Microsoft .net

MICROSOFT® VISUAL C#® .NET

APRENDA YA

John Sharp
Jon Jagger

McGraw-Hill Profesional

Microsoft®

Importante Las dos primeras recomendaciones deberían considerarse como obligatorias, ya que se refieren a la compatibilidad con la Common Language Specification (CLS). Si se desea escribir programas que puedan operar con otros lenguajes, como Microsoft Visual Basic .NET, es necesario acatar dichas recomendaciones.

Por ejemplo, *marcador*, *cuarentayDos*, *_marcador* y *CuarentayDos* son todos ellos nombres de variable válidos, pero sólo se recomienda utilizar los dos primeros.

Declaración de variables

Hay que tener en cuenta que las variables son como cajas de memoria que pueden contener un valor. C# puede procesar muchos tipos diferentes de valores, números enteros, números en coma flotante y cadenas de caracteres, por nombrar tres de ellos. Cuando se declara una variable es necesario indicar el tipo de los datos que va a contener.

El tipo y el nombre de una variable se declara en una sentencia de declaración. Por ejemplo, la sentencia siguiente declara que la variable de nombre *edad* contiene valores *int* (enteros); y se incluye el punto y coma, por supuesto:

```
int edad;
```

Importante Los programadores de Microsoft Visual Basic deberían tener en cuenta que C# no permite declaraciones implícitas. Es necesario declarar de forma explícita todas las variables antes de utilizarlas para que sea posible compilar el código.

El tipo de variable *int* es un ejemplo de palabra clave. Se trata del nombre de uno de los tipos primitivos de C#: entero.

Más adelante en este capítulo se tratan varios tipos de datos primitivos. Después de declarar una variable es posible asignarle un valor. La sentencia siguiente asigna a *edad* el valor 42. De nuevo, puede comprobarse que es necesario agregar un punto y coma.

```
edad = 42;
```

El símbolo = es el operador de *asignación*, el cual asigna el valor de su derecha a la variable de su izquierda. Después de esta asignación, se puede utilizar la variable *edad* en el código para hacer referencia al valor que contiene. La sentencia siguiente escribe *42* en la consola:

```
Console.WriteLine(edad);
```

Consejo Si se coloca el puntero del ratón sobre una variable en el panel de Código de Visual Studio .NET aparece información acerca del tipo de la variable. Con esta característica, no existe ninguna razón para utilizar la notación húngara en Visual Studio .NET.

Tipos de datos primitivos

C# posee cierto número de tipos incorporados, denominados *tipos de datos primitivos*. La tabla siguiente muestra un listado con los tipos de datos primitivos de C# utilizados habitualmente. Los tipos *int* y *long* contienen números enteros, donde *long* puede contener números mayores que *int*, pero necesita más memoria; los tipos *float* y *double* contienen números con puntos decimales, donde *double* contiene valores con una precisión doble que *float*, de ahí su nombre; el tipo *decimal* se ha diseñado para contener valores de moneda de hasta 28 cifras significativas; el tipo *char* contiene caracteres individuales, en representación Unicode; el tipo *string* contiene secuencias de caracteres y el tipo *bool* contiene true (verdadero) o false (falso).

Variables locales no asignadas

Cuando se declara, una variable contiene un valor indeterminado hasta que se le asigna uno. Esto constituye una enorme fuente de errores en programas en C y C++ que crean una variable y la utilizan como fuente de información antes de asignarle un valor. C# no lo permite. Es necesario asignar un valor a una variable antes de utilizarla, en caso contrario el programa no se compilará. Este requisito se conoce como *Regla de asignación definitiva (Definite Assignment Rule)*. Por ejemplo, la sentencia siguiente originará un error de compilación debido a que *edad* no está asignada:

```
int edad;
Console.WriteLine(edad); // error de compilación
```

Tipo de datos	Descripción	Tamaño (bits)	Intervalo*	Ejemplo de uso
int	Números enteros	32	-2^{31} a $2^{31}-1$	*int contador; contador = 42;*
long	Números enteros (intervalo mayor)	64	-2^{63} a $2^{63}-1$	*long espera; espera = 42L;*
float	Números en coma flotante	32	$\pm 3{,}4 \times 10^{38}$	*float lejos; lejos = 0,42F;*
double	Números en coma flotante con precisión doble (más exactos)	64	$\pm 1{,}7 \times 10^{308}$	*double problema; problema = 0,42;*
decimal	Valores monetarios	128	28 cifras significativas	*decimal moneda; moneda = 0,42M;*
string	Secuencia de caracteres	16 bits por carácter	No aplicable	*string chaleco; chaleco = "42";*
char	Carácter individual	16	0 a $2^{16}-1$	*char parrilla; parrilla = '4';*
bool	Booleano (verdadero o falso)	8	true (verdadero) o false (falso)	*bool dientes; dientes = false;*

* El valor de 2^{16} es 32.768, el valor de 2^{31} es 2.147.483.648 y el valor de 2^{63} es 9.223.372.036.854.775.808.

Valores de tipos de datos primitivos

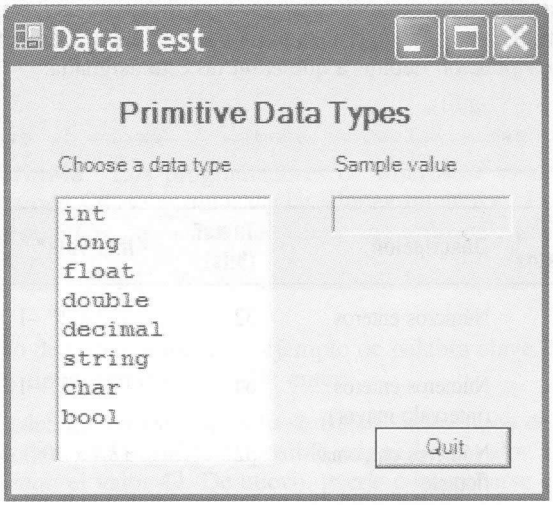
En el ejercicio siguiente se utiliza un programa en C#, denominado PrimitiveDataTypes, para mostrar el funcionamiento de varios tipos de datos primitivos.

Valores de tipos de datos primitivos

1. Inicie Visual Studio .NET.
2. En el menú Archivo, seleccione Abrir y, a continuación, haga clic en Proyecto.

   Se abre el cuadro de diálogo Abrir proyecto.
3. Abra el proyecto PrimitiveDataTypes, que se encuentra en la carpeta \Visual C# Step by Step\Chapter 2\PrimitiveDataTypes.
4. En el menú Depurar, haga clic en Iniciar.

   Aparece la ventana de aplicación siguiente.



5. En la lista Choose a data type, seleccione String.

   Aparece el valor "42" en el cuadro Sample Value.
6. Seleccione Int de la lista.

   Aparece el valor *"to do"* en el cuadro Sample Value, indicando que aún están por escribir las sentencias para mostrar un valor *int*.
7. Seleccione cada uno de los tipos de datos de la lista.

   Confirme que también es necesario realizar el código para los tipos *double* y *bool*.
8. Haga clic en Quit para detener el programa.

   El control vuelve al entorno de programación de Visual Studio .NET.

Utilización de tipos de datos primitivos en el código

1. Haga clic en la ficha Form1.cs del panel de Código. Haga clic con el botón secundario del ratón en el formulario y seleccione Ver código si la ficha Form1.cs no está visible.

   Se cierra la vista Diseño y se abre el panel de Código.

2. En el panel de Código, busque el método *showFloatValue* que se muestra a continuación:

```
private void showFloatValue()
{
    float var;
    var = 0.42F;
    value.Text = "0.42F";
}
```

Consejo Para buscar un elemento de un proyecto, seleccione Buscar y reemplazar en el menú Edición y haga clic en Buscar símbolo. Se abre un cuadro de diálogo que solicita qué se está buscando. Escriba el nombre del elemento a buscar y, a continuación, haga clic en Buscar. Si no está seguro de si el elemento contiene letras mayúsculas, desactive la casilla de verificación Coincidir mayúsculas y minúsculas. También puede probar las demás opciones. Si desea obtener más información acerca de las opciones, haga clic en Ayuda en el cuadro de diálogo.

   Se ejecuta el método *showFloatValue* al hacer clic en el tipo *float* del cuadro de lista. Este método contiene tres sentencias. La primera declara una variable denominada *var* de tipo *float*. La segunda sentencia asigna el valor 0,42F a *var*. La F es un sufijo de tipo que indica que hay que tratar a 0,42 como un valor *float*. Si se omite la F, el valor 0,42 se tratará como *double* y el programa no se compilará debido a que no es posible asignar un valor de un tipo a una variable de otro tipo de esta forma. La tercera sentencia asigna a la propiedad *value.Text* la cadena "0,42F". Más adelante se enseñarán más cosas acerca de las propiedades pero, de momento, se las puede considerar similares a las variables. La variable *value* hace referencia al cuadro de texto de la pantalla. Dicha variable se declara en otro lugar del programa.

3. En el panel de Código, busque el método *showIntValue* que se muestra a continuación:

```
private void showIntValue()
{
    value.Text = "to do";
}
```

   Se llama al método *showIntValue* al hacer clic en Int en el cuadro de lista. Este método contiene una sentencia.

4. Escriba las dos sentencias siguientes al comienzo del método *showIntValue:*

```
int var;
var = 42;
```

El método *showIntValue* debería tener un aspecto similar al siguiente:

```
private void showIntValue()
{
    int var;
    var = 42;
    value.Text = "to do";
}
```

5. En el menú Generar, haga clic en Generar solución.

   Se realiza la generación.

6. En la sentencia original, cambie la cadena *"to do"* por *"42"*.

   Ahora el método debería ser exactamente igual al siguiente:

```
private void showIntValue()
{
    int var;
    var = 42;
    value.Text = "42";
}
```

7. En el menú Depurar, haga clic en Iniciar.

   Aparece la ventana de la aplicación.

Consejo Si ha editado el código fuente desde la última generación, el comando Iniciar vuelve a generar de manera automática el programa antes de iniciar la aplicación.

---

8. Seleccione int en el cuadro de lista.

   Confirme que se muestra el valor *42* en el cuadro Sample Value.

9. Haga clic en Quit para detener el programa.

10. En el panel de Código, busque el método *showDoubleValue*.

11. Edite el método *showDoubleValue* exactamente como se muestra a continuación:

```
private void showDoubleValue()
{
    double var;
    var = 0.42;
    value.Text = "0.42";
}
```

12. En el panel de Código, busque el método *showBoolValue*.

13. Edite el método *showBoolValue* exactamente como se muestra a continuación:

```
private void showBoolValue()
{
```