educación IT
CODE YOUR FUTURE

Introducción a C# .NET

Módulo 1

Operadores

Operadores

Un **operador** en C# es un símbolo formado por uno o más caracteres que permite realizar una determinada operación entre uno o más datos y produce un resultado.

A continuación, vamos a describir cuáles son los operadores incluidos en el lenguaje y clasificados según el tipo de operaciones que permiten realizar. Hay que tener en cuenta que **C# permite la redefinición del significado de la mayoría de los operadores,** según el tipo de dato sobre el que se apliquen, así que lo que **aquí se muestra se corresponde con los usos más comunes.**

Un operador **es una expresión que produce otro valor** (así como las funciones o construcciones que devuelven un valor).

Existen operadores de comparación, de negación y, de incremento y decremento.

Las operaciones matemáticas se comportan de igual manera en C#. Las operaciones *** (producto)** y **/ (división)** tienen precedencia sobre la **+ (suma)** y la **- (resta)** y se pueden usar paréntesis para expresiones más complejas.

Tipos de operadores

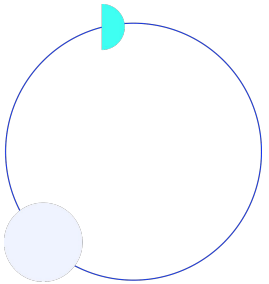
Operaciones aritméticas

Los operadores aritméticos incluidos en C# son los típicos de suma (+), resta (-), producto (*), división (/) y módulo (%).

También se incluyen operadores de “menos unario” (–) y “más unario” (+).

Operaciones relacionales

Se han incluido los tradicionales operadores de igualdad (==), desigualdad (!=), “mayor que” (>), “menor que” (<), “mayor o igual que” (>=) y “menor o igual que” (<=).



Operaciones lógicas

Se incluyen operadores que permiten realizar las operaciones lógicas típicas: "*and*" (**&&** y **&**), "*or*" (**||** y **|**), "***not***" (**!**) y "***xor***" (**^**).

Los operadores **&&** y **||** se diferencian de **&** y **|** en que los primeros realizan la *evaluación perezosa* y los segundos no. La evaluación perezosa consiste en que si el resultado de evaluar el primer operando permite deducir el resultado de la operación, entonces no se evalúa el segundo y se devuelve dicho resultado directamente.
La evaluación *no perezosa* consiste en evaluar siempre ambos operandos.

Es decir, si el primer operando de una operación **&&** es falso, se devuelve **`false`** directamente, sin evaluar el segundo; y si el primer operando de una **||** es verdadero, se devuelve **`true`** directamente, sin evaluar el otro.

Operaciones de asignación

Para realizar asignaciones se usa en C# el operador **=**, operador que, además de realizar la asignación que se le solicita, devuelve el valor asignado. Por ejemplo, la expresión **a = b** asigna a la variable **a** el valor de la variable **b** y devuelve dicho valor, mientras que la expresión **c = a = b** asigna a las variables **c** y **a** el valor de **b** (el operador **=** es asociativo por la derecha).

También se han incluido **operadores de asignación compuestos** que permiten ahorrar tecleo a la hora de realizar asignaciones tan comunes como las que vemos a la derecha.

```
temperatura = temperatura + 15; // Sin usar asignación compuesta

temperatura += 15; // Usando asignación compuesta
```

Las dos líneas anteriores son equivalentes, pues el operador compuesto **+=** asigna a su primer operando el valor que tenía más el de su segundo operando (o sea, le suma el segundo operando). Como se ve, **permite compactar el código.**

Aparte del operador de asignación compuesto **+=**, también se ofrecen operadores de asignación compuestos para la mayoría de los operadores binarios ya vistos. Estos son: **+=**, **-=**, ***=**, **/=**, **%=**, **&=**, **|=**, **^=**, **<<=** y **>>=**. Nótese que no hay versiones compuestas para los operadores binarios **&&** y **||**.

Otros dos operadores de asignación incluidos son los de incremento (**++**) y decremento (**--**). Estos operadores permiten, respectivamente, aumentar y disminuir en una unidad el valor de la variable sobre el que se aplican. Así, estas líneas de código son equivalentes:

```
static void Main(string[] args)
        {
            double temperatura = 0;
            temperatura = temperatura + 1; // Sin asignación compuesta ni incremento
            temperatura += 1; // Usando asignación compuesta
            temperatura++; // Usando incremento
        }
```

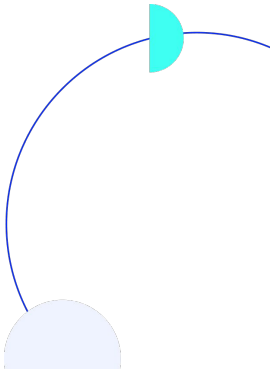
Si el operador **++** se coloca tras el nombre de la variable (como en el ejemplo del slide anterior) devuelve el valor de la variable antes de incrementarla, mientras que si se coloca antes, devuelve el valor de ésta tras incrementarla; y lo mismo ocurre con el operador **--**.

Por ejemplo:

```
c = b++; // Se asigna a c el valor de b y luego se incrementa b

c = ++b; // Se incrementa el valor de b y luego se asigna a c
```

La ventaja de usar los operadores **++** y **--** es que en muchas máquinas son más eficientes que otras formas de realizar sumas o restas de una unidad, pues el compilador puede traducirlos en una única instrucción en código máquina.



Operaciones con cadenas

Para realizar operaciones de concatenación de cadenas se puede usar el mismo operador que para realizar sumas, ya que en C# se ha redefinido su significado para que cuando se aplique entre operandos que sean cadenas o que sean una cadena y un carácter lo que haga sea concatenarlos.

Por ejemplo, **`“Hola ”+ “mundo”`** devuelve **`“Hola mundo”`**, y **`“Hola mund”`** + **`‘o’`** también.

Operador condicional ternario

Es el único operador incluido en C# que toma 3 operandos, y se usa así:

`<condición>?<expresión1>:<expresión2>`

Esto es, se evalúa `<condición>`. Si es cierta, se devuelve el resultado de evaluar `<expresión1>`, y si es falsa se devuelve el resultado de evaluar `<expresión2>`.

Un ejemplo de su uso es:

```
b = (a>0)? a : 0;
// Suponemos a y b de tipos enteros
```

En el ejemplo, si el valor de la variable **`a`** es mayor a **`0`** se asignará a **`b`** el valor de **`a`**, en caso contrario el valor que se le asignará será **`0`**. Hay que tener en cuenta que este operador es asociativo por la derecha, por lo que una expresión como **`a?b:c?d:e`** es equivalente a **`a?b:(c?d:e)`**

No hay que confundir este operador con la instrucción condicional **`if`** que se tratará en el *Tema Instrucciones*, pues aunque su utilidad es similar al de ésta ,**`?`** devuelve un valor e **`if`** no.

> **Fuentes consultadas:**
> operador condicional ternario | Microsoft Learn

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!