100 XP

Review the solution to do versus while challenge activity

8 minutes

The following examples should use a `do` because you know that you need to execute the code block at least once. You CAN also use a `while` to achieve the same result. Some developers feel that the logic of a `while` makes the code more readable. If that is the case for you, you can choose to implement a `while`. In this case, be aware that most code compilers will optimize your code for you by converting the iteration statement to a `do-while`.

Project 1 code

The following code is one possible solution for challenge project 1 from the previous unit.

The code uses a `do` statement because the code block must be executed at least once. You CAN also use a `while` to achieve the same result. Some developers may feel that the logic of a `while` makes the code more readable. If that's the case for you, you can choose to implement a `while` statement here.

C#

```
string? readResult;
string valueEntered = "";
int numValue = 0;
bool validNumber = false;

Console.WriteLine("Enter an integer value between 5 and 10");

do
{
    readResult = Console.ReadLine();
    if (readResult != null)
    {
        valueEntered = readResult;
    }

    validNumber = int.TryParse(valueEntered, out numValue);
```

```
    if (validNumber == true)
    {
        if (numValue <= 5 || numValue >= 10)
        {
            validNumber = false;
            Console.WriteLine($"You entered {numValue}. Please enter a number between
5 and 10.");
        }
    }
    else
    {
        Console.WriteLine("Sorry, you entered an invalid number, please try again");
    }
} while (validNumber == false);

Console.WriteLine($"Your input value ({numValue}) has been accepted.");

readResult = Console.ReadLine();
```

Project 2 code

The following code is one possible solution for challenge project 2 from the previous unit.

The code uses a `do` statement because the code block must be executed at least once. You CAN also use a `while` to achieve the same result. Some developers may feel that the logic of a `while` makes the code more readable. If that's the case for you, you can choose to implement a `while` statement here.

C#

```
string? readResult;
string roleName = "";
bool validEntry = false;

do
{
    Console.WriteLine("Enter your role name (Administrator, Manager, or User)");
    readResult = Console.ReadLine();
    if (readResult != null)
    {
        roleName = readResult.Trim();
    }

    if (roleName.ToLower() == "administrator" || roleName.ToLower() == "manager" ||
roleName.ToLower() == "user")
    {
```

```
        validEntry = true;
    }
    else
    {
        Console.Write($"The role name that you entered, \"{roleName}\" is not valid.
");
    }

} while (validEntry == false);

Console.WriteLine($"Your input value ({roleName}) has been accepted.");
readResult = Console.ReadLine();
```

Project 3 code

The following code is one possible solution for challenge project 3 from the previous unit.

The code uses a `for` statement for the outer loop because you cannot modify the value assigned to a 'foreach iteration variable'. You could work around this by declaring an additional string variable inside the `foreach` loop, but then you would be adding unwanted complexity to your code logic. In other words, using the iteration statement `foreach (string myString in myStrings)` and then attempting to process the `myString` variable will generate a compilation error.

The code uses a `while` statement for the inner loop because, depending on the value of the data string, the code block may not be executed (when the string does not contain a period). You should not use a `do` statement in situations where the iteration block may not need to be executed.

C#

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

string myString = "";
int periodLocation = 0;

for (int i = 0; i < stringsCount; i++)
{
    myString = myStrings[i];
    periodLocation = myString.IndexOf(".");

    string mySentence;
```

```
    // extract sentences from each string and display them one at a time
    while (periodLocation != -1)
    {

        // first sentence is the string value to the left of the period location
        mySentence = myString.Remove(periodLocation);

        // the remainder of myString is the string value to the right of the location
        myString = myString.Substring(periodLocation + 1);

        // remove any leading white-space from myString
        myString = myString.TrimStart();

        // update the comma location and increment the counter
        periodLocation = myString.IndexOf(".");

        Console.WriteLine(mySentence);
    }

    mySentence = myString.Trim();
    Console.WriteLine(mySentence);
}
```

Next unit: Knowledge check

Continue >