educación IT
CODE YOUR FUTURE

Introducción a C# .NET

Módulo 3

Instrucciones Break, Continue, Return, Goto

Instrucción break

Ya vimos que la **instrucción break** sólo se puede incluir dentro de bloques de instrucciones asociados a instrucciones iterativas o instrucciones **switch**, e indica que se desea abortar la ejecución de las mismas y seguir ejecutando a partir de la instrucción siguiente a ellas.

Se usa así:

```
break;
```

Veamos un ejemplo en el próximo slide.

```
static void Break_Ejemplo()
{
      for (int numero = 2; numero <= 30; numero += 2)
            {
                  Console.WriteLine(numero);

                  if (numero == 20)
                  {
                        Console.WriteLine("Llegó a 20 y salió con Break.");
                        break;
                  }
       }
       Console.WriteLine("Aquí salió con el break");
}
```

Cuando esta sentencia se usa dentro de un bloque **`try`** con cláusula **`finally`** (que tiene como objetivo atrapar errores en tiempo de ejecución, tema a ver en el siguiente curso de C#), antes de abortarse la ejecución de la instrucción iterativa o del **`switch`** que la contiene y seguirse ejecutando por la instrucción que le siga, se ejecutarán las instrucciones de la cláusula **`finally`** del **`try`**. Esto se hace para asegurar que el bloque **`finally`** se ejecute aún en caso de salto.

Además, si dentro de una cláusula **`finally`** incluida en un **`switch`** o de una instrucción iterativa se usa **`break`**, no se permite que como resultado del **`break`** se salga del **`finally`**.

Instrucción continue

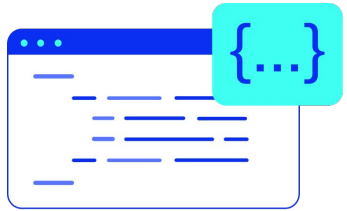
Ya se ha comentado que la instrucción llamada `continue` **sólo puede usarse dentro del bloque de instrucciones de una instrucción iterativa.** Indica que se desea pasar a reevaluar directamente la condición de la iteración sin ejecutar el resto de instrucciones que contiene.

La evaluación de la condición se haría de la forma habitual: si es cierta se repite el ciclo y si es falsa se continúa ejecutando por la instrucción que le sigue.

Su sintaxis de uso es así de sencilla:

```
continue;
```

Veamos un ejemplo en el próximo slide.



```
static void Continue_Ejemplo()
{
	for (int numero = 2; numero <= 30; numero += 2)
	{
		Console.WriteLine(numero);

		if (numero == 20)
		{
			Console.WriteLine("con Continue vuelve a ingresar al ciclo. No sale nunca");
			numero = 8;
			continue;
		}
	}
}
```

En cuanto a sus usos dentro de sentencias **`try`**, tiene las mismas restricciones que **`break`**: antes de salir de un **`try`** se ejecutará siempre su bloque **`finally`** y no es posible salir de un **`finally`** incluido dentro de una instrucción iterativa como consecuencia de un **`continue`**.

Instrucción return

Esta instrucción se usa para indicar cuál es el objeto que ha de devolver una función. Su sintaxis es la siguiente:

```
return <objetoRetorno>;
```

La ejecución de esta instrucción provoca que se aborte la ejecución de la función en la que aparece, y que se devuelva el `<objetoRetorno>` al código que llamó a la función. Como es lógico, este objeto ha de ser del tipo de retorno de la función en que aparece el **`return`** o de alguno compatible con él, por lo que esta instrucción solo podrá incluirse en funciones cuyo tipo de retorno no sea **`void`**, o en los bloques **`get`** de las propiedades o indizadores. De hecho, es obligatorio que toda función con tipo de retorno termine con un **`return`**.

Las funciones que devuelvan **`void`** pueden tener un **`return`** con una sintaxis especial en la que no se indica ningún valor a devolver sino que simplemente se usa **`return`** para indicar que se desea terminar su ejecución:

```
return;
```

Ejemplos:

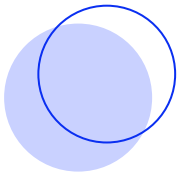
```
static int Return_Ejemplo_Funcion()
      {
      int numero1 = 10, numero2 = 20;
      retur nnumero1 + numero2;
      }
```

```
static void Return_Ejercicio4()
      {
      for (int numero = 2; numero <= 30; numero += 2)
            {
            Console.WriteLine(numero);

            if (numero == 20)
                {
                   Console.WriteLine("Sale con Return.....");
                   return;
                }
            }
       }
```

Nuevamente, como con el resto de las instrucciones de salto hasta ahora vistas, si se incluyese un **`return`** dentro de un bloque **`try`** con cláusula **`finally`**, antes de devolverse el objeto especificado se ejecutarían las instrucciones de la cláusula **`finally`**.

Si hubiesen varios bloques **`finally`** anidados, las instrucciones de cada uno se ejecutarían de manera ordenada (o sea, del más interno al más externo). Ahora bien, lo que no es posible es incluir un **`return`** dentro de una cláusula **`finally`**.



Instrucción goto

La **instrucción goto** permite pasar a ejecutar el código a partir de una instrucción cuya etiqueta se indica en el **goto**. La sintaxis de uso de esta instrucción es:

```
goto <etiqueta>;
```

Como en la mayoría de los lenguajes, **goto** es una **instrucción nada recomendable** cuyo uso dificulta innecesariamente la legibilidad del código y suele ser fácil de simular usando instrucciones iterativas y selectivas con las condiciones apropiadas.

Sin embargo, en C# se incluye porque puede ser eficiente usarla si se anidan muchas instrucciones y para reducir sus efectos negativos se le han impuesto unas restricciones. Vamos a ver estas restricciones en el próximo slide.

¡Esta instrucción mal utilizada puede provocar complejos errores de lógica en la programación!

- Sólo se pueden etiquetar instrucciones, y no directivas de preprocesado, directivas **`using`** o definiciones de miembros, tipos o espacios de nombres.
- La etiqueta indicada no pueda pertenecer a un bloque de instrucciones anidado dentro del bloque desde el que se usa el **`goto`** ni que etiquete a instrucciones de otro método diferente a aquél en el cual se encuentra el **`goto`** que la referencia.
- Para etiquetar una instrucción de modo que pueda ser destino de un salto con **`goto`** basta precederla del nombre con el que se la quiera etiquetar seguido de dos puntos (**`:`**).

En el ejemplo de la siguiente slide veremos cómo usar **`goto`** y definir una etiqueta.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GOTO
{
class SalirGOTO
{
      static void Main(string[] args)
      {
            for (int numero = 1; numero <= 5; numero++)
            {
                  if (numero <= 3)
                  {
                        Console.WriteLine(numero);
                  }
```

```
                else
                {
                goto salida;
                }
            }

            salida: Console.WriteLine("Llegó a 3 e interrumpió la ejecución del for");
            Console.ReadKey();
        }
    }
}
```

El programa del ejemplo anterior muestra por pantalla los números del 1 al 5, pero una vez mostrado el 3 se aborta su ejecución.

Véase además que el ejemplo pone de manifiesto una de las utilidades de la instrucción nula, ya que si no se hubiese escrito tras la etiqueta **salida,** el programa no compilaría en tanto que toda etiqueta ha de preceder a alguna instrucción (aunque sea la instrucción nula).

Nótese que al fin y al cabo los usos de **`goto`** dentro de instrucciones **`switch`** que se vieron al estudiar dicha instrucción no son más que variantes del uso general de **`goto`**, ya que **`default:`** no es más que una etiqueta y **`case <valor>:`** puede verse como una etiqueta un tanto especial cuyo nombre es **`case`** seguido de espacios en blanco y un valor. En ambos casos, la etiqueta indicada ha de pertenecer al mismo **`switch`** que el **`goto`** usado.

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!