

Assignment 2: Iteration and Selection Algorithms

Loops, if-else, switch

1. Objectives

The main objectives of this assignment are:

- To practice with conditional and iteration algorithms.
- To learn how to use strings and string formatting.
- To work with methods that have one or more parameters and a return type.

In this assignment, we will be working with **if-else** and **switch** statements, loops with **for**, **while** and **do-while** statements as string.

In this assignment, we will be working with **if-else** and **switch** statements, as well as loops including **for**, **while**, and **do-while**. Additionally, focus will be placed on using string formatting.

An important part of this assignment involves writing numerous methods. It is important to adhere to a key principle: Create a method for every distinct task that you want the objects of a class to perform. Methods can be categorized as either void or returning a value.

- **Void methods:** A **void** method performs operations when invoked but does not return any value to the caller.
- **Returning value methods:** These function similarly but with one key distinction: they must always return a value to the caller through a **return** statement.

Both types may have parameters if they require information not readily available within their scope

2. Description

This assignment has four major parts. The first two are required for grades D and C, and the last two are required for a B and A, respectively. The following is a summary of the parts, but detailed description is provided in the subsequent sections.

Assignment 2A (Grade D): TemperatureConverter

In this part, your task is to write a class that displays a list of temperatures. Specifically, it should convert temperatures between Celsius and Fahrenheit, and vice versa.,

Assignment 2B (Grade C): StringFunctions

Create a class that reads a text input from the user via the Console window. This class should calculate and display the length of the text (number of characters) in the Console Window. Additionally, it should convert and display the original text in uppercase.

Assignment 2C (Grade B): MathWork:

Create a class for performing simple mathematical calculations. The class should contain methods that utilize nested loops.

Assignment2D (Grade A): Scheduler

Write a class that provides a working schedule for someone working according to a schedule based on a pattern determined by week numbers.

General Instructions for All Parts:

- Structure your classes using methods, with each method performing a specific task.
- Comment your methods using triple slashes `///`. To do this, place the cursor on the line just above the method definition and type the slashes. Visual Studio (VS) will generate a template for the comments, which you should then complete.

3. Assignment 2A (Grade D): TemperatureConverter.cs

In this part, we will develop a program that displays a list of temperatures in Celsius degrees converted to Fahrenheit and vice versa. The program includes a menu that allows the user to select between these two conversion options. This menu will repeat until the user selects '0' to exit the program.

- When option 1 is chosen, the program calculates and displays a list of temperatures from 0 to 212 degrees Fahrenheit, converted to Celsius.
- When option 2 is selected, it calculates and lists temperatures from 0 to 100 degrees Celsius, converted to Fahrenheit.
- When option 0 is selected. The loop should end.

The conversion formulas to be used are:

$$F = 9/5 * C + 32$$

$$C = 5/9 * (F - 32)$$

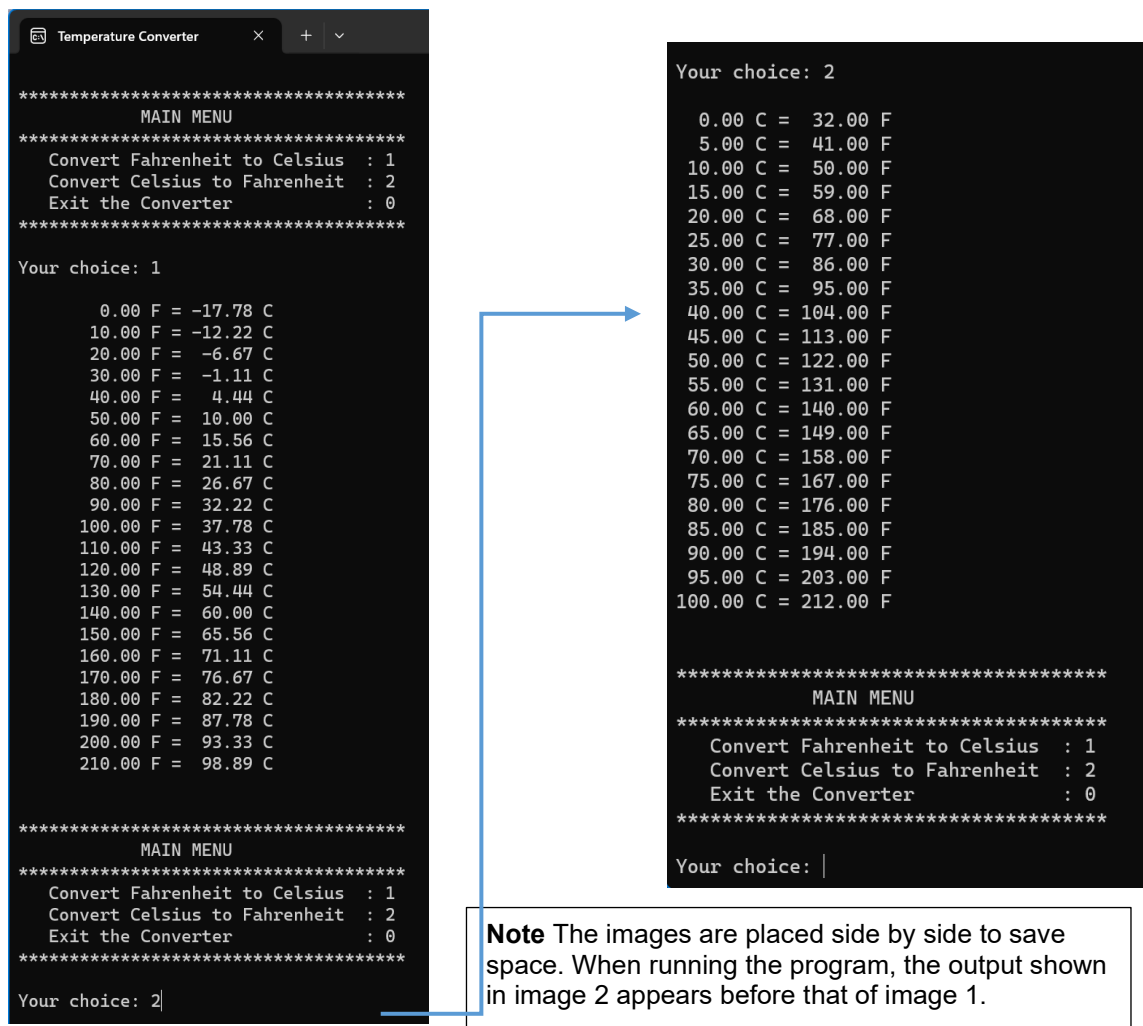
where: **F** = **Fahrenheit**, **C** = **Celsius**

In the following images, which are screenshots taken during a sample run-time, the results are calculated at intervals of 10 for option 1 and 5 for option 2. This is to keep the list concise in the Console window. You may use other intervals.

The Main Menu is programmed to repeat after each operation and will only stop when the user enters '0' as choice. When the user inputs a '0', the Main Menu loop ends, and the program execution returns to the point where the object was initiated and called, which is the Main method.

Requirements for Grade D

- 3.1. The application must include at least two classes: one containing the Main method and another for the calculation of temperatures (**TemperatureConverter**).
- 3.2. The **Main** method should be kept short and concise. Its primary function is to instantiate the **TemperatureConverter** class and invoke its Start method, as in the previous assignment.



- 3.3. All class fields (if used) should be declared as **private**.
- 3.4. User input must be carefully controlled to ensure that the provided values are as expected. For instance, the menu choice should be restricted to an integer value of 0, 1, or 2, and nothing else.
- 3.5. Use appropriate names for variables, methods, and other identifiers, as well as for your solution and project.

If you need help, a step-by-step guidance and some examples are available in Canvas.

Optional part (but recommended):

Display the results from the previous part in multiple columns, as shown in the picture below, where the results are organized into three columns. Tips and instructions for accomplishing this task are included with the assignment description.

```

Temperature Converter  X + v

*****
MAIN MENU
*****
Convert Fahrenheit to Celsius : 1
Convert Celsius to Fahrenheit : 2
Exit the Converter           : 0
*****

Your choice: 2

      0.00 C = 32.00 F      5.00 C = 41.00 F      10.00 C = 50.00 F
    15.00 C = 59.00 F    20.00 C = 68.00 F    25.00 C = 77.00 F
    30.00 C = 86.00 F    35.00 C = 95.00 F    40.00 C = 104.00 F
    45.00 C = 113.00 F   50.00 C = 122.00 F   55.00 C = 131.00 F
    60.00 C = 140.00 F   65.00 C = 149.00 F   70.00 C = 158.00 F
    75.00 C = 167.00 F   80.00 C = 176.00 F   85.00 C = 185.00 F
    90.00 C = 194.00 F   95.00 C = 203.00 F   100.00 C = 212.00 F
  
```

The program class:

When you are done with the above class, navigate to the Program.cs file and create an instance of your class in the Main method. Call the a starting method of the instance to run the program.

```

class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.Clear();
        Console.Title = "Assignment 2 <By: First and last names>";

        // Instantiate and start Menu object
        // Grade D
        TemperatureConverter converter = new TemperatureConverter();
        converter.Start();

        //continue with other classes
    }

    0 references
    private static void ContinueToNextPart(string title)
    {
        Console.WriteLine("\nPRESS ENTER TO CONTINUE TO THE NEXT PART");
        Console.ReadLine();
        Console.Clear();
        Console.Title = title;
        Console.WriteLine("+++++++ " + title + " ++++++++");
    }
}
  
```

Question: why static?

4. Assignment 2B (Grade C): StringFunctions.cs

Complete the following tasks in addition to the requirements for grade D. In this section, you will practice using a `switch` statement. Develop a class and complete it as follows:

4.1. Create a new class, assign it a suitable name, and save it in a file, such as **StringFunctions.cs**. Implement the following methods within this class.

4.2. **StringLength()**: This method should prompt the user to enter any text. Once the text is entered, the method should calculate and display the length of the string using the following code:

```
int length = text.Length;
```

4.3. Display the length (character count) of the text and the original string converted to uppercase as the output (see the next image).

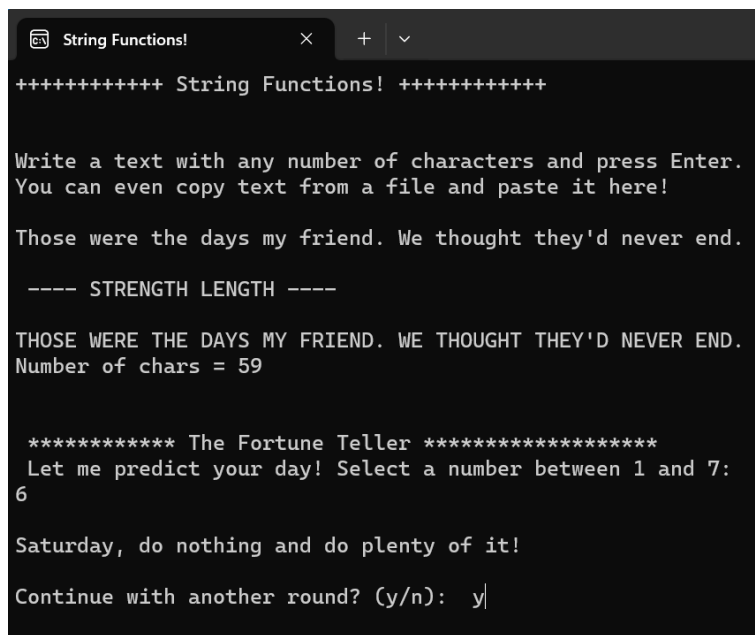
4.4. **PredictMyDay()**: This method should prompt the user to enter a number between 1 and 7, and then display a humorous or interesting comment corresponding to the day of the week (1 = Monday, 7 = Sunday). If the user inputs a number outside of this range (not between 1 and 7), the method should provide a suitable error message to the user. You may use the following phrases as comments:

```
"Keep calm on Mondays! You can fall apart!"
"Tuesdays and Wednesdays break your heart!"
"Thursday is your lucky day, don't wait for Friday!"
"Friday, you are in love!"
"Saturday, do nothing and do plenty of it!"
"And Sunday always comes too soon!"
"No day? A good day but it doesn't exist!"
```

Requirement: use a `switch` statement in this part, to determine which day corresponds to the user's chosen choice.

4.5. **RunAgain()**: The above steps should repeat as long as the user desires. Write a method that asks the user whether to continue or not.

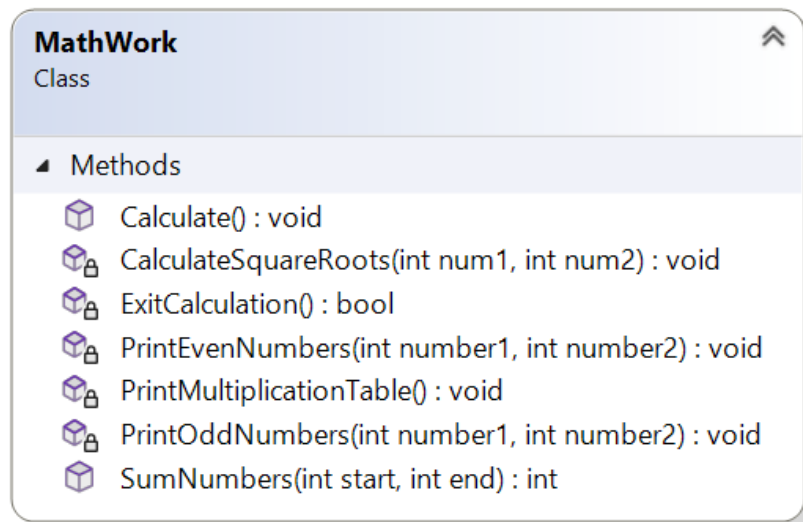
The image illustrates a sample run-time session:



```
String Functions!
+++++ String Functions! +++++
Write a text with any number of characters and press Enter.
You can even copy text from a file and paste it here!
Those were the days my friend. We thought they'd never end.
---- STRENGTH LENGTH ----
THOSE WERE THE DAYS MY FRIEND. WE THOUGHT THEY'D NEVER END.
Number of chars = 59
***** The Fortune Teller *****
Let me predict your day! Select a number between 1 and 7:
6
Saturday, do nothing and do plenty of it!
Continue with another round? (y/n): y
```

5. Assignment 2C (Grade B): MathWork.cs

In this class, we will develop the methods outlined in the class diagram presented below. The primary objective of creating this class is to practice working with numbers and nested loops. It is important to note, as indicated in the diagram, that this class does not contain any instance variables.



5.1. **Calculate ()**: Execute this method in a loop until the user decides to exit. The method should first prompt the user to provide a starting and an ending integer number. Once these numbers are provided, the method should proceed with the following actions:

- Call a method (**SumNumbers**) that calculates the sum of all numbers within the given interval, including the starting and the ending number, and returns this sum. Display the returned value.
- Call a method (**PrintEvenNumbers**) that outputs the even numbers within the range.
- Call a method (**PrintOddNumbers**) that outputs the odd numbers within the range.
- Call a method (**CalculateSquareRoot**) that, using a nested loop, calculates the square root of numbers within the range. In each iteration, it should calculate from the counter's current value to the end number. For example, if the start number is 5 and the end number is 8, it should display rows with the first row showing results for numbers 5, 6, 7, 8 and the second row showing values for 6, 7, 8, and so on. Refer to the sample run-session below for an example. Design the algorithm by yourself.

5.2. **ExitCalculation()**: This method is to be used to control the loop. It is similar to **RunAgain** method in the previous section, but in this case, the method is intended to inquire whether the user intends to terminate the calculation or not.

The image below shows a run-time session:

```

Math Works!
+++++ Math Works! +++++

Sum numbers between any two numbers
Give number1 number: 12
Give end number: 5

The sum of numbers between 5 and 12 is: 68

****Even numbers between 5 och 12
  6   8   10  12

**** Odd numbers between 5 och 12
  5   7   9   11

          ***** Square Roots *****
Sqrt( 5 to 12) 2.24 2.45 2.65 2.83 3.00 3.16 3.32 3.46
Sqrt( 6 to 12) 2.45 2.65 2.83 3.00 3.16 3.32 3.46
Sqrt( 7 to 12) 2.65 2.83 3.00 3.16 3.32 3.46
Sqrt( 8 to 12) 2.83 3.00 3.16 3.32 3.46
Sqrt( 9 to 12) 3.00 3.16 3.32 3.46
Sqrt(10 to 12) 3.16 3.32 3.46
Sqrt(11 to 12) 3.32 3.46
Sqrt(12 to 12) 3.46

Exit Math Work? (y/n)

```

6. Assignment 2D (Grade A) – Scheduler.cs

In addition to the requirements specified in all sub-assignments so far, you have to solve the following problem. In this part, you are provided with an opportunity analyze the problem and decide what kind of a loop to implement.

- 6.1. Write a class **Scheduler** to program a work schedule for an employee who has to work night shifts and weekends. The user of this option is an employee who must work every other (**2nd**) weekend, starting from week number **2**. Additionally, the employee is required to work a night shift every **4th** week, starting from week **1**. The schedule should cover a one-year period, assuming that the last week of the year is numbered 52.
- 6.2. The program should present a list of the weeks during which the user is required to work on weekends and the weeks when the user should work night shifts. To interact with the user, you may, follow the scenario below:
 - Present a menu to the user, displaying the options (Weekends or Nights).
 - Display a list of weeks for which they are scheduled for the selected option.
 - Repeat these steps until the user opts to exit.

```

The Scheduler!
+++++++ The Scheduler! ++++++
-----
                YOUR WORK SCHEDULE
-----

1 Show a list of the weekends to work.
2 Show a list of the nights to work.
0 Exit

Your choice: 1
Week 2      Week 4      Week 6      Week 8
Week 10     Week 12     Week 14     Week 16
Week 18     Week 20     Week 22     Week 24
Week 26     Week 28     Week 30     Week 32
Week 34     Week 36     Week 38     Week 40
Week 42     Week 44     Week 46     Week 48
Week 50     Week 52

-----

1 Show a list of the weekends to work.
2 Show a list of the nights to work.
0 Exit

Your choice: 2
Week 1      Week 5      Week 9      Week 13
Week 17     Week 21     Week 25     Week 29
Week 33     Week 37     Week 41     Week 45
Week 49

-----

1 Show a list of the weekends to work.
2 Show a list of the nights to work.
0 Exit

Your choice: |

```

Hint: It is possible to write one method that can work for both options, if you think of startWeek, endWeek and interval as variables, when running a loop.

Optional: You may program so the user provides the start week and the interval for each option.

7. Submission

The project should compile without errors and the program should work satisfactorily. Compile, run, and test your project before submission.

Submit your assignment in the same way as the previous one.