200 XP

Exercise - Use the string's IndexOf() and Substring() helper methods

25 minutes

In this exercise, you use the `IndexOf()` method to locate the position of one or more characters string inside a larger string. You use the `Substring()` method to return the part of the larger string that follows the character positions you specify.

You'll also use an overloaded version of the `Substring()` method to set the length of characters to return after a specified position in a string.

Prepare your coding environment

This module includes hands-on activities that guide you through the process of building and running demonstration code. You're encouraged to complete these activities using Visual Studio Code as your development environment. Using Visual Studio Code for these activities help you to become more comfortable writing and running code in a developer environment that's used by professionals worldwide.

> **Note**
>
> If you have completed other Microsoft Learn modules in this C# series, you may have already created a project folder for code samples. If that's the case, you can skip the following section of steps, and delete the code in the `Project.cs` file used for a previous exercise.

1. Open Visual Studio Code.

   You can use the Windows Start menu (or equivalent resource for another OS) to open Visual Studio Code.

2. On the Visual Studio Code **File** menu, select **Open Folder**.

3. In the **Open Folder** dialog, navigate to the Windows Desktop folder.

If you have different folder location where you keep code projects, you can use that folder location instead. For this training, the important thing is to have a location that's easy locate and remember.

4. In the **Open Folder** dialog, select **Select Folder**.

   If you see a security dialog asking if you trust the authors, select **Yes**.

5. On the Visual Studio Code **Terminal** menu, select **New Terminal**.

   Notice that a command prompt in the Terminal panel displays the folder path for the current folder. For example:

   ```dos
   C:\Users\someuser\Desktop>
   ```

6. At the Terminal command prompt, to create a new console application in a specified folder, type **dotnet new console -o ./CsharpProjects/TestProject** and then press Enter.

   This .NET CLI command uses a .NET program template to create a new C# console application project in the specified folder location. The command creates the CsharpProjects and TestProject folders for you, and uses TestProject as the name of the `.csproj` file.

7. In the EXPLORER panel, expand the **CsharpProjects** folder.

   You should see the TestProject folder and two files, a C# program file named Program.cs and a C# project file named TestProject.csproj.

8. In the EXPLORER panel, to view your code file in the Editor panel, select **Program.cs**.

9. Delete the existing code lines.

   You are using this C# console project to create, build, and run code samples during this module.

10. Close the Terminal panel.

Write code to find parenthesis pairs embedded in a string

1. Ensure that you have Visual Studio Code open and Program.cs displayed in the Editor panel.

> **Note**
>
> Program.cs should be empty. If if isn't, select and delete all code lines.

2. Type the following code into the Visual Studio Code Editor:

```
C#

string message = "Find what is (inside the parentheses)";

int openingPosition = message.IndexOf('(');
int closingPosition = message.IndexOf(')');

Console.WriteLine(openingPosition);
Console.WriteLine(closingPosition);
```

3. On the Visual Studio Code **File** menu, select **Save**.

   The Program.cs file must be saved before building or running the code.

4. In the EXPLORER panel, to open a Terminal at your TestProject folder location, right-click **TestProject**, and then select **Open in Integrated Terminal**.

   A Terminal panel should open, and should include a command prompt showing that the Terminal is open to your TestProject folder location.

5. At the Terminal command prompt, to run your code, type **dotnet run** and then press Enter.

> **Note**
>
> If you see a message saying "Couldn't find a project to run", ensure that the Terminal command prompt displays the expected TestProject folder location. For example: `C:\Users\someuser\Desktop\csharpprojects\TestProject>`

   You should see the following output:

```
Output

13
36
```

In this case, the index of the `(` character is 13. Remember, these values are zero-based, so it's the 14th character in the string. The index of the `)` character is `36`.

Now that you have the two indexes, you can use them as the boundaries to retrieve the value between them.

Add code to retrieve the value between parenthesis

1. Update your code in the Visual Studio Code Editor as follows:

```
C#

string message = "Find what is (inside the parentheses)";

int openingPosition = message.IndexOf('(');
int closingPosition = message.IndexOf(')');

// Console.WriteLine(openingPosition);
// Console.WriteLine(closingPosition);

int length = closingPosition - openingPosition;
Console.WriteLine(message.Substring(openingPosition, length));
```

2. Save your code file, and then use Visual Studio Code to run your code. You should see the following output:

```
Output

(inside the parentheses
```

The `Substring()` method needs the starting position and the number of characters, or length, to retrieve. So, you calculate the length in a temporary variable called `length`, and pass it with the `openingPosition` value to retrieve the string inside of the parenthesis.

The result is close, however the output includes the opening parenthesis. In this exercise, the inclusion of the parenthesis isn't desired. To remove the parenthesis from output, you have to update the code to skip the index of the parenthesis itself.

Modify the starting position of the sub string

1. Update your code in the Visual Studio Code Editor as follows:

C#

```
string message = "Find what is (inside the parentheses)";

int openingPosition = message.IndexOf('(');
int closingPosition = message.IndexOf(')');

openingPosition += 1;

int length = closingPosition - openingPosition;
Console.WriteLine(message.Substring(openingPosition, length));
```

2. Save your code file, and then use Visual Studio Code to run your code. You should see the following output:

Output

```
inside the parentheses
```

3. Take a moment to review the previous code and the line `openingPosition += 1;`.

   By increasing the `openingPosition` by `1`, you skip over the opening parenthesis character.

   The reason you're using the value `1` is because that is the length of the character. If you attempt to locate a value starting after a longer string, for example, `<div>` or `---`, you would use the length of that string instead.

4. Update your code in the Visual Studio Code Editor as follows:

C#

```
string message = "What is the value <span>between the tags</span>?";

int openingPosition = message.IndexOf("<span>");
int closingPosition = message.IndexOf("</span>");

openingPosition += 6;
int length = closingPosition - openingPosition;
Console.WriteLine(message.Substring(openingPosition, length));
```

5. Take a moment to review the previous code and the line `openingPosition += 6;`.

   The preceding snippet of code shows how to find the value inside an opening and closing `<span>` tag.

In this case, you're adding `6` to the `openingPosition` as the offset to calculate the length of the sub string.

Avoid magic values

Hardcoded strings like `"<span>"` in the previous code listing are known as "magic strings" and hardcoded numeric values like `6` are known as "magic numbers". These "Magic" values are undesirable for many reasons and you should try to avoid them if possible.

1. Review the previous code to consider how the code might break if you hardcoded the string `"<span>"` multiple times in your code, but misspelled one instance of it as `"<sapn>"`.

   The compiler won't catch `"<sapn>"` at compile time because the value is in a string. The misspelling will likely cause problems at run time, and depending on the complexity of your code, it might be difficult to track down.

   Furthermore, if you change the string `"<span>"` to `"<div>"`, but forget to change the number `6`, then your code will produce undesirable results.

2. Update your code in the Visual Studio Code Editor as follows:

   C#

   ```
   string message = "What is the value <span>between the tags</span>?";

   const string openSpan = "<span>";
   const string closeSpan = "</span>";

   int openingPosition = message.IndexOf(openSpan);
   int closingPosition = message.IndexOf(closeSpan);

   openingPosition += openSpan.Length;
   int length = closingPosition - openingPosition;
   Console.WriteLine(message.Substring(openingPosition, length));
   ```

3. Take a minute to examine the updated code and the use of the keyword `const` as used in `const string openSpan = "<span>";`.

   The code uses a constant with the `const` keyword. A constant allows you to define and initialize a variable whose value can never be changed. You would then use that constant in the rest of the code whenever you needed that value. This ensures that the value is only defined once and misspelling the `const` variable will be caught by the compiler.

The previous code listing is a safer way to write the same code you examined in the previous section. Now, if the value of `openSpan` changes to `<div>`, the line of code that uses the `Length` property continues to be valid.

Recap

This unit covered much material. Here's the most important things to remember:

- `IndexOf()` gives you the first position of a character or string inside of another string.
- `IndexOf()` returns `-1` if it can't find a match.
- `Substring()` returns just the specified portion of a string, using a starting position and optional length.
- There's often more than one way to solve a problem. You used two separate techniques to find all instances of a given character or string.
- Avoid hardcoded **magic values**. Instead, define a `const` variable. A constant variable's value can't be changed after initialization.

Check your knowledge

1. What is the return value of `myString.IndexOf('C');` where `string myString = "C# Time";`? *

- ◯ `0`
- ◯ `1`
- ◯ `-1`

Check your answers