Lab 2 – Hello World! An introduction to C#

# Description

This lab is designed to give you a basic understanding of a C# application and provide an opportunity to use many simple C# techniques.

# Estimated Time

This lab will take an estimated 3 hours to complete.

# Deliverable

Push your code to Github and submit the Github link to Brightspace.

Make sure that you add the bin and obj directories to gitignore.

# Step 1: Preview the demo application provided.

1. Demo the application for yourself, you can model your functionality after this application. The demo is provided as a zip file named ‘Lab 1 Demo.zip’

# Step 2: Create a console application.

Create a windows console application that provides the following functionality.

1. Create a Console application.
   1. Open Visual Studio
   2. Create a new project.
   3. In the new project window select the following filters, C#, Windows, Console, for language, platform, and project type, respectively.
   4. Select Console App template.
   5. Click next.
   6. Name the project ‘Lab2.’
   7. Click next.
   8. Select .NET 8.0 as the framework.
   9. Check the “Do not use top-level statements” checkbox.
   10. Click ‘Create’
2. Create a typical application that accepts user input from the keyboard and processes the result.
   1. Create a console menu with an option for each of the items below.
      1. (1) Import Words from File
         1. Create a method that takes the words from a text file and stores them in a List.
            1. The namespace that provides this functionality is in System.IO
            2. The easiest way to read a text file is to use the class System.IO.StreamReader, the constructor of which takes a path to the file you want to read.
            3. For each word you find in the file add it to a List or array object. Use a generic list of type IList<string>. Coded as such: **IList<string> words = new List<string>();** Lists are a very powerful tool in .NET and provide dozens of extension methods that will help with this lab.
         2. Once the method runs, display the number of words you read from the file.
         3. Test your method using the file ‘Words.txt’ provided in the example.
      2. (2) Bubble Sort words (alphabetically ascending)
         1. Create a method that accepts a list or array of strings and provides a bubble sort on the collection.
            1. Your methods signature must look like this: **IList<string> BubbleSort(IList<string> words)**
         2. Use your extensive knowledge of Data Structures (google) and write code to perform a bubble sort on the list.
         3. Once your list is sorted, display how long it took to sort the list of words.
      3. (3) LINQ/Lambda sort words (alphabetically ascending)
         1. Create a method that accepts a list or array of strings and provides a LINQ sort on the collection.
            1. Your methods signature must look like this: **IList<string> LINQSort(IList<string> words)**
         2. Use a LINQ query or a Lambda expression to sort your list of strings.
         3. Once your list is sorted, display how long it took to sort the list of words. (You should note that it will be significantly faster to sort the list using the built-in technologies, more on that later.)
      4. (4) Count the Distinct Words
         1. Use a LINQ query or Lambda expression to count all the distinct words.
         2. Display the result.
      5. (5) Take the last 10 words.
         1. Use a LINQ query or a Lambda expression to achieve that.
         2. Display these words.
      6. (6) Reverse print the words - Print the words from end to the beginning. In other words, reverse print the list.
         1. Use a LINQ query or Lambda expression to reverse the list without modifying the original list.
         2. Display the result.
      7. (7) Get and display the words that end with ‘d’ and display the count.
         1. Use a LINQ query or Lambda expression to achieve that.
         2. Display the words and the number of words found.
      8. (8) Get and display the words that start with letter ‘q’ and display the count.
         1. Use a LINQ query or Lambda expression to achieve that.
         2. Display the words and the number of words found.
      9. (9) Get and display the words that are more than 3 characters long and contain the letter ‘a’ and display the count.
         1. Use a LINQ query or Lambda expression to achieve that.
         2. Display the words and the number of words found.
      10. (X) Exit
          1. Close the application.
   2. Write your methods in a way so none of your methods create side effects, meaning that, for instance, sorting will not impact the original list read from file.
   3. You must handle exceptions that might occur in your code.

# Notes/Useful Classes/Namespaces

* System.Console
  + Read and Write information from the console using a keyboard.
* System.IO
  + Read information from a text file.
* Make sure that the line ‘using System.Linq;’ is at the top of your class. The namespace is needed to access perform LINQ queries and Lambda expressions.