100 XP

Exercise - Complete a challenge activity to differentiate between do and while iteration statements

14 minutes

Code challenges will reinforce what you've learned and help you gain some confidence before continuing on.

Examine the difference between do and while statement iterations

As you have seen, C# supports four types of iteration statements: `for`, `foreach`, `do-while`, and `while`. Microsoft's language reference documentation describes these statements as follows:

- The `for` statement: executes its body while a specified Boolean expression (the 'condition') evaluates to true.
- The `foreach` statement: enumerates the elements of a collection and executes its body for each element of the collection.
- The `do-while` statement: conditionally executes its body one or more times.
- The `while` statement: conditionally executes its body zero or more times.

The `for` and `foreach` iterations seem to be clearly differentiated from each other and from the `do-while` and `while` iterations. The definitions for the `do-while` and `while` statements, however, appear to be quite similar. Knowing when to choose between a `do-while` and a `while` seems more arbitrary, and can even be a bit confusing. Some challenge projects may help to make the differences clear.

In this challenge, you'll be presented with conditions for three separate coding projects. Each project will require you to implement an iteration code block using either a `do-while` or a `while` statement. You'll need to evaluate the specified conditions in order to choose between the `do-while` and `while` statements. You can switch after you start if your first choice isn't working out as well as you had hoped.

> **Note**
>
> The conditions for your coding project can be used to help you select between the `do-while` and `while` statements. What you know, or don't know, about the Boolean expression that will be evaluated can sometimes help you to select between the `do-while` and `while` statements. In this challenge exercise, the project conditions include information that will be used to construct the Boolean expression.

Manage user input during this challenge

When using a `Console.ReadLine()` statement to obtain user input, it's common practice to use a nullable type string (designated `string?`) for the input variable and then evaluate the value entered by the user. The following code sample uses a nullable type string to capture user input. The iteration continues while the user-supplied value is null:

C#

```
string? readResult;
Console.WriteLine("Enter a string:");
do
{
    readResult = Console.ReadLine();
} while (readResult == null);
```

The Boolean expression evaluated by the `while` statement can be used to ensure user input meets a specified requirement. For example, if a prompt asks the user to enter a string that includes at least three characters, the following code could be used:

C#

```
string? readResult;
bool validEntry = false;
Console.WriteLine("Enter a string containing at least three characters:");
do
{
    readResult = Console.ReadLine();
    if (readResult != null)
    {
        if (readResult.Length >= 3)
        {
            validEntry = true;
```

```
        }
        else
        {
            Console.WriteLine("Your input is invalid, please try again.");
        }
    }
} while (validEntry == false);
```

If you want to use `Console.ReadLine()` input for numeric values, you need to convert the string value to a numeric type.

The `int.TryParse()` method can be used to convert a string value to an integer. The method uses two parameters, a string that will be evaluated and the name of an integer variable that will be assigned a value. The method returns a Boolean value. The following code sample demonstrates using the `int.TryParse()` method:

C#

```
// capture user input in a string variable named readResult

int numericValue = 0;
bool validNumber = false;

validNumber = int.TryParse(readResult, out numericValue);
```

If the string value assigned to `readResult` represents a valid integer, the value will be assigned to the integer variable named `numericValue`, and `true` will be assigned to the Boolean variable named `validNumber`. If the value assigned to `readResult` doesn't represent a valid integer, `validNumber` will be assigned a value of `false`. For example, if `readResult` is equal to "7", the value `7` will be assigned to `numericValue`.

Code project 1 - write code that validates integer input

Here are the conditions that your first coding project must implement:

- Your solution must include either a `do-while` or `while` iteration.
- Before the iteration block: your solution must use a `Console.WriteLine()` statement to prompt the user for an integer value between 5 and 10.

- Inside the iteration block:
  - Your solution must use a `Console.ReadLine()` statement to obtain input from the user.
  - Your solution must ensure that the input is a valid representation of an integer.
  - If the integer value isn't between 5 and 10, your code must use a `Console.WriteLine()` statement to prompt the user for an integer value between 5 and 10.
  - Your solution must ensure that the integer value is between 5 and 10 before exiting the iteration.
- Below (after) the iteration code block: your solution must use a `Console.WriteLine()` statement to inform the user that their input value has been accepted.

1. Ensure that you have an empty Program.cs file open in Visual Studio Code.

   If necessary, open Visual Studio Code, and then complete the following steps to prepare a Program.cs file in the Editor:

   a. On the **File** menu, select **Open Folder**.

   b. Use the Open Folder dialog to navigate to, and then open, the **CsharpProjects** folder.

   c. In the Visual Studio Code EXPLORER panel, select **Program.cs**.

   d. On the Visual Studio Code **Selection** menu, select **Select All**, and then press the Delete key.

2. Write the code that implements each condition for code project 1.

3. Run your application and verify that your code validates user input based on the specified requirements.

   For example, when you run your application, it should reject input values such as "two" and "2", but it should accept an input value of "7".

   The example described above, the console output should look similar to the following:

   Output

   ```
   Enter an integer value between 5 and 10
   two
   Sorry, you entered an invalid number, please try again
   2
   You entered 2. Please enter a number between 5 and 10.
   ```

```
7
Your input value (7) has been accepted.
```

Code project 2 - write code that validates string input

Here are the conditions that your second coding project must implement:

- Your solution must include either a `do-while` or `while` iteration.
- Before the iteration block: your solution must use a `Console.WriteLine()` statement to prompt the user for one of three role names: Administrator, Manager, or User.
- Inside the iteration block:
  - Your solution must use a `Console.ReadLine()` statement to obtain input from the user.
  - Your solution must ensure that the value entered matches one of the three role options.
  - Your solution should use the `Trim()` method on the input value to ignore leading and trailing space characters.
  - Your solution should use the `ToLower()` method on the input value to ignore case.
  - If the value entered isn't a match for one of the role options, your code must use a `Console.WriteLine()` statement to prompt the user for a valid entry.
- Below (after) the iteration code block: Your solution must use a `Console.WriteLine()` statement to inform the user that their input value has been accepted.

1. Comment out all of the code in the Visual Studio Code Editor panel
   a. Select all of the code lines in the code editor
   b. On the **Edit** menu, select **Toggle Block Comment**.
2. Write the code that implements each condition for code project 2.
3. Run your application and verify that your code validates user input based on the specified requirements.

   For example, when you run your application, it should reject an input value such as "Admin", but it should accept an input value of " administrator ".

   The console output for this example should look similar to the following:

```
Output
```

```
Enter your role name (Administrator, Manager, or User)
Admin
The role name that you entered, "Admin" is not valid. Enter your role name
(Administrator, Manager, or User)
   Administrator
Your input value (Administrator) has been accepted.
```

Code project 3 - Write code that processes the contents of a string array

Here are the conditions that your third coding project must implement:

- your solution must use the following string array to represent the input to your coding logic:

```c#
string[] myStrings = new string[2] { "I like pizza. I like roast chicken. I like salad", "I like all three of the menu choices" };
```

- Your solution must declare an integer variable named `periodLocation` that can be used to hold the location of the period character within a string.
- Your solution must include an outer `foreach` or `for` loop that can be used to process each string element in the array. The string variable that you'll process inside the loops should be named `myString`.
- In the outer loop, your solution must use the `IndexOf()` method of the `String` class to get the location of the first period character in the `myString` variable. The method call should be similar to: `myString.IndexOf(".")`. If there's no period character in the string, a value of -1 will be returned.
- Your solution must include an inner `do-while` or `while` loop that can be used to process the `myString` variable.
- In the inner loop, your solution must extract and display (write to the console) each sentence that is contained in each of the strings that are processed.
- In the inner loop, your solution must not display the period character.

- In the inner loop, your solution must use the `Remove()`, `Substring()`, and `TrimStart()` methods to process the string information.

1. Comment out all of the code in the Visual Studio Code Editor panel
   a. Select all of the code lines in the code editor
   b. On the **Edit** menu, select **Toggle Block Comment**.

2. Write the code that implements each condition listed for code project 3.

3. Run your application and verify that your output meets the requirements.

   If your code logic works correctly, your output should look similar to the following:

   Output

```
I like pizza
I like roast chicken
I like salad
I like all three of the menu choices
```

Whether you get stuck and need to peek at the solution or you finish successfully, continue on to view the solution projects for this challenge.

Next unit: Review the solution to do versus while challenge activity

Continue >