.
4. Estructuras condicionales



Teorema del programa estructurado



El teorema del programa estructurado establece que toda función computable puede ser implementada en un lenguaje de programación que combine sólo tres estructuras lógicas. Esas tres formas (también llamadas estructuras de control) específicamente son:

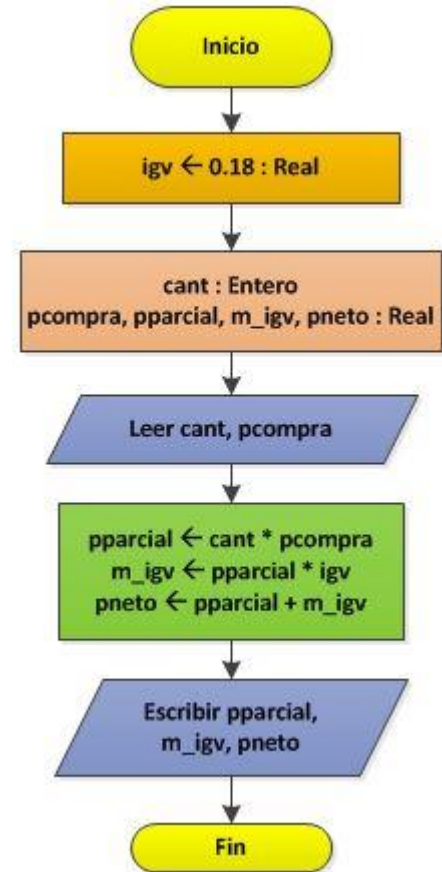
- La estructura de secuencia
- La estructura de selección
- La estructura de repetición

El teorema del programa estructurado fue enunciado por Bohm y Jacopini en 1966.



Repaso de estructuras.

Estructuras Secuenciales

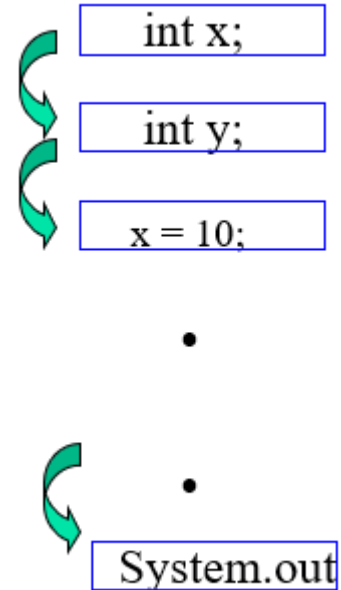


- Son estructuras que obedecen la linealidad, es decir, no admiten saltos
- A la ejecución de una instrucción le sigue otra o la siguiente instrucción y así sucesivamente.
- Mantiene la secuencia de acciones en el orden en el que aparecen las instrucciones.

Estructuras Secuenciales

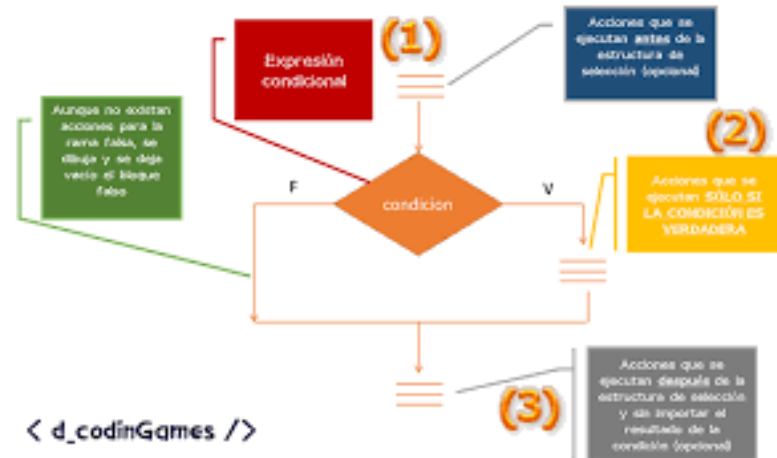
Realiza un programa que lea dos variables (x e y) y calcule la resta de x-y y haga la división de x entre 4.

```
public class Secuencial
{
    public static void main(String [] args)
    {
        int x;
        int y;
        x = 10;
        y = 2;
        x = x - y;
        y = x/4;
        System.out.println("El valor de x es = " + x);
        System.out.println("El valor de y es = " + y);
    }
}
```



Índice

1. Introducción
2. Programación estructurada .
3. Estructuras de selección.
4. Estructuras condicionales



Estructuras de Selección. Sentencia IF



Esta sentencia de control permite ejecutar o no una sentencia simple o compuesta según se cumpla o no una determinada condición. Esta sentencia tiene la siguiente forma general:

Explicación:

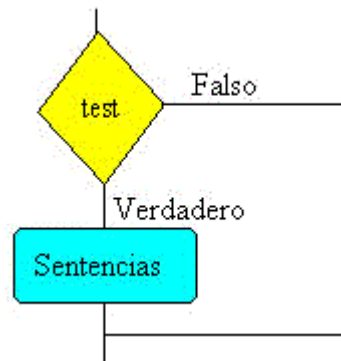
```
if (condición) {  
    sentencia;  
}
```

Se evalúa la condición.

Si el resultado es true , se ejecuta sentencia; si el resultado es false, se salta sentencia

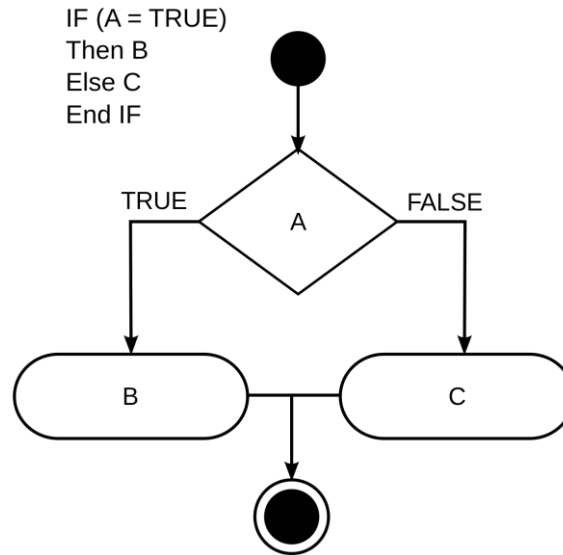
y se prosigue en la línea siguiente.

Hay que recordar que sentencia puede ser una sentencia simple o compuesta (bloque { ...}).



Estructuras de Selección.Sentencia if ... else

Esta sentencia permite realizar una bifurcación, ejecutando una parte u otra del programa según se cumpla o no una cierta condición.



Estructuras de Selección.Sentencia if ... else



```
if(condición) {
```

```
    sentencia_1;
```

```
}
```

```
else {
```

```
    sentencia_2;
```

```
}
```

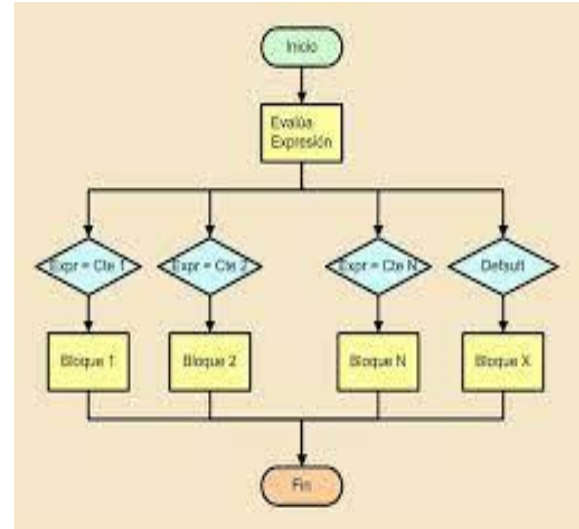
1. Se evalúa expresión.
2. Si el resultado es true, se ejecuta sentencia_1 y se prosigue en la línea siguiente a sentencia_2; si el resultado es false, se salta sentencia_1, se ejecuta sentencia_2 y se prosigue en la línea siguiente.
3. Hay que indicar aquí también que sentencia_1 y sentencia_2 pueden ser sentencias simples o compuestas (bloques { ... }).



Estructuras de Selección. Sentencia if ... else múltiple

Esta sentencia permite realizar una ramificación múltiple, ejecutando una entre varias partes del programa según se cumpla una entre n condiciones. La forma general es la siguiente:

```
if (expresión_1) {  
    sentencia_1;  
}  
else if (expresión_2) {  
    sentencia_2;  
}  
else if (expresión_3) {  
    sentencia_3;  
}  
else if (...) {  
    ...  
}  
else {  
    sentencia_n  
}
```



Estructuras de Selección. Sentencia if ... else múltiple



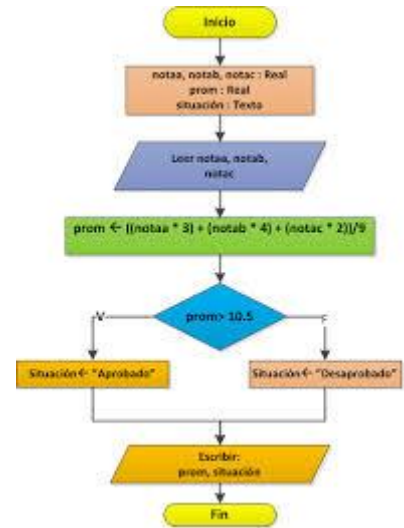
- Se evalúa expresión_1.
- Si el resultado es true, se ejecuta sentencia_1.
- Si el resultado es false, se salta sentencia_1 y se evalúa expresión_2.
- Si el resultado es true se ejecuta sentencia_2, mientras que si es false se evalúa expresión_3 y así sucesivamente.
- Si ninguna de las expresiones o condiciones es true se ejecuta sentencia_n que es la opción por defecto.
- Todas las sentencias pueden ser simples o compuestas.

```
if (expresión_1) {  
    sentencia_1;  
} else if (expresión_2) {  
    sentencia_2;  
}  
else if (expresión_3)  
    sentencia_3;  
}  
else if (...) {  
    ...  
}  
else {  
    sentencia_n  
}
```



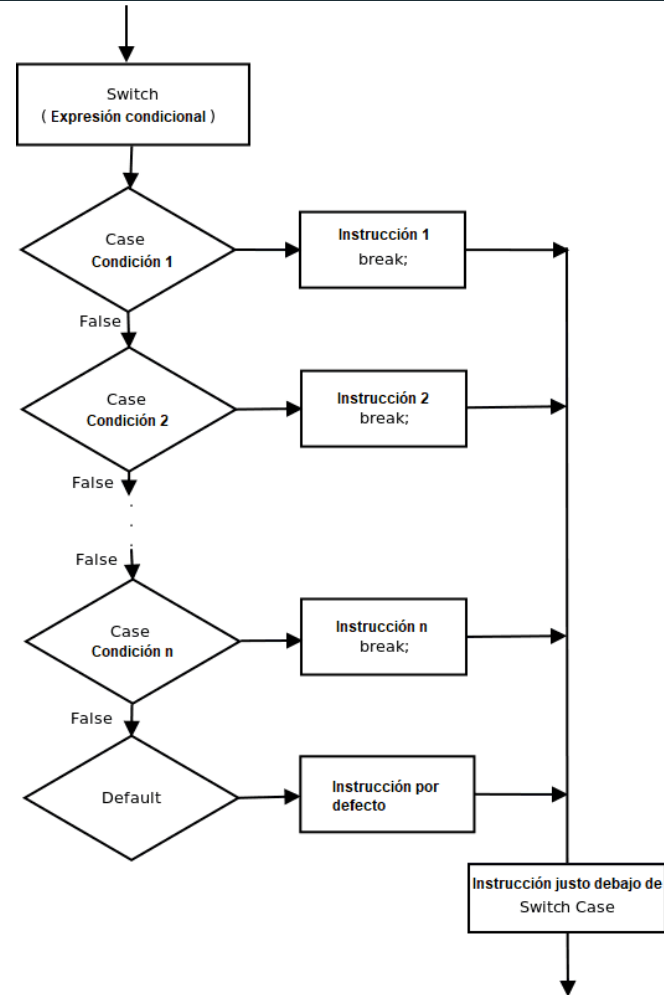
IF anidados

```
if (a >= b) {  
    if (b != 0.0)  
        c = a/b;  
}  
else  
    c = 0.0;
```



Sentencia switch

- La sentencia que se va a describir a continuación desarrolla una función similar a la de la sentencia if ... else con múltiples ramificaciones, aunque como se puede ver presenta también importantes diferencias.





La forma general de la sentencia switch es la siguiente:

```
switch (expresión o variable) {  
    case const1:  
        sentencia_1;  
    case const2:  
        sentencia_2;  
    ...  
    case constn:  
        sentencia_n;  
    [default: sentencia;]  
}
```



Ejemplo switch

```
switch (day)
{
    case 1:  dayString = "Lunes";
             break;
    case 2:  dayString = "Martes";
             break;
    case 3:  dayString = "Miercoles";
             break;
    case 4:  dayString = "Jueves";
             break;
    case 5:  dayString = "Viernes";
             break;
    case 6:  dayString = "Sabado";
             break;
    case 7:  dayString = "Domingo";
             break;
    default: dayString = "Dia inválido";
             break;
}
System.out.println(dayString);
}
```



Resumen. Estructuras de Selección



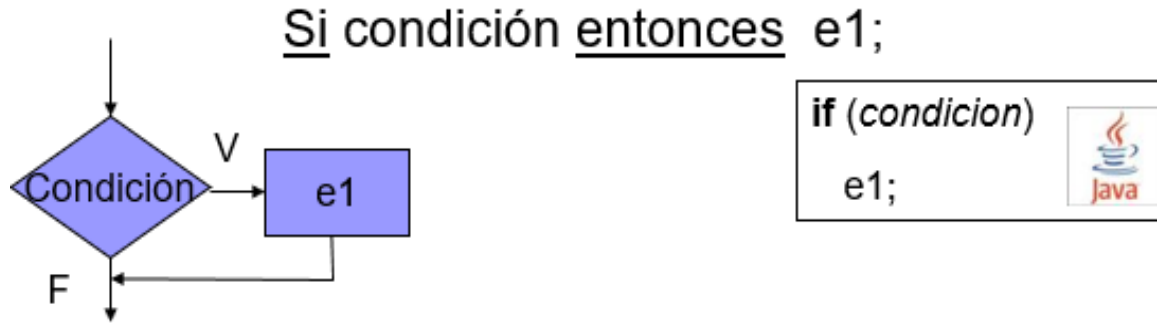
- `if (expresion booleana)`
 `instrucción`
- `if (expresion booleana)`
 `instrucción1`
 `else`
 `instruccion2`
- `switch (expresion)`
 {
 `case expresión-constantel: instrucciones; break;`
 `case expresión-constante2: instrucciones; break;`
 `...`
 `default: instrucciones`
 }
 - La expresion puede ser de tipo char, byte, short o int





Estructuras de Selección. Selección de opción única

- Decisiones sencillas. Condición booleana





Realiza un programa que lea un número por pantalla y que compruebe si es negativo

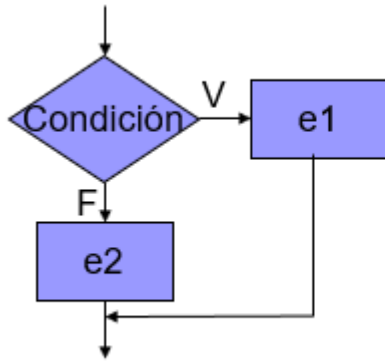
```
{  
    int N;  
    Teclado teclado = new Teclado();  
    System.out.print("Número: ");  
    N = teclado.readInt();  
    if (N < 0) System.out.println("negativo");  
}
```



Estructuras de Selección. Selección de dos opciones



- Decisiones sencillas. Condición booleana.



Si condición
entonces e1;
otro e2;

```
if (condicion)
    e1;
else
    e2;
```



Realizar un programa que compare dos números a y b. Si a es mayor que b que haga la resta de (a-b) y si b es mayor que a que multiplique a por b.



```
public class Concatenacion_If
{
    public static void main(String [] args)
    {   int a, b;
        a = 1; b = 5;
        if (a > b) { // por Verdad
            a = a - b;
            System.out.println("El valor de A es = " + a);
        }
        else { // por Falso
            b = b*a;
            System.out.println("El valor de B es = " + b);
        }
    }
}
```

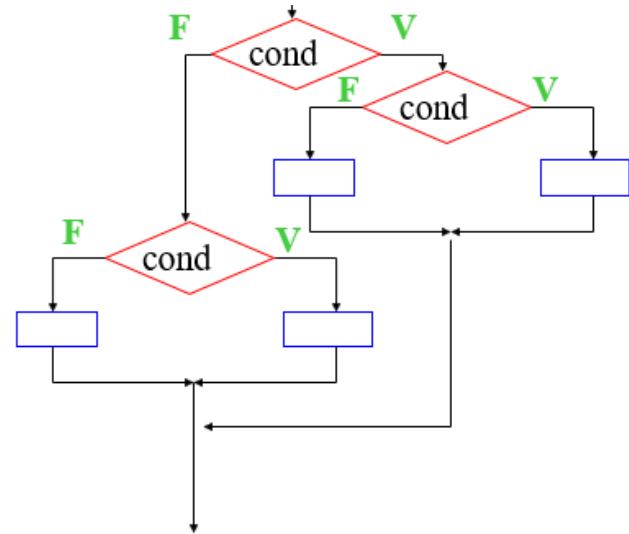


Estructuras de Selección. Selección múltiple



- Selección de varias posibilidades bajo diferentes condiciones

```
if (condicion1)
    e1;
else if (condicion2)
    e2;
else if (condicion3)
    e3;
```



Realiza un programa que lea por teclado una calificación y si es menor que 5 que de No apto, si es mayor de 5 y menor de 6 que de suficiente, si es mayor de 6 y menor de 9 que de bien y sino que de muy bien.



```
{  
    double promedio;  
    String calificacion  
    Teclado teclado = new Teclado();  
    System.out.print("promedio final: ");  
    promedio = teclado.readDouble();  
    if (promedio < 5) calificacion="NA";  
    else if (promedio < 6) calificacion="S";  
        else if (promedio < 9) calificacion="B";  
            else calificacion = "MB";  
    System.out.println("cal= "+calificacion);  
}
```



Estructuras de Selección. Selección Múltiple



Cuando en una condición existen diversas posibilidades nos vemos obligados a utilizar if anidados, lo que complica la realización y depuración del código.

Para facilitar esta situación se tiene la **condicional switch** que permite definir un número ilimitado de ramas basadas en una misma condición.

```
switch (Expresión) {  
    case valor1: //Instrucciones  
        break;  
    default: //Instrucciones  
        break;  
}
```

Estructura selectiva → switch (Expresión) {

Variable a comparar → case valor1:

Caso1 → //Instrucciones

Instrucciones si se cumple el caso 1 → break;

Caso default → default: //Instrucciones

Instrucciones del caso por defecto → break;

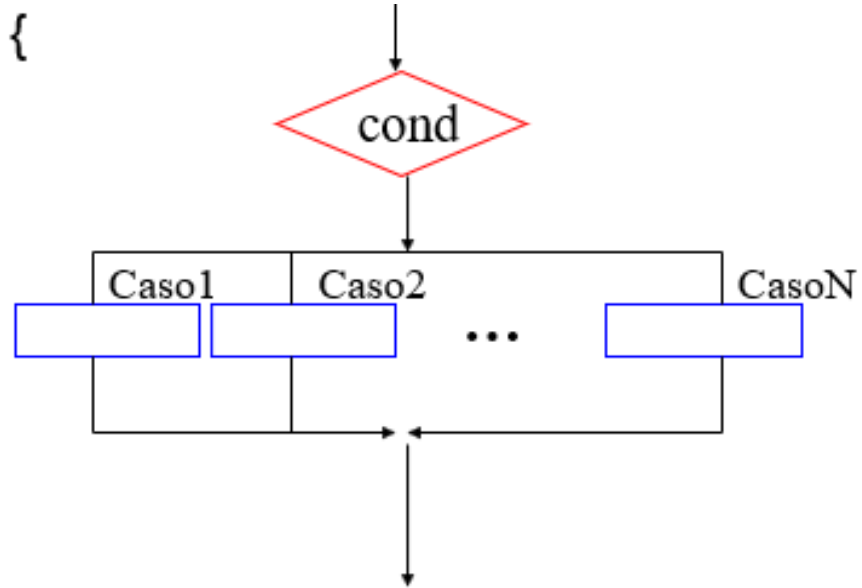
Salir del Switch → }



Estructuras de Selección. Selección Múltiple



```
switch (condición) {  
    case literal1:  
        instrucciones;  
        break;  
    case literal2:  
        instrucciones;  
        break;  
    ...  
    default  
        bloque;  
}
```



Estructuras de Selección. Selección Múltiple



Realiza un programa que lea una letra y que si es la A, que salga un mensaje es la vocal A, si es la e que lance otro programa diciendo que es la vocal e y si no que diga que es otro caracter.

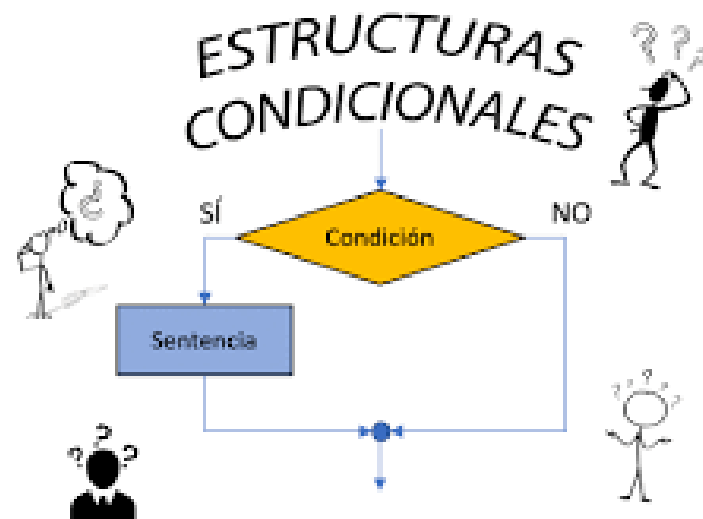
```
public class Switch{  
    public static void main (String [] args) {  
        char letra='p';  
        switch(letra) {  
            case 'a':    case 'A':  
                System.out.println("Es la vocal a, que viene de Araña");  
                break;  
            case 'e':    case 'E':  
                System.out.println("Es la vocal e, que viene de Elefante\n");  
                break;  
            ...  
            default: System.out.println("Es otra caracter\n");  
                } // switch  
        } // main  
    }
```



Índice



1. Introducción
2. Programación estructurada .
3. Estructuras de selección.
- 4. Estructuras condicionales**

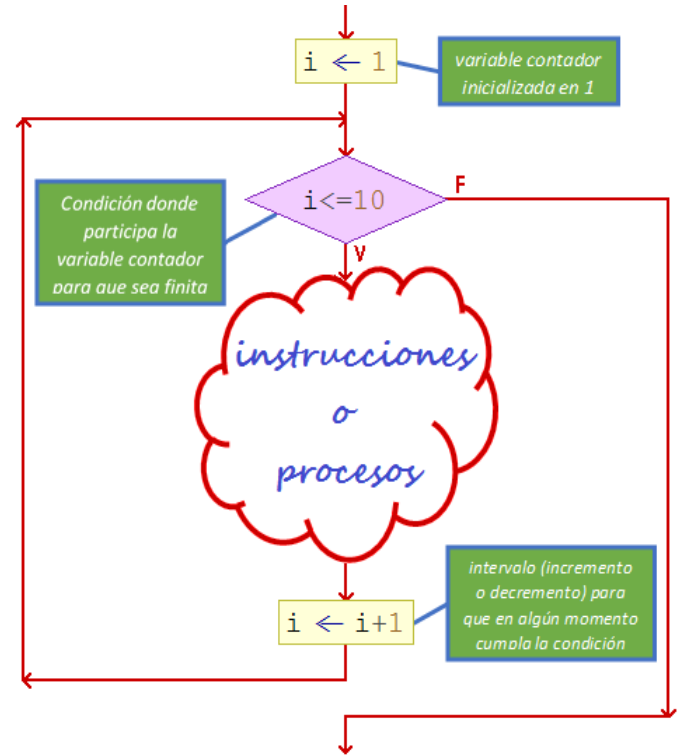


Estructuras de Repetición.



Tipos de Bucles

- **Bucles controlador por contador:** Se repite un número conocido de veces.
- **Bucles controlados por suceso:** Se repite hasta que ocurre algo dentro del cuerpo del bucle que provoca la terminación del proceso de iteración, es decir, indica que se debe salir del bucle.



Estructuras de Repetición. Bucle For (controlado por contador)

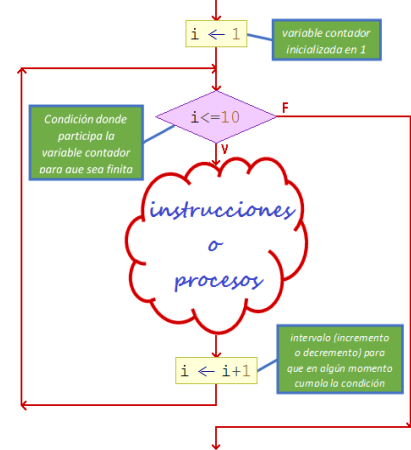


Cuando se desea ejecutar un grupo de instrucciones un **número determinado de veces**, es mejor utilizar la sentencia for.

- Hace uso de una variable de tipo contador para control del bucle

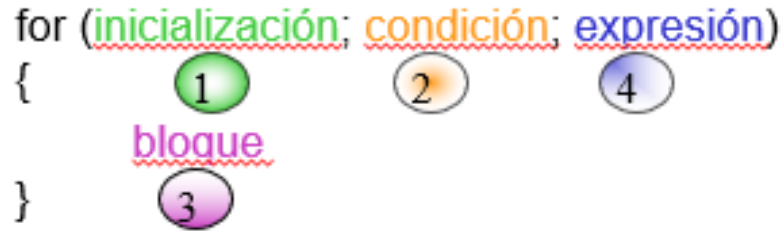
Hay tres operaciones que se realizan siempre en este tipo de bucles

- Inicializar el contador
- Evaluar el contador en la condición del bucle
- Variar o actualizar el contador.

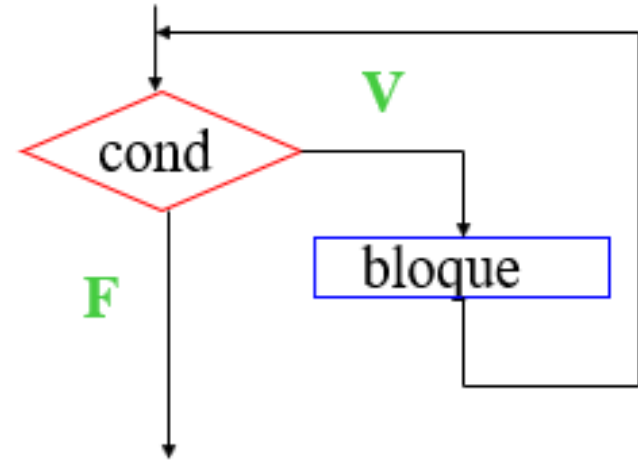


Estructuras de Repetición. Bucle For (controlado por contador)

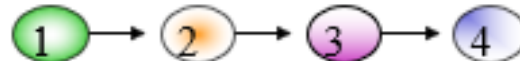
```
for (inicialización; condición; expresión)  
{  
    bloque  
}
```



inicialización, de la variable de control
condición, es la que determina si
seguir o parar la repetición
expresión, es aquella en la que el
valor de la variable de
control se actualiza

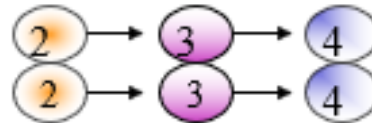


Orden de ejecución:



(Primera vez)

Orden de ejecución:



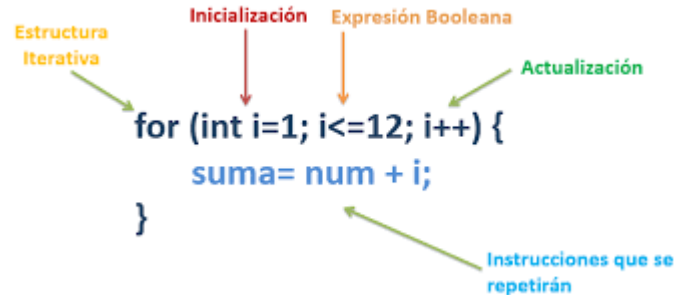
(Segunda vez)

Estructuras de Repetición. Bucle For (controlado por contador)



Realiza un programa que ejecute una lista de valores multiples de 5

```
public class Sentencia_for
{
    public static void main(String [] args)
    {
        System.out.println(" Genera una lista de valores multiplos de 5");
        for (int i = 0; i <= 5; i++)
        {
            System.out.print( " " +i*5);
        }
    }
}
```





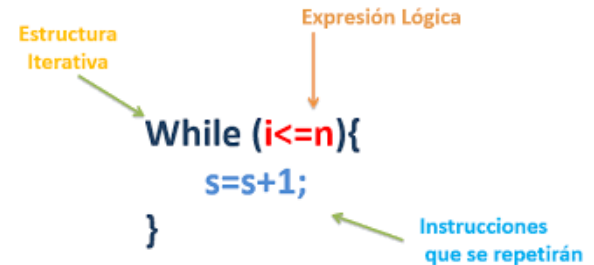
Estructuras de Repetición. Bucles While/ Do While (controlado por sucesos)

No se sabe a priori el número de veces que se va a repetir el bucle.

Las estructuras de control correctas son:

- **while**: Si el cuerpo del bucle va a realizarse 0 o más veces.
- **do while**: Si el cuerpo del bucle va a realizarse una o más veces.

Esta estructura tiene detractores como ya se comentó.

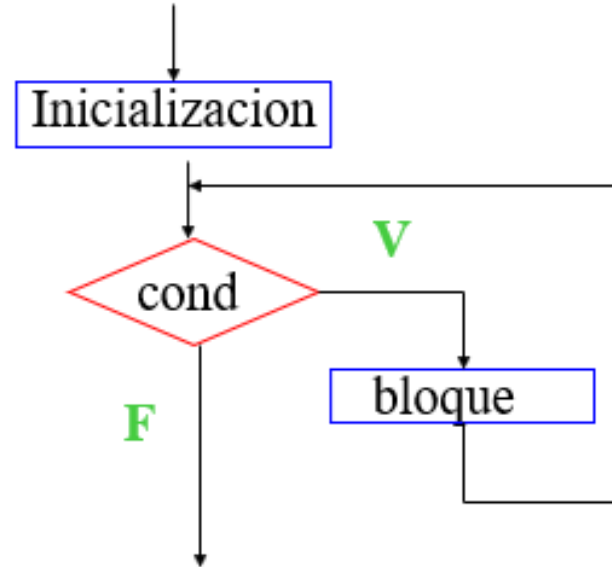


Estructuras de Repetición. Bucle While



Inicialización de variable de control;

```
while (condición) {  
    bloque  
    // actualizar variable  
    de control  
}
```



Variable de control, es la variable que sirve para llevar el control de las repeticiones.

Estructuras de Repetición. Bucle While



Realiza un programa que muestre desde el 20 hasta el cero los números pares.

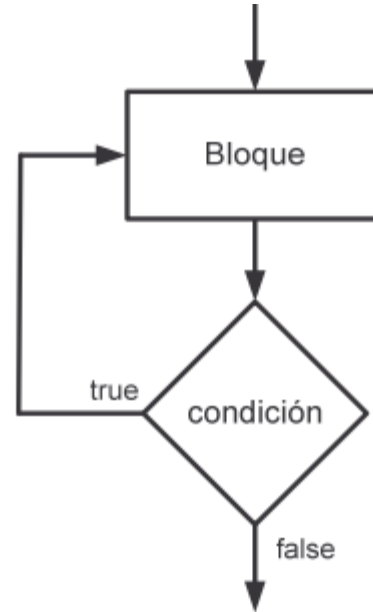
```
public class Sentencia_While
{
    public static void main(String [] args)
    {
        int i = 20;
        System.out.println( " --- Lista de números pares en orden decreciente---");
        while (i!= 0)
        {
            System.out.println( " i = " +i);
            i = i - 2;
        }
    }
}
```



Estructuras de Repetición. Bucle Do While



Estructura similar al While, que realiza la comprobación de la condición, después de ejecutar el cuerpo del bucle.



```
do {  
    bloque  
} while (condición);
```

Estructuras de Repetición. Bucle Do While



```
public static void main(String args[])

{
    int x = 21;

    do {

        System.out.println("Valor de x : " + x);

        x++;

    }

    while (x < 20);

}
```



Resumen. Estructuras de Repetición



- `while (expresión_booleana)`
 `instrucción`
- `do`
 `instrucción`
 `while (expresión_booleana)`
- `for (inicialización;condición;incremento)`
 `instrucción`



¿Cómo diseñar un bucle?



Diseño del flujo de control del bucle:

- ¿Cuál es la condición de continuidad del bucle? Respondiendo a esta pregunta sabremos si es por contador o por suceso.

Diseño del proceso interior del bucle:

- ¿Cuál es el proceso que se va a repetir?
- ¿Cómo debe inicializarse y actualizarse el proceso que se repite?