Variables y operadores

Índice

1 - Qué son las variables

Lo primero que tenemos que tener en cuenta es qué son las variables. Son nombres que apuntan a un valor que el compilador toma de forma dinámica. Cada variable necesita tener un tipo el cual determina su tamaño y la forma en la que es almacenado en memoria

1.1 - Tipos de variable

Disponemos de multitud de tipos de variables, en el video solo enumero las principales, ya que los videos estan enfocados al mundo real, y en el mundo real tan solo utilizamos unas pocas aquí enumerare alguna más.

Tipo	Descripción
int, long, sbyte, byte, short, ushort, uint, ulong, char	Números Enteros
Float, doble	Números de coma flotante
Decimal	Números decimales
Bool	verdadero o falso
null	tipos nulos

Además de estos tipos C# permite tener enum y referencias tipo de variable y que es muy común utilizar variables de tipo de referencia como pueden ser las clases, interfaces, o delegados en el mundo real, ya que al trabajar se intenta hacer lo más similar al mundo real posible, lo cual nos lo facilita mucho. Los tipos de referencia los veremos más adelante.

1.2 - Declaración de variables

Declarar una variable es muy sencillo, únicamente tenemos que indicar el tipo de variable y su nombre.

```
int ejemploVariable;
```

Pero en este ejemplo la variable no tienen ningun valor, si intentaramos consultarla nos diria que es `null` por lo que es como si no tuviésemos nada y si fueramos a operar con ella el compilador nos

daria un error. Para evitar esto, debemos asignarle un valor.

para ello tenemos dos formas, asignarle el valor una vez está creada, o asignarle el valor en la propia creación de la variable.

```
int ejemploVariable;
ejemploVariable = 5;

//Segunda forma

int ejemploVariable = 5;
```

Ambos ejemplos nos crearan una variable `ejemploVariable` con un valor numérico de 5. Personalmente prefiero la segunda forma. Si utilizamos la primera, el compilador nos dirá que qué tal si utilizamos la segunda, que es mejor para que la lean los humanos.

Podemos definir mas de una variable si las separamos porp coma (`,`) y hay que poner punto y coma (`;`) al terminar cada sentencia.

```
int ejemploVariable, ejemploVariable2;
```

1.3 - Tipo implícito de variables

Desde la entrada de C# 3 podemos utilizar el tipo implícito para crear una variable.

Esto que quiere decir, que no tenemos porqué indicar que tipo de variable es cuando la declaramos ya que cogerá el tipo de variable de la inicialización.

```
var ejemploVariable = 5;
```

Como vemos en el ejemplo utilizamos la palabra reservada `var` la cual se le asigna el tipo del valor que asignamos a la variable.

Esto se puede hacer tanto con tipos primitivos como con tipos de referencia creados por nosotros mismos. Personalmente recomiendo utilizar el tipo de variable en vez de var ya que a la hora de leer el codigo, o hacer un "code review" (**curso git y github**) es mucho mas sencillo de entender.

1.4 - Declaración de constantes

Algunas veces solo tenemos que asignar el valor a una variable una vez ya que este no va a cambiar en toda la ejecución del programa. Para ello utilizaremos lo que se denomina como constante. Ello nos proporciona ciertas ventajas sobre las variables convencionales.

- Nos aseguramos que ese código no cambia; en un proyecto pequeño nos puede dar mas igual, pero en uno grande puede llegar a ayduar bastante.
- Debido al punto anteior el mantenimiento, o si falla es mas sencillo, ya que no tenemos que ir buscando lugares en los que el valor cambia.
- El compilador es mas eficiente; Ya que no tiene que estar preocupandose de si el valor cambia o no, por lo que puede optimizar el código a sabiendas de que el valor siempre es el msimo.

Un ejemplo de consante podría ser el siguiente, recordad que es un valor que no cambia nunca.

```
public const int numeroMeses = 12;
```

2 - Qué son los operadores

Ahora que hemos visto que son las variables pasaremos a otro punto de vital importancia, los operadores. Los cuales nos permiten realizar operaciones tanto matemáticas con operadores como lógicas.

C# tiene una gran variedad de operadores como vamos a ver a continuación.

2.1 - Operadores aritméticos.

Son aquellos operadores que nos permiten realizar acciones, normalmente matemáticas:

Operador	Descripción	Ejemplo
`+`	Suma o concatena dos operadores.	`A + B = 10`
`-`	Resta el segundo operador al primero.	`A - B = 5`
`*`	Multiplica ambos operadores.	`A * B = 25`
`/`	Divide ambos operaores.	`A / B = 1`
`%`	Operador módulo, nos devuelve el resto de la operación.	`A % B = 0`
`++`	Incrementa el valor en una unidad.	`A++` devuelve 6
`--`	Decrementa el valor en una unidad.	`A--` devuelve 4

Esto es un pequeño ejemplo, si cambiamos el operador con cualquiera de los de arriba veremos como nos da la salida correspondiente.

```
int operadorA = 5;
int operadorB = 5;
int resultado = operadorA + operadorB;
//Resultado nos devolvera 10 en este caso.
```

2.2 - Operadores relacionales

Los operadores relacionales son aquellos que nos permiten realizar una comparación; Supongamos que los numeros a comparar son `A = 5, B =2` :

Operador	Descripción	Ejemplo
`==`	Comprueba si amos operadores son iguales.	`(A == B )` devuelve falso
`>`	Comprueba que el valor de la izquierda es mayor que el de la derecha. Si lo es, el valor es verdadero.	`( A > B )` devuelve verdadero

`<`	Comprueva que el valor de la izquierda es menor que el de la derecha. Si lo es, el valor es verdadero.	`( A < B )` devuelve falso
`>=`	Comprueba que el valor de la izquierda es mayor o igual que el de la derecha. Si lo es, el valor es verdadero.	`( A > B )` devuelve verdadero
`<=`	Comprueva que el valor de la izquierda es menor o igual que el de la derecha. Si lo es, el valor es verdadero	`( A < B )` devuelve falso
`!=`	Comprueba que ambos valores son iguales o no. Si no son iguales devuelve verdadero	`( A != B )` devuelve verdadero

2.3 - Operadores lógicos

Son las expresiones lógicas y comparan boleanos; Para el ejemplo `A = true, B = false` :

Operador	Descripción	Ejemplo
`&&`	Operador lógico AND, si ambos booleanos son verdadero el resultado será verdadero.	`( A && B )` devuelve falso
`||`	Operador lógico OR, si alguno de los dos es verdadero el resultado sera verdadero.	`( A || B )` devuelve verdadero
`!`	Operador lógico NOT, usado para negar una operación logica,, si una condición es verdadera entonces el operador NOT la convertira en falsa.	`!(A && B )` devuelve verdadero

2.4 - Operadores de asignación

Son los operadores que utilizamos para asignar valor a una variable.

Operador	Descripción	Ejemplo
`=`	Operador de asignación. Asigna un valor desde el lado derecho a la parte izquierda de la operación.	`C = A + B` ; Asigna el valor de A + B dentro de C
+=	Operador Aritmético de añadir y operador de asignación; Añade el valor de la izquierda al de la drecha y lo suma. Esta operación se puede hacer con cualquier operador aritmético.	`B += A` es equivalente a B = B + A