SOFTWARE DEVELOPMENT

Assignment 2

ZEYNEB ARIF
1813730
UNIVERSITY OF WALES TRINITY SAINT DAVID
Cloud Computing

Table of Contents

Part A....2
Arrays....2
Part B....3
Part C....4
Task 1....4
Task 2....6
Task 3....7
References....8

Figure 1 Array Declaration....2
Figure 2 Array Declaration & Initialization....2
Figure 3 Array Declaration using var....2
Figure 4 Input Issue....3
Figure 5 Task 1 Code....4
Figure 6 Task 1 Screenshot....5
Figure 7 Task 2 Code....6
Figure 8 Task 2 Screenshot....6
Figure 9 Read Number Fix....7
Figure 10 Default Case....7
Figure 11 Is Quitting Fix....7
Figure 12 Case 4 Fix....7

Part A

Arrays

A variable is used to store the literal value, while multiple literal values are stored in an array. An array is the data structure which holds a fixed number of literal values (elements) of the same type of data. Collection components are stored in the memory contiguously. In C #, there may be an array of three types: single, multidimensional, and jagged array. You will find out more about the single-dimensional array here. (TutorialsTeacher, 2020)

Example: Array Declaration

```
int[] evenNums;  // integer array
string[] cities; // string array
```

Figure 1 Array Declaration

Example: Array Declaration & Initialization

```
int[] evenNums = new int[5] { 2, 4, 6, 8, 10 };
string[] cities = new string[3] { "Mumbai", "London", "New York" };
```

Figure 2 Array Declaration & Initialization

Even Nums list can hold up to 5 integers above. The number 5 in the current int[5] square brackets determines the size of an array. Likewise, the number of cities is three in size. Set elements are inserted within curly braces in a comma-separated array}. Variables of the form of arrays can be declared without square brackets using Var. (TutorialsTeacher, 2020)

Example: Array Declaration using var

```
var evenNums = new int[] { 2, 4, 6, 8, 10 };
var cities = new string[] { "Mumbai", "London", "New York" };
```

Figure 3 Array Declaration using var

Part B

There are 3 bugs with the program:

1. When trying to enter anything except for an integer the program is not displaying any error message but instead crashes completely.
2. When trying to quit the program the only valid confirmation is “YES”.
3. When choosing the option to display the entered numbers the user is instead asked for a confirmation to quit the program.

```
*****************************************************************************
* This application is designed to allow you to choose numbers *
* and finds the biggest and the smallest value *
*****************************************************************************



                    Select an option
                    1. Enter the numbers
                    2. Find the smallest
                    3. Find the largest
                    4. Display all numbers
                    5. Quit
                    Please choose an option: asd
Unhandled exception. System.FormatException: Input string was not in a correct format.
```

Figure 4 Input Issue

Part C

Task 1

```
using System;

namespace Assignment_2_Exercise_1
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            int player1;
            int player2;
            int p1Score = 0;
            int p2Score = 0;

            for (int i = 0; i < 10; i++)
            {
            Player1:
                Console.Write("Goals Player 1: ");
                if (!int.TryParse(Console.ReadLine(), out player1))
                {
                    Console.WriteLine("Invalid Input. Please try again.");
                    goto Player1;
                }
            Player2:
                Console.Write("Goals Player 2: ");
                if (!int.TryParse(Console.ReadLine(), out player2))
                {
                    Console.WriteLine("Invalid Input. Please try again.");
                    goto Player2;
                }

                if (player1 > player2)
                {
                    p1Score += 2;
                    Console.WriteLine("Player 1 Wins the Round.");
                }
                else if (player2 > player1)
                {
                    p2Score += 2;
                    Console.WriteLine("Player 2 Wins the Round.");
                }
                else
                {
                    p1Score++;
                    p2Score++;
                }
            }
            Console.WriteLine("Player 1 score is {0}\nPlayer 2 score is {1}", p1Score, p2Score);
            if (p1Score > p2Score)
            {
                Console.WriteLine("Player 1 wins with {0} points!", p1Score);
            }
            else if (p2Score > p1Score)
            {
                Console.WriteLine("Player 2 wins with {0} points!", p2Score);
            }
            else
            {
                Console.WriteLine("It's a Draw!");
            }
        }
    }
}
```

Figure 5 Task 1 Code

We need 4 variables for this program, 2 for each player, 1 for to keep track of the score, 1 to use for comparing the goals and determine the winner of match. TryParse + Labels combo to ensure the user input is in the correct data type.

```
Goals Player 1: asd
Invalid Input. Please try again.
Goals Player 1: 1
Goals Player 2: asd
Invalid Input. Please try again.
Goals Player 2: 2
Player 2 Wins the Round.
Goals Player 1: 3
Goals Player 2: 1
Player 1 Wins the Round.
Goals Player 1: 2
Goals Player 2: 3
Player 2 Wins the Round.
Goals Player 1: 1
Goals Player 2: 3
Player 2 Wins the Round.
Goals Player 1: 2
Goals Player 2: 1
Player 1 Wins the Round.
Goals Player 1: 3
Goals Player 2: 2
Player 1 Wins the Round.
Goals Player 1: 1
Goals Player 2: 23
Player 2 Wins the Round.
Goals Player 1: 13
Goals Player 2: 13
Goals Player 1: 23
Goals Player 2: 1
Player 1 Wins the Round.
Goals Player 1: 2
Goals Player 2: 3
Player 2 Wins the Round.
Player 1 score is 9
Player 2 score is 11
Player 2 wins with 11 points!
```

Figure 6 Task 1 Screenshot

Task 2

```
using System;
using System.Collections.Generic;
namespace Assignment_2_Exercise_2
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            var input = new List<int>();
            var result = new List<int>();
            int number;
            while (true)
            {
                Console.Write("Enter a number or press enter to continue: ");
                if (int.TryParse(Console.ReadLine(), out number))
                {
                    input.Add(number);
                }
                else
                {
                    break;
                }
            }
            if (input.Count % 2 == 1)
            {
                input.Add(1);
            }
            for (int i = 0; i < input.Count; i += 2)
            {
                result.Add(input[i] * input[i + 1]);
            }
            Console.WriteLine("The new list is: " + string.Join(", ", result));
        }
    }
}
```

Figure 7 Task 2 Code

Here we need 2 lists, 1 to fill with the user input and the second to fill with the result. If the first list has odd number of elements in it we append 1 because it's not going to affect the final result and makes the code simpler to write. The program displays the result when the user enters an invalid or empty input.

```
Enter a number or press enter to continue: 1
Enter a number or press enter to continue: 2
Enter a number or press enter to continue: 3
Enter a number or press enter to continue: 4
Enter a number or press enter to continue: 5
Enter a number or press enter to continue: 6
Enter a number or press enter to continue: 7
Enter a number or press enter to continue: 8
Enter a number or press enter to continue: 9
Enter a number or press enter to continue:
The new list is: 2, 12, 30, 56, 9
```

Figure 8 Task 2 Screenshot

Task 3

As stated in Part B the application has 3 bugs and here's the solution to all of them:

```
static int ReadNumber(string prompt)
{
    string text;
    int number;
    WriteText(prompt, 20, 14);
    text = Console.ReadLine();
    //Check with TryParse
    if (!int.TryParse(text, out number))
    {
        number = 0;
    }
    ClearText(20, 14, prompt.Length + text.Length);
    return number;
}
```

Figure 9 Read Number Fix

For the crashing when invalid input is entered, we can use a TryParse, if it fails the input is set to 0 which sends the algorithm to the default case of the switch and the default case is designed to handle invalid inputs.

```
default:
    WriteText("Unknown option value entered", 20, 15);
    break;
```

Figure 10 Default Case

For the quitting confirmation we can convert the user input to lowercase and check for "yes"

```
static bool IsQuitting()
{
    string response;
    bool quit = false;
    WriteText("Do you really wish to quit? ", 20, 13);
    //Conver to lowercase
    response = Console.ReadLine().ToLower();
    if (response == "yes")
    {
        quit = true;
    }
    return quit;
}
```

Figure 11 Is Quitting Fix

And finally, for displaying the numbers we can add the following functionality

```
case 4:
    //Print with WriteText
    WriteText(string.Join(", ", numbers), 20, 15);
    Console.ReadKey();
    break;
```

Figure 12 Case 4 Fix

References

TutorialsTeacher, 2020. *Array CSharp.* [Online]
Available at: https://www.tutorialsteacher.com/csharp/array-csharp