100 XP

Exercise - Combine strings using string interpolation

10 minutes

While string concatenation is simple and convenient, *string interpolation* is growing in popularity in situations where you need to combine many literal strings and variables into a single formatted message.

What is string interpolation?

String interpolation combines multiple values into a single literal string by using a "template" and one or more *interpolation expressions*. An **interpolation expression** is indicated by an opening and closing curly brace symbol `{ }`. You can put any C# expression that returns a value inside the braces. The literal string becomes a template when it's prefixed by the `$` character.

In other words, instead of writing the following line of code:

```C#
string message = greeting + " " + firstName + "!";
```

You can write this more concise line of code instead:

```C#
string message = $"{greeting} {firstName}!";
```

In this simple example, you save a few keystrokes. You can imagine how much more concise string interpolation can be in more complex operations. Moreover, many find the string interpolation syntax cleaner and easier to read.

In the following exercise, you'll rewrite the previous messages using string interpolation.

Use string interpolation to combine a literal string and a variable value

To interpolate two strings together, you create a literal string and prefix the string with the `$` symbol. The literal string should contain at least one set of curly braces `{}` and inside of those characters you use the name of a 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 .NET Editor:

C#

```csharp
string firstName = "Bob";
string message = $"Hello {firstName}!";
Console.WriteLine(message);
```

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

Output

```
Hello Bob!
```

Use string interpolation with multiple variables and literal strings

You can perform several interpolation operations in the same line of code.

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

c#

```csharp
int version = 11;
string updateText = "Update to Windows";
string message = $"{updateText} {version}";
Console.WriteLine(message);
```

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

Output

```
Update to Windows 11
```

Avoid intermediate variables

Just as you did in the previous exercise, you can eliminate the temporary variable to store the message.

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

C#

```c#
int version = 11;
string updateText = "Update to Windows";
Console.WriteLine($"{updateText} {version}!");
```

2. Now, run the code. The result in the output console should be the same even if you simplified the code:

Output

```
Update to Windows 11!
```

Combine verbatim literals and string interpolation

Suppose you need to use a verbatim literal in your template. You can use both the verbatim literal prefix symbol `@` and the string interpolation `$` symbol together.

1. Delete the code from the previous steps and type the following code into the .NET Editor:

C#

```c#
string projectName = "First-Project";
Console.WriteLine($@"C:\Output\{projectName}\Data");
```

2. Now, run the code and you should see the following result.

Output

```
C:\Output\First-Project\Data
```

In this example, the `$` symbol allows you to reference the `projectName` variable inside the braces, while the `@` symbol allows you to use the unescaped `\` character.

Recap

Here's what you've learned about string interpolation so far:

- String interpolation provides an improvement over string concatenation by reducing the number of characters required in some situations.
- You can combine string interpolation and verbatim literals by combining the symbols for each and using that as a prefix for the string template.

Module complete:

Unlock achievement

.NET Editor

Press CTRL + M, TAB to exit the editor

Clear Run

```
```

Output