Contents

First Arrays Manipulations

This lab serves multiple goals:

- To introduce you to arrays of different datatypes,
- To introduce you to the different ways of declaring, assigning and initializing arrays,
- To iterate over arrays,
- To use the `Length` property of array,

Declaration, Assignment & Initialization of an Array

Warm-up

Write a program that implements the following steps:

1. declares an array `myArray` of `int` of size 5,
2. initializes `myArray` with the values $1, 2, 3, 4$ and 5,
3. displays the content of `myArray` on the screen.

Questions

- What values are stored in this array after declaring it *but before initializing it*?
- There are a few different ways you can declare and initialize an array of size 5 holding values $1, 2, 3, 4$ and 5. Can you think of two different ways of doing this?

Answer:

- All the values in the array are set to 0,
- Two possible ways are `int[] myArray = new int[] {1, 2, 3, 4, 5};` and `int[] myArray = {1, 2, 3, 4, 5};`.

Going wrong

Now, let us write *incorrect* statements. For each of the programs below, compile them and make sure you understand the error messages that are displayed.

Trying to set all the values at once after declaring

```
int[] myArrayA = new int[5];
myArrayA = {1, 2, 3, 4, 5};
```

Out of bound error (read)

```
int[] myArrayB = new int[5];
Console.WriteLine(myArrayB[5]);
```

Out of bound error (write)

```
int[] myArrayC = new int[5];
myArrayC[5] = 12;
```

Reading the array as a whole (technically not an error)

```
int[] myArrayD = new int[5];
Console.WriteLine(myArrayD);
```

This last statement is not "incorrect" in the sense that it will not prevent your program from executing, but it is not doing what you could or would have expected.

Second Array Manipulation

Write a program that

1. declares an array `myArray` of `int` of size 10,
2. initializes `myArray` with the values $1, 2, 3, \ldots, 9$ and 10,
3. displays the content of `myArray`.
4. sums the values stored in `myArray` and displays the result.
5. computes the product of the values stored in `myArray` and displays the result.

If you are unsure how to get started, you can use the following code.

Getting started:

```
int[] myArray = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int i = 0;
int sum = 0;
int product = 1;
while(i < myArray.Length){
    // Fill this!
    i++;
}
Console.WriteLine("The sum of the values in the array is " + sum + ".");
```

```
Console.WriteLine("The product of the values in the array is " + product + ".");
```

Exploring Arrays

For this part, create a new array:

1. declare a `char` array of length 6, name it `letters`
2. initialize the first 4 indices of `letters` with the following values: `'a', 'b', 'c', 'd'`
3. initialize *index 5* of `letters` with the value `'f'`

Now, write the following statements:

1. Write a statement to display the last `char` value in `letters` (should display `f`).
2. Write a statement to display the value stored at index 4. What is that value? Why?
3. Write a statement to display the characters in the *first half* of the array (`'a', 'b', 'c'` but no others).

Execute your program to ensure you are seeing the expected output before proceeding.

Next, update the part of the program where `letters` is declared and change the length of `letters` to 8. Do not modify any other parts of the program. Then execute the program again.

Answer the following questions:

1. What is the last `char` of the `letters` array now, after changing its length?
2. Does your program still output *the last* `char` value in `letters` array?
3. When displaying the first half of the array, does your program still display *the first half*? (After changing the length, the first half contains the values `'a', 'b', 'c', 'd'`)
4. If you did not get the last value or the first half you expected, can you think of a way to perform these array operations in a way that can accommodate arrays of different lengths?