# Tipos de datos

Las variables representan las ubicaciones de almacenamiento. Cada variable tiene un tipo que determina qué valores se pueden almacenar en la variable. C# es un lenguaje seguro para tipos, y el compilador de C# garantiza que los valores almacenados en las variables sean siempre del tipo adecuado. El valor de una variable se puede cambiar mediante asignación o mediante el uso de los operadores ++ y --.

La siguiente tabla muestra las palabras reservadas para los tipos predefinidos en C#, lo cuales son alias para los tipos de dato existentes en el namespace System.

|  |  |
| --- | --- |
| Tipo | Alias del tipo |
| sbyte | System.SByte |
| byte | System.Byte |
| short | System.Int16 |
| ushort | System.UInt16 |
| int | System.Int32 |
| uint | System.UInt32 |
| long | System.Int64 |
| ulong | System.UInt64 |
| char | System.Char |
| float | System.Single |
| double | System.Double |
| bool | System.Boolean |
| decimal | System.Decimal |
| string | System.String |
| object | System.Object |

## Tipos enteros

La siguiente tabla muestra los tamaños y los rangos de los tipos de dato enteros.

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Rango** | **Tamaño** |
| sbyte | -128 a 127 | Enteros de 8 bits con signo |
| byte | 0 a 255 | Enteros de 8 bits sin signo |
| short | -32768 a 32767 | Enteros de 16 bits con signo |
| ushort | 0 a 65535 | enteros de 16 bits sin signo |
| int | -2147483648 a 2147483647 | Enteros de 32 bits con signo |
| uint | 0 a 4294967295 | Enteros de 32 bits sin signo |
| long | -9223372036854775808 a 9223372036854775807 | Enteros de 64 bits con signo |
| ulong | 0 a 18446744073709551615 | Enteros de 64 bits sin signo |
| char | 0 a 65535 | Enteros de 16 bits sin signo |

**Con signo (signed)** significa que la variable puede almacenar números positivos y negativos, **sin signo** **(unsigned)** significa que la variable solo puede almacenar números positivos.

## Tipos de punto flotante

La siguiente tabla muestra la precisión y el rango aproximado para los tipos de punto flotante.

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Rango** | **Precisión** |
| float | 1.5 × 10⁻⁴⁵ to 3.4 × 10³⁸ | 7 dígitos |
| double | 5.0 × 10⁻³²⁴ to 1.7 × 10³⁰⁸ | 15-16 dígitos. |
| decimal | 7.9 × 10⁻²⁸ to 7.9 × 10²⁸ | 28 dígitos |

# Declaración de variables

Las variable son estructuras de datos que pueden variar durante la ejecución de un programa.

Declarar variables significa indicar el tipo de dato que almacenara las variables que participan en el programa.

Las variables se van a declarar primero indicando el tipo de dato, luego el nombre de la variable que se quiere definir, y opcionalmente puede indicar un valor inicial con el operador de igual, se le da el valor y al final se pone punto y coma.

Ejemplos

int numero = 10;

long numeroLargo = 90192910290219;

bool vivo = true;

char caracter = 'C';

string cadena = "Este es un texto";

float precio = 9.99F;

double total = 5.99;

decimal total2 = 9.99M;

Notas

Los tipos más utilizados para números son int (para números enteros) y double (para números de coma flotante).

### Long

Esto se usa cuando int no es lo suficientemente grande para almacenar el valor. Opcionalmente puede terminar el valor con L.

### Float

Debe terminar el valor con una F.

### Double

Opcionalmente puede terminar el valor con una d.

### Decimal

Debe terminar el valor con una M. Es utilizado para dinero (money).

### Bool

Solo puede tomar los valores true o false. Se utilizan principalmente para pruebas condicionales.

**Char**

Se utiliza para almacenar un solo carácter. El carácter debe estar entre comillas simples.

### String

Los valores de cadena deben estar entre comillas dobles.

# Operaciones básicas

Los operadores se utilizan para realizar operaciones en variables y valores.

En el siguiente ejemplo, usamos el operador + para sumar dos valores:

int suma = 8 + 20;

Aunque el operador + se usa a menudo para sumar dos valores, como en el ejemplo anterior, también se puede usar para sumar una variable y un valor, o una variable y otra variable:

int suma1 = 20 + 50; // 70 (20 + 50)

int suma2 = suma1 + 100; // 170 (70 + 100)

int suma3 = suma2 + suma2; // 340 (170 + 170)

## Operadores aritméticos

Los operadores aritméticos se utilizan para realizar operaciones matemáticas comunes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Nombre** | **Descripción** | **Ejemplo** |
| + | Suma | Suma dos valores. | x + y |
| - | Resta | Resta un valor de otro. | x - y |
| \* | Multiplicación | Multiplica dos valores. | x \* y |
| / | División | Divide un valor por otro. | x / y |
| % | Módulo | Devuelve el residuo de la división. | x % y |
| ++ | Incremento | Aumenta el valor de la variable en 1. | x++ |
| -- | Decremento | Disminuye el valor de la variable en 1. | x-- |

## Operadores de asignación

Los operadores de asignación se utilizan para asignar valores a las variables.

En el siguiente ejemplo, usamos el operador de asignación = para asignar el valor 10 a una variable llamada x.

int x = 5;

El operador de asignación de suma += agrega un valor a la misma variable.

int x = 5;

x += 10;

### Lista de todos los operadores de asignación

|  |  |  |
| --- | --- | --- |
| **Operador** | **Ejemplo** | **Equivale a** |
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| \*= | x \*= 3 | x = x \* 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| &= | x &= 3 | x = x & 3 |
| |= | x |= 3 | x = x | 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

## Operadores de comparación

Los operadores de comparación se utilizan para comparar dos valores:

|  |  |  |
| --- | --- | --- |
| **Operador** | **Nombre** | **Ejemplo** |
| == | Igual a | x == y |
| != | No es igual a | x != y |
| > | Mayor que | x > y |
| < | Menor que | x < y |
| >= | Mayor o igual que a | x >= y |
| <= | Menor o igual que a | x <= y |

## Operadores lógicos

Los operadores lógicos se utilizan para determinar la lógica entre variables o valores:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Nombre** | **Descripción** | **Ejemplo** |
| && | AND | Devuelve verdadero si ambas declaraciones son verdaderas | x < 5 &&  x < 10 |
| || | OR | Devuelve verdadero si una de las afirmaciones es verdadera | x < 5 || x < 4 |
| ! | NOT | Invierte el resultado, devuelve falso si el resultado es verdadero | !(x < 5 && x < 10) |

## Fundamentos de cadenas y caracteres

Las cadenas se utilizan para almacenar texto.

Una variable de tipo string contiene una colección de caracteres entre comillas dobles:

string saludo = "Hola";

### Length

Un string en C# es en realidad un objeto que contiene propiedades y métodos que pueden realizar ciertas operaciones en strings. Por ejemplo, la longitud de una cadena se puede encontrar con la propiedad Length:

string saludo = "Hola";

Console.WriteLine("La longitud de saludo en caracteres es de: {0}", saludo.Length);

// 4

El contenedor {0} es un mensaje con formato, significa imprimir/formatear el objeto de índice cero que insertó en los argumentos de los parámetros de su función. Es un número basado en cero que obtiene el índice de objeto que desea, aquí hay un ejemplo:

string saludo = "Hola";

string nombre = "José";

Console.WriteLine("{0}, mi nombre es {1}", saludo, nombre);

// Hola, mi nombre es José

Es decir, sustituye los contenedores por los argumentos.

### Concatenación de strings

El operador + se puede usar entre cadenas para combinarlas. Esto se llama concatenación.

string nombre = "José";

string apellido = "Rivera";

string nombreCompleto = nombre + " " + apellido;

// José Rivera

### Interpolación de cadenas

Otra opción de concatenación de cadenas es la interpolación de cadenas, que sustituye valores de variables en marcadores de posición en una cadena. Tenga en cuenta que no tiene que preocuparse por los espacios, como con la concatenación:

string nombre = "José";

string apellido = "Rivera";

string nombreCompleto = $"Mi nombre completo es {nombre} {apellido}";

// Mi nombre completo es José Rivera

### Cadenas de acceso

Puede acceder a los caracteres de una cadena haciendo referencia a su número de índice entre corchetes [ ].

Este ejemplo imprime el primer carácter de saludo:

string saludo = "Hola";

Console.WriteLine(saludo[0]);

// H

También puede encontrar la posición de índice de un carácter específico en una cadena, utilizando el método IndexOf():

string saludo = "Hola";

Console.WriteLine(saludo.IndexOf("a"));

// 3

### Otros métodos

Hay muchos métodos de cadena disponibles, por ejemplo, ToUpper() y ToLower(), que devuelven una copia de la cadena convertida a mayúsculas o minúsculas respectivamente.

string saludo = "Hola";

Console.WriteLine(saludo.ToUpper()); // HOLA

Console.WriteLine(saludo.ToLower()); // hola

### Caracteres especiales

Debido a que las cadenas deben escribirse entre comillas, C# malinterpretará esta cadena y generará un error:

string texto = "La siguiente "palabra" está entre comillas dobles";

La solución para evitar este problema es utilizar el carácter de escape de barra invertida. El carácter de escape de barra invertida (\) convierte los caracteres especiales en caracteres de cadena:

|  |  |  |
| --- | --- | --- |
| **Carácter de escape** | **Resultado** | **Descripción** |
| \' | ' | Comilla simple |
| \" | " | Comillas dobles |
| \\ | \ | Barra invertida |

string texto = "La siguiente \"palabra\" está entre comillas dobles";

// La siguiente "palabra" está entre comillas dobles

Otros caracteres de escape útiles en C# son:

|  |  |
| --- | --- |
| **Carácter de escape** | **Resultado** |
| \n | Salto de línea |
| \t | Tabulador |
| \b | Retroceso |

# Bibliografía

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/variables>

<https://www.discoduroderoer.es/variables-en-c-sharp/>

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/types>

<https://www.w3schools.com/cs/cs_operators.php>

<https://www.udemy.com/course/c-sharp-para-principiantes/learn/lecture/4615546>

<https://www.w3schools.com/cs/cs_strings.php>