

Exercise 2a: The for-statement

1. Objectives

This exercise gives you practice with for-loops. For-loops are used when you know the number of iterations already before the start of the loop.

2. To Do

Create an application that performs the following tasks:

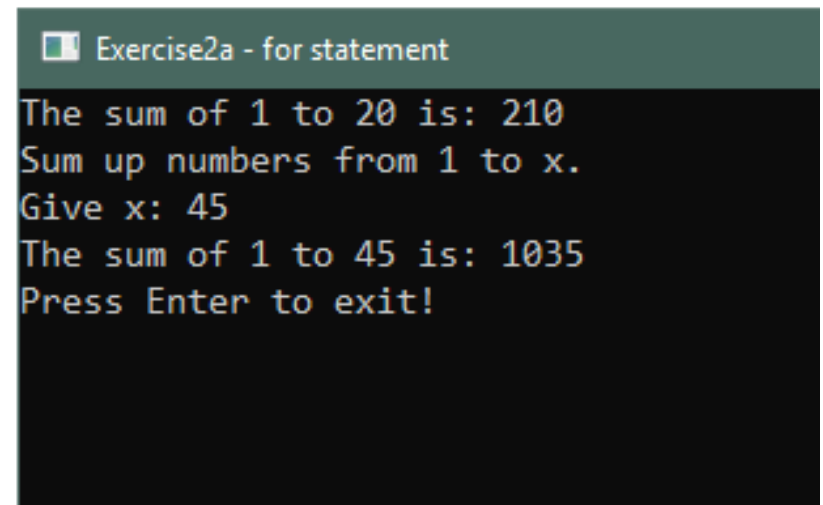
- 2.1. Sums up all numbers from 1 to 20 and displays the sum.
- 2.2. Sums up all numbers from 1 to a bigger number x, where x is given by the user.

A run example is illustrated in the figure.

The solution is given on the next page. You may try to solve the exercise before looking at the solution, but you can of course refer to the given code whenever you feel to make a cross-check or you are unsure how to proceed. ,

Following the Solution-section, there are some recommended improvements, if you would like to work further with this exercise.

If you follow the steps and the code given in the solutions, it is better that you write the code word for word instead of copy and paste. Rewriting makes you think about what you are doing and why.



```
Exercise2a - for statement
The sum of 1 to 20 is: 210
Sum up numbers from 1 to x.
Give x: 45
The sum of 1 to 45 is: 1035
Press Enter to exit!
```

3. Solution:

3.1. Create a new project in VS, save it as Exercise2a. VS creates a start class, Program.cs and prepares the body of the Main-method for you. Leave this class as it is for now and proceed with the next step.

3.2. Create a new class in your project, and name it **ForStatements**. Write the following code in this new class and save the file:

```
public class ForStatements
{
    //Calculate sum of numbers 1 to 20
    public void SumNumbers()
    {
        int sum = 0;
        for (int i = 1; i <= 20; i++)
        {
            sum = sum + i;
        }

        Console.WriteLine("The sum of 1 to 20 is: " + sum);
    }
}
```

3.3. Go back to the Program class and complete the Main method with the code below:

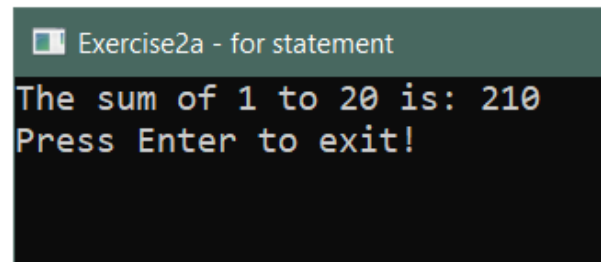
```
public class Program
{
    public static void Main(string[] args)
    {
        Console.Title = "Exercisela - for statement";

        ForStatements obj = new ForStatements();
        obj.SumNumbers();

        Console.WriteLine("Press Enter to exit!");
    }
}
```

```
        Console.ReadLine();  
    }  
}
```

3.4. In case you have written the code correctly, you can run the program. Otherwise, read the compiler message and fix the errors. Sometimes, you may forget to write a semicolon at the end of a statement or miss a closing parenthesis, which result in compilation errors. To run the program, press F5. You should get the output as shown in the figure below:



The application is working well but how useful is program? It only sums values 1 to 20. Let us bring one improvement by letting the upper value be given by the user!

3.5. Go back to ForStatements-class and add the following method:

```
public void SumNumbersFromOneTo()  
{  
    int endNumber;  
    int sum = 0;  
  
    Console.WriteLine("Sum up numbers from 1 to x.");  
    Console.Write("Give x: ");  
    string str = Console.ReadLine();  
    endNumber = int.Parse(str);  
  
    for (int i = 1; i <= endNumber; i++)
```

```

    {
        sum = sum + i;
    }

    Console.WriteLine("The sum of 1 to " + endNumber + " is: " + sum);
} //end method

```

This method should be added inside the class. See that you have a closing curly brace after this method to mark the end of the class' body, and one more for closing the namespace, i.e. there should be two more ending curly braces } } after line "//end method" in above, from the last time

3.6. Open the Program.cs and complete it as follows:

```

public class Program
{
    public static void Main(string[] args)
    {
        Console.Title = "Exercisela - for statement";

        ForStatements obj = new ForStatements();

        obj.SumNumbers();

        obj.SumNumbersFromOneTo();

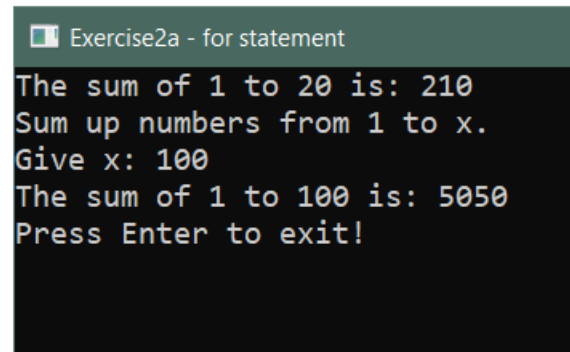
        Console.WriteLine("Press Enter to exit!");
        Console.ReadLine();
    }
}

```

Run the program and you now should see the program running as illustrated in the next figure. What happens in the Main method? Consider the statement:

```
ForStatements obj = new ForStatements();
```

Here, we create an object, an instance, of the class **ForStatements**. We name the object "**obj**" in order to refer to it. The object (obj) is created by the expression at the right of the "=" sign, using the keyword "new". We now have one particular and unique copy of the **ForStatement**. We use this instance by calling its methods, using the dot notation obj.MethodName.



```
Exercise2a - for statement
The sum of 1 to 20 is: 210
Sum up numbers from 1 to x.
Give x: 100
The sum of 1 to 100 is: 5050
Press Enter to exit!
```

Fantastic! By writing a loop with only a few lines of code, we can now add numbers from one to any bigger number, 20, 45, 100, 20000 and so on. Marvelous! But a lot more can be done to make the application even more useful. Here are some suggestions.

4. Further development

- 4.1. The Main method is an entry to the program; it is like a gate to a building area. The class that contains the Main method does not really represent any group of objects. Based on these two arguments, the Main method should not contain code that otherwise belongs to, or is the responsibility of, other objects. In addition, a normal application usually has many classes, 10's, 100's or even 1000's. It is not practical to call methods of so many classes from the Main method.

We bring this change in this little exercise but we will follow the same pattern in all the future programs, whether it is a larger one, a Console application or a Windows Form application. Keep the Main-method short!

- 4.2. Move the call to the methods of the **ForStatements**- class from the Main method to a new method, (you can call it Start), in the aforementioned class: The Start method may now look like this:

```
public class ForStatements
{
    public void Start()
    {
        SumNumbers();
        SumNumbersFromOneTo();
    }
    //rest of the class as before
```

Note that we do not write "obj." here, as the methods are located in this class, in the same instance.

- 4.3. Change the Main method accordingly:

```
public class Program
{
    public static void Main(string[] args)
    {
        Console.Title = "Exercise2a - for statement";

        ForStatements obj = new ForStatements();
        obj.Start(); //The only call to the ForStatement object

        Console.WriteLine("Press Enter to exit!");
        Console.ReadLine();
    }
}
```

Compile and run the application. It should function exactly as before.

- 4.4. The method, Start, which we have just written, is now an entry to the class, ForStatements. It is in fact an interface towards other classes. We can now mark all other methods inside the ForStatements class as **private**, because they are used only internally in that class. This way you protect those methods from being called by outsiders. You now can change the methods without affecting other classes. This is a part of Object Orientation!

The VS project with the above changes are available in the module.

- 4.5. Write a new method in the class ForStatements that sums values from zero to a negative number. Call the method from the Start method. Run and test the method so the output is correct.
- 4.6. Write a method that sums values from any start number to any end number. Both numbers should be given by the user at run time. Check so the start number is less than the end number; if not message the user and do nothing more.

Write these last two methods on your own. Use the forum to ask questions.

Good Luck!