EL CICLO FOR

```
for( inicializacion;  condicion; incremento)
    código
```

El ciclo **for** necesita una o varias **variables de control**. Veamos las diferentes partes del ciclo **for**. En primer lugar, encontramos la **inicialización**. En esta sección le damos a la variable de control su valor inicial. El valor inicial se lleva a cabo por medio de una **asignación** normal, tal y como las que hemos trabajado. La segunda sección lleva una **condición** en forma de una expresión relacional. Ésta nos sirve para controlar cuándo termina el ciclo y generalmente aquí indicamos la cantidad de vueltas que da el ciclo. Luego tenemos el **código**. En esta sección colocamos la parte del código que deseamos que se repita. Esta sección puede ser una sentencia o un bloque de código. Por último, se ejecuta la sección del **incremento**. Aquí indicamos cómo se modificará el valor de la variable de control para cada vuelta del ciclo.

El valor de inicio

Ahora ya podemos empezar a experimentar con el ciclo. Lo primero que haremos será colocar el valor de inicio del ciclo y ver cómo se modifica la ejecución del programa. Esto es útil ya que no siempre es necesario empezar a contar a partir de **1**, a veces necesitamos empezar a contar a partir de otros números.

Supongamos que ahora tenemos que contar del **3** al **10**. Para lograr esto, simplemente modificamos el valor de inicio en la sección de inicialización del ciclo:

```
for  (n = 3; n <= 10;  n = n + 1)
    Console.WriteLine("{0}", n);
```

El contador y el acumulador

Existen dos categorías de variables dependiendo de cómo guardan la información, en primer lugar, tenemos el **contador**. El contador es una variable que será incrementada o disminuirá su valor de uno en uno. Por su parte, el **acumulador** es una variable que puede incrementar o disminuir su valor en cualquier número.

Incrementos y decrementos

Cuando trabajamos el C# con el ciclo **for** es muy común que tengamos que incrementar o disminuir siempre de uno en uno. Para facilitarnos esto, tenemos operadores nuevos que son el **operador de incremento** y el **operador de decremento**.

OPERADOR	SIGNO
Incremento	++
Decremento	—

***Tabla 1.** Este tipo de operadores nos permiten cambiar el valor de la variable en 1.*

Estos operadores pueden ser usados como **sufijos** o **prefijos** en la variable. En el caso de los sufijos lo escribimos como **variable++** y para el prefijo como **++variable**. El valor contenido en la variable en ambos casos se incrementará en uno. Lo que cambia es cómo se evalúa la expresión que contienen los operadores.

Cuando tenemos el caso del sufijo, se evalúa la expresión con el valor actual de la variable y luego se incrementa a la variable. En el caso del prefijo, se incrementa primero el valor de la variable y posteriormente se evalúa la expresión.

EL CICLO DO WHILE

La instrucción while ejecuta una instrucción o un bloque de instrucciones mientras que una expresión booleana especificada se evalúa como verdadera. Debido a que esa expresión se evalúa antes de cada ejecución del ciclo, un ciclo while se ejecuta cero o más veces. Esto difiere del bucle do, que se ejecuta una o más veces.

En cualquier punto dentro del bloque de instrucción while, puede salir del ciclo utilizando la instrucción break.

Puede pasar directamente a la evaluación de la expresión while utilizando la instrucción continue. Si la expresión se evalúa como verdadera, la ejecución continúa en la primera instrucción del bucle. De lo contrario, la ejecución continúa en la primera instrucción después del ciclo.

```
do  {
      Código
}(condición);
```

EL CICLO WHILE

El ciclo **while** también puede ser utilizado cuando tenemos algo que se debe repetir pero no conocemos el número de repeticiones previamente. La repetición del ciclo tiene que ver con el cumplimiento de una condición. A diferencia del ciclo **do while**, este ciclo puede no ejecutarse ni siquiera una vez.

```
while(condicion)  {
            Código
}
```