

CSCI 1301 – Lab 29

November 12, 2019

1 A Simple Loop

This exercise is to practise `for` loops if you feel the need to.

Write a program that asks the user to enter a positive integer, and then uses a `for` loop to compute the sum of all the integers between 1 and the integer given by the user. For instance, if the user enters 5, your program should display 15 at the screen (i.e., $1 + 2 + 3 + 4 + 5 = 15$).

Then, answer the following questions:

1. Without running your program, can you tell what will happen if the user enter a negative value?
2. Do you think you could have written the same program using a `while` loop?
3. How could you change your program so that it would compute the product instead of the sum (i.e., for 5, $1 \times 2 \times 3 \times 4 \times 5 = 120$)?
4. How could you change your program so that it would display on the screen the divisors of the integer entered (i.e., for 5, only 1 and 5)?

You can modify your program to check your answers to the previous questions. Once you are done, modify your original program with two respects:

1. Once the result of the computation is displayed at the screen, ask the user if (s)he wants to compute the sum using another integer or quit, and act accordingly.
2. Make some input-validation: floating-point values, strings and negative values should not be allowed (i.e., your program should ask for another value).

2 Manipulating Two Arrays at the Same Time

Write a program that

1. declares two array of `int` of size 8,
2. initializes the values of the first array with random numbers between 0 and 9,
3. initializes the values of second array with random numbers between 0 and 9,
4. displays the content of the two arrays, and, for each index, “W” if the value in the first array is greater than the value of the second array, “T” if they are equal, and “L” if it is lesser.

An example of execution of this program would display:

0	8	L
5	3	W
3	3	T
1	2	L
3	1	W
9	0	W
9	0	W
1	5	L

Where the first array contains “0 5 3 1 3 9 9 1” and the second, “8 3 3 2 1 0 0 5”.

3 Pushing Further (Optional)

For this lab, we will introduce only one notion, but a crucial one: the difference between value and reference types. This topic is introduced in your textbook (“Value Types vs. Reference Types”, Chapter 7.17), and will be studied in CSCI 1302. You can have a look at <https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/data-types/value-types-and-reference-types>: this page is not so complex, a short (and excellent) read.

Let us motivate why this notion is so critical with an example:

```
int[] arrayA = { 1, 2, 3, 4, 5 }; // Let me declare a simple array of integer

// I'd like to make a copy of that array. Let me try the following:
int[] arrayCopyWrong = arrayA;

foreach (int i in arrayCopyWrong)
    Console.Write(i + " ");

Console.WriteLine();

// It seems to be working! Except that if we change a value in our copy:
arrayCopyWrong[0] = 6;

// It also changes the value in our original array!
foreach (int i in arrayA)
    Console.Write(i + " ");

Console.WriteLine();
```

What happened is that we copied the reference to the array, and not the array itself. We now have two ways of accessing our array, using `arrayA` or `arrayCopyWrong`, but still only one array.

To perform a copy of the array, we need to do something like the following:

```
int[] arrayB = { 1, 2, 3, 4, 5 };
int[] arrayCopyRight = new int[arrayB.Length];

// We copy each value, one by one:
for(int i = 0 ; i < arrayB.Length; i++)
    arrayCopyRight[i] = arrayB[i];

// If we change a value in our copy:
arrayCopyRight[0] = 6;

// It changes the value only in that copy:
foreach (int i in arrayB)
    Console.Write(i + " ");
```

```
Console.WriteLine();  
  
foreach (int i in arrayCopyRight)  
    Console.Write(i + " "); }
```

Array is actually a class (cf. [https://msdn.microsoft.com/en-us/library/system.array\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.array(v=vs.110).aspx)), and as such provides several methods. For `x` an `int`, `array1` and `array2` two arrays containing the same type of values and of size at least `x`, you can copy the first `x` values of `array1` into `array2` using `Array.Copy(array1, array2, x);`. Try to use this method with the previous example.