


4.2 Estructuras de repetición.

Se pueden ejecutar repetidamente una secuencia de sentencias creando un ciclo.

Sentencia while

Formato del while	Ejemplo	Funcionamiento
<code>while (condición)</code> <code> sentencia (s);</code>	<pre>char ch; // imprime el alfabeto utilizando ciclo ch = 'a'; while (ch <= 'z') { System.out.println (ch); ch ++; }</pre> <p>La variable ch se inicializa con 'a' minúscula. Cada vez que entre al ciclo o lo ejecute el ciclo, la variable ch se visualiza y se incrementa en 1. Este proceso continúa hasta que la variable ch sea menor o igual a z (minúscula).</p>	Se utiliza para repetir un bloque de instrucciones, siempre que una condición en particular sea verdadera.
	<pre>int c = 0; int suma = 0; while (c < 10) { c+=2; suma +=c; } System.out.println ("La sumatoria es: " + suma);</pre>	Equivale $c=c+2$ $\text{suma}=\text{suma}+c$

Formato del while	Ejemplo	Funcionamiento
<code>while (condición)</code> <code> sentencia (s);</code>	<pre>char resp = 's'; while (resp == 's') {  System.out.print ("/nEntre s para continuar y cualquier tecla para finalizar"); resp= (char) br.read (); br.skip(1); }</pre> <p>La variable resp se inicializa en s. Cada vez que se entra en el ciclo, antes de subir a verificar la condición la variable resp se lee. Este proceso continúa hasta que la variable resp sea diferente de s.</p>	Se utiliza para repetir un bloque de instrucciones, siempre que una condición en particular sea verdadera.

Sentencia do while

Formato del do while	Ejemplo	Funcionamiento
<pre>do { sentencia(s); } while (condición);</pre>	<pre>char ch; // imprime el alfabeto ch = 'a'; do { System.out.println (ch); ch ++; } while (ch <= 'z');</pre>	<p>El ciclo do while verifica su condición en la parte inferior del ciclo. Esto significa que un ciclo do while siempre se ejecuta por lo menos una vez.</p>
	<pre>int c = 0; int suma = 0; do { c+=2; suma +=c; } while (c < 10); System.out.println ("La sumatoria es: " + suma);</pre>	

Sentencia for

Formato del for	Ejemplo	Funcionamiento
<pre>for (inicialización;condición;incremento) sentencia(s);</pre>	<p>En forma positiva</p> <pre>int x; for (x = 1; x < 10; x++) System.out.println (x);</pre> <p>En forma negativa</p> <pre>int x; for (x = 100; x > -1000; x -=5) System.out.println (x);</pre>	<p>Inicialización establece el valor inicial de la variable de control del ciclo, la cual actúa como el contador que controla el ciclo.</p> <p>Condición es una expresión lógica que se evalúa y si el resultado es cierto el ciclo se repetirá. De lo contrario finaliza y continua con la siguiente instrucción después del for.</p> <p>Iteración define la cantidad en la que cambiará la variable de control del ciclo cada vez que el ciclo se repite.</p>

Ciclos anidados

Los ciclos anidados se utilizan para resolver una amplia variedad de problemas de programación y son una parte esencial de ésta.

Es posible insertar un ciclo dentro de otro, si todas las instrucciones del ciclo interno están contenidas en el ciclo externo.

El siguiente diagrama ilustra este concepto:

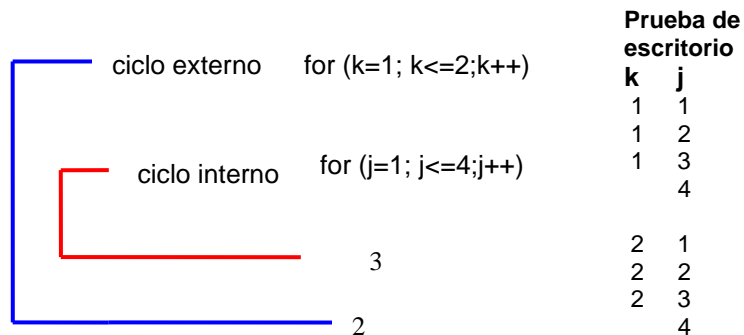
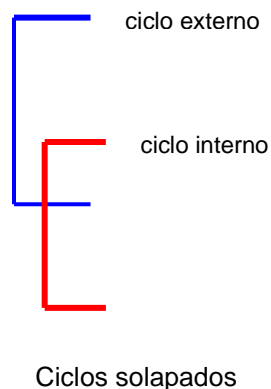


Figura 4.5 Estructura de repetición anidada

En el esquema podemos observar que j varía más rápido que k ; es decir, que el ciclo interno varía más rápido, que el ciclo externo.

Se pueden anidar tantas estructuras como sean necesarias, pero debemos recordar que se debe incluir un ciclo dentro del otro y no solaparlos. Por ejemplo, veamos la estructura a continuación:



Podemos combinar las diferentes estructuras: Para y Mientras, Mientras y Hasta que, Para y Para, etc.

Ejemplo:

```
for (int i = 2; i <= 100; i++)
{   System.out.println ("Factores de " + i + ":");
    for (int j = 2; j < i; j++)
    {   if ((i%j)==0)
        System.out.println(j + " ");
        System.out.println();
    }
}
```