



CONTROL DEL FLUJO DE EJECUCIÓN

En principio, las sentencias de un programa en C# se ejecutan *secuencialmente*, esto es, cada una a continuación de la anterior empezando por la primera y acabando por la última. El lenguaje C# dispone de varias sentencias para modificar este flujo secuencial de la ejecución.

Las más utilizadas se agrupan en dos familias: las **bifurcaciones**, que permiten elegir entre dos o más opciones según ciertas condiciones, y los **bucles**, que permiten ejecutar repetidamente un conjunto de instrucciones tantas veces como se desee, cambiando o actualizando ciertos valores.

Bifurcaciones o decisiones



Las sentencias de selección permiten controlar el flujo del programa, seleccionando distintas sentencias en función de diferentes valores.

Sintaxis:

```
if (condición) sentencias;  
  [else if (condición) sentencias;]  
  [else if (condición) sentencias;]  
  .....  
  [else sentencias;]
```

Ejemplo:

```
if (letra == 'a')  
    Console.WriteLine("Letra {0}",letra);  
else if (letra == 'e')  
    Console.WriteLine("Letra {0}",letra);  
else if (letra == 'i')  
    Console.WriteLine("Letra {0}",letra);  
else if (letra == 'o')  
    Console.WriteLine("Letra {0}",letra);  
else if (letra == 'u')  
    Console.WriteLine("Letra {0}",letra);  
else  
    printf ("No hay vocales");
```



Switch

Cuando se usa la sentencia "switch" el control se transfiere al punto etiquetado con el "case" cuya expresión constante coincida con el valor de la variable del "switch". A partir de ese punto todas las sentencias serán ejecutadas hasta el final del "switch", es decir hasta llegar al "}". Esto es así porque las etiquetas sólo marcan los puntos de entrada después de una ruptura de la secuencia de ejecución, pero no marcan las salidas. Esta característica nos permite ejecutar las mismas sentencias para varias etiquetas distintas, y se puede eludir usando la sentencia de ruptura "break" al final de las sentencias incluidas en cada "case". Si no se satisface ningún "case", el control parará a la siguiente sentencia después de la etiqueta "default". Esta etiqueta es opcional y si no aparece se abandonará el "switch".

Sintaxis:

```
switch (variable)
{
    case constante1 :
        sentencias;
        break;
    case constante2 :
        sentencias;
        break;
    .....
    [default:
        sentencias;
        break;]
}
```

(variable se va comparando con cada constante)

Ejemplo:

```
switch(letra)
{
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
        EsVocal = true;
        break;
    default:
        EsVocal = false;
        Break;
}
```



Bucles

While

El bucle **while** se ejecuta mientras una condición es cierta. Cuando esta condición se torna falsa, el bucle termina su operación.

Sintaxis:

```
inicialización
while (condición)
{
    sentencias;
}
```

Ejemplo:

```
int contador;
contador = 0;
while (contador <5)
{
    printf ("%d\n", contador);
    contador = contador + 1;
}
```

Salida:

0
1
2
3
4

Do-while

La sentencia es ejecutada repetidamente mientras la condición resulte falsa. Si no se especifica condición se asume que es "true", y el bucle se ejecutará indefinidamente. A diferencia del bucle "while", la evaluación se realiza después de ejecutar la sentencia, de modo que se ejecutará al menos una vez.

Sintaxis:

```
inicialización
do
{
    sentencias
} while (condición) ;
```

Ejemplo:

```
int contador;
contador = 0;
do
{
    printf ("%d\n", contador);
    contador = contador + 1;
} while (contador < 5);
```

Salida:

0
1
2
3
4
5



For

La sentencia es ejecutada repetidamente hasta que la evaluación de la condición resulte falsa. Esta expresión se compone realmente de tres campos cada uno separado por un punto y coma. El primer campo contiene la expresión "**índice = 0**" y se le llama campo de inicialización. El segundo campo es la prueba que se hace al principio de cada ciclo del bucle y puede ser cualquier expresión que pueda evaluarse a verdadero ó falso. La expresión del tercer campo se ejecuta en cada ciclo del bucle pero solo hasta que se hayan ejecutado todas las instrucciones contenidas dentro del cuerpo principal del bucle.

Sintaxis:

```
for (inicialización; condición; incremento)  
{  
    sentencias;  
}
```

Ejemplo:

```
int contador;  
for (contador = 0; contador <= 5; contador++) {  
    Console.WriteLine("{0}", contador);  
}
```

Salida:

```
0  
1  
2  
3  
4  
5
```

Un bucle **while** es útil cuando se desconoce cuantas veces será ejecutado un bucle, en tanto que la instrucción **for** se usa generalmente en aquellos casos en donde debe existir un número fijo de interacciones, además, el bucle **for** es conveniente porque contiene toda la información del control del bucle en un solo lugar, dentro de un paréntesis.

Interrupciones

break

La instrucción **break** lleva al programa a salirse del bucle que se estaba ejecutando para continuar con los enunciados inmediatos al bucle, terminando éste en forma efectiva. Se trata de una instrucción muy útil cuando se desea salir del bucle dependiendo de los resultados que se obtengan dentro del bucle.

continue

La instrucción **continue** no finaliza el bucle pero suspende el presente ciclo. El enunciado **continue** siempre brinca al final del bucle, justo antes de la llave que indica el fin del bucle.



Ejercicios de la Práctica

Objetivo. El alumno comprenderá de manera práctica los conceptos relacionados con estructuras de control.

Llama a tu solución Practica4_estructuras_control. Para cada uno de los siguientes ejercicios realiza un proyecto diferente.

Bifurcación o desición

1. Escriba un programa que dado el sueldo del trabajador, aplique un 15% si su sueldo es inferior a \$1000. Imprima en este caso el nuevo sueldo del trabajador.
2. Escriba un programa que dado el sueldo del trabajador, aplique un 15% si su sueldo es inferior a \$1000 y 12% en caso contrario. Imprima el nuevo sueldo del trabajador.
3. Escriba un programa que dado como datos la categoría y el sueldo de un trabajador, calcule el aumento correspondiente teniendo en cuenta la siguiente tabla. Imprima la categoría del trabajador y su nuevo sueldo.

Tabla de categorías y sueldos	
CATEGORÍA	AUMENTO
A	15%
B	10%
C	8%
D	7%

Bucles (Repetición)

4. Escribir un programa que pida una contraseña y permita tres intentos. Si el usuario da la contraseña correcta despliegue "Bienvenido al sistema!". En caso contrario despliegue "Datos incorrectos.". Y en la última iteración "Datos incorrectos. El sistema ha quedado bloqueado!"
5. Escribir un programa que imprima por pantalla los códigos ASCII correspondientes a los números 32 al 127.
6. Escriba un programa que obtenga la suma e imprima los términos de la siguiente serie:
2, 5, 7, 10, 12, 15, 17, ..., 1800