Programming with C#

# Looping Structures Practice

1. Open Visual Studio if it is not already open.
2. Locate the folder where you extracted the companion content.
3. In the Chapter 5 folder, locate either the Windows 7 or Windows 8 folder.
4. Open the WhileStatement project.
5. On the Debug menu, click Start Debugging or press F5.
6. Visual Studio will build and run the application. The UI displays an Open File button and two text boxes. The application is a simple viewer for ASCII text files.
7. Click the Open File button.
8. The File Open dialog appears.
9. Navigate to the folder that contains the current project and select the MainWindow.xaml.cs file and click Open.
10. The application will display the file name in the smaller text box but it does not open the file yet.
11. Close the application or in Visual Studio, on the Debug menu click Stop Debugging.
12. Expand the MainWindows.xaml branch in Solution Explorer and then double-click the MainWindow.xaml.cs file to open the file in the Text Editor window.
13. Locate the private async void openFileClick(object sender, RoutedEventArgs e)method.
14. This method includes code that you may not be familiar with. It is responsible for displaying the file open dialog, checking that a file has been selecting, establishing a stream in memory to hold the contents of the file, and then opening the file into that memory string.
15. The use of the async and await keywords is a method of programming known as asynchronous programming and is intended to allow a UI to remain responsive while an operation is pending or ongoing in the background. In this case, opening the file.
16. You will notice the last line in this method is a call to the displayData() method, passing in a TextReader object with the contents of the file in the memory stream. The file does not display in the text box yet. You will enter code to perform this operation using a while statement.
17. Replace the // TODO comment in the displayData() method with your own code. You will need to:
    1. Set the text content of the source.Text to an empty string “”
    2. Create a string variable called **line** to hold each line of the file that will be read and assign to it the value returned from the reader. Use **reader.ReadLine()** to get a single line at a time. This will read the first line of the file
    3. Now create a while loop that checks for null in the **line** variable, appends the text of the line to the source.Text value, concatenates an newline character, then assigns the next line of the file to line again. You code should look like this:  
        **source.Text = "";  
       string line = reader.ReadLine();  
       while (line != null)  
       {  
        source.Text += line + '\n';   
        line = reader.ReadLine();  
       }**
18. If you are using Windows 8.1, type the following statement after the closing brace at the end of the *while* loop:

**reader.Dispose();**

1. If you are using Windows 7 or Windows 8, type the following statement:

**reader.Close();**

1. These statements release the resources associated with the file and close it. This is good practice because it makes it possible other applications to use the file and also frees up any memory and other resources used to access the file.
2. On the Debug menu, click Start Debugging.
3. When the form appears, click Open File.
4. Navigate to the Chapter 5, WhileStatement folder again and open the MainWindow.xaml.cs file
5. Fix any errors that may occur.
6. If all goes as planned you should see the contents of the file in the lower text box.
7. Close the application.
8. Select the File menu in Visual Studio.
9. Select Close Solution.

## The do Statement

1. Open the DoStatement project from the Chapter 5 folder
2. In Solution Explorer, double-click MainWindow.xaml to open the UI of the application.
3. This application asks the user to input a number. When you click the Show Steps button, the application is intended to show the steps used to convert the number to its octal version.
4. In Solution Explorer, expand MainWindow.xaml and then double-click MainWindow.xaml.cs to open the code file.
5. Locate the private void showStepsClick(object sender, RoutedEventArgs e) and note that it currently contains no implementation code. You will complete this method.
6. Enter the following code to setup the variables to be used in this method.  
     
   int amount = int.Parse(number.Text); // Get the value from the text box and cast to an integer  
   steps.Text = “”; // Set the steps textbox to a blank string to clear its contents  
   string current = “”; // Create a string variable and set it to an empty string
7. The do loop will perform the conversion on the value. Enter this code for your do loop.  
     
   do  
   {  
    int nextDigit = amount % 8;   
    amount /= 8;   
    int digitCode = '0' + nextDigit;   
    char digit = Convert.ToChar(digitCode);   
    current = digit + current;   
    steps.Text += current + "\n";   
   } while (amount != 0);  
     
   Note: Some programmers prefer to place the while portion of the do loop immediately after the closing curly brace but you can also place it on its own line.
8. The algorithm used repeatedly performs integer arithmetic to divide the *amount* variable by 8 and determine the remainder. The remainder after each successive division constitutes the next digit in the string being built. Eventually, when *amount* is reduced to 0, the loop finishes. Notice that the body must run at least once. This behavior is exactly what is required because even the number 0 has one octal digit.
9. Run the application and test the results.
10. If all goes well, stop the application. Otherwise fix any errors in your code and run again.
11. When finished, you may close Visual Studio.