100 XP

Exercise - Setting and getting values from variables

10 minutes

Because variables are temporary storage containers for data, they're meant to be *written to* and *read from*. You'll get a chance to do both in the following exercise.

Exercise - Working with variables

In this exercise, you'll declare a variable, assign it a value, retrieve its value, and more.

Create your first variable

1. Select all of the code in the .NET Editor, and press `Delete` or `Backspace` to delete it.

2. Enter the following code in the code editor:

C#

```
string firstName;
firstName = "Bob";
```

To declare a variable, you enter the data type you want to use followed by a name for the variable. To assign a value to a variable, you use the *assignment operator*, which is a single equals character `=`.

Note

Assigning a value is also referred to as "setting the variable", or simply, a "set" operation.

Improperly assign a value to a variable

It's important to notice that assignment happens from right to left. In other words, the C# compiler must first understand the value on the right side of the assignment operator, then it can perform the assignment to the variable on the left side of the assignment operator. If you reverse the order, you'll confuse the C# compiler.

1. Modify the code you wrote to match the following code:

   C#

   ```
   string firstName;
   "Bob" = firstName;
   ```

2. Now, run the code. You'll see the following error in the output console:

   Output

   ```
   (2,1): error CS0131: The left-hand side of an assignment must be a variable,
   property or indexer
   ```

Improperly assign a value of the incorrect data type to the variable

You learned that C# was designed to enforce types. When you're working with variables, *enforcing types* means you can't assign a value of one data type to a variable declared to hold a different data type.

1. Modify the code you wrote to match the following code:

   C#

   ```
   int firstName;
   firstName = "Bob";
   ```

2. Now, run the code. You'll see the following error in the output console:

   Output

   ```
   (2,9): error CS0029: Cannot implicitly convert type 'string' to 'int'
   ```

The error message hints at what the C# compiler tries to do behind the scenes. It tried to "implicitly convert" the string "Bob" to be an int value; however, that is impossible. Even so, C# tried to do the conversion but fails since there's no numeric equivalent for the word "Bob".

You'll learn more about implicit and explicit type conversion later. For now, just remember that a variable can only hold values matching its specified data type.

Retrieve a value you stored in the variable

To retrieve a value from a variable, you just use the name of the variable. This example will set a variable's value, then retrieve that value and display it in the console.

1. Modify the code you wrote to match the following code:

C#

```csharp
string firstName;
firstName = "Bob";
Console.WriteLine(firstName);
```

2. Now, run the code. You'll see the following result in the output console:

Output

```
Bob
```

Retrieving a value from a variable is also referred to as "getting the variable", or simply, a "get" operation.

As you write lines of code, you'll see that the compiler is checking your code and spotting possible mistakes. The compiler is a great tool to help you get code correct sooner. Now that you're familiar with different types of errors, you can quickly fix mistakes with the help of the compiler's error messages.

Reassign the value of a variable

You can reuse and reassign the variable as many times as you want. This example illustrates that idea.

1. Modify the code you wrote to match the following code:

C#

```
string firstName;
firstName = "Bob";
Console.WriteLine(firstName);
firstName = "Liem";
Console.WriteLine(firstName);
firstName = "Isabella";
Console.WriteLine(firstName);
firstName = "Yasmin";
Console.WriteLine(firstName);
```

2. Now, run the code. You'll see the following result in the output console:

Output

```
Bob
Liem
Isabella
Yasmin
```

Initialize the variable

You must *set* a variable to a value before you can *get* the value from the variable. Otherwise, you'll see an error.

1. Modify the code you wrote in Step 6 to match the following code:

C#

```
string firstName;
Console.WriteLine(firstName);
```

2. Now, run the code. You'll see the following result in the output console:

Output

```
(2,19): error CS0165: Use of unassigned local variable 'firstName'
```

To avoid the possibility of an unassigned local variable, it is recommended that you set the value as soon as possible after you declare it.

In fact, you can perform both the declaration and setting the value of the variable in one line of code. This technique is called *initializing* the variable.

1. Modify the code you wrote to match the following code:

   C#

   ```
   string firstName = "Bob";
   Console.WriteLine(firstName);
   ```

2. Now, run the code. You should see the following output:

   Output

   ```
   Bob
   ```

Recap

Here's what you've learned about working with variables so far:

- You must assign (set) a value to a variable before you can retrieve (get) a value from a variable.
- You can initialize a variable by assigning a value to the variable at the point of declaration.
- Assignment happens from right to left.
- You use a single equals character as the assignment operator.
- To retrieve the value from the variable, you merely use the variable's name.

Module complete:

Unlock achievement

English (United States) Your Privacy Choices Theme

Manage cookies Previous Versions Blog Contribute Privacy Terms of Use Trademarks

© Microsoft 2023

.NET Editor

Press CTRL+M, TAB to exit the editor

Clear Run

Output