Project for Lesson 2 – Statements

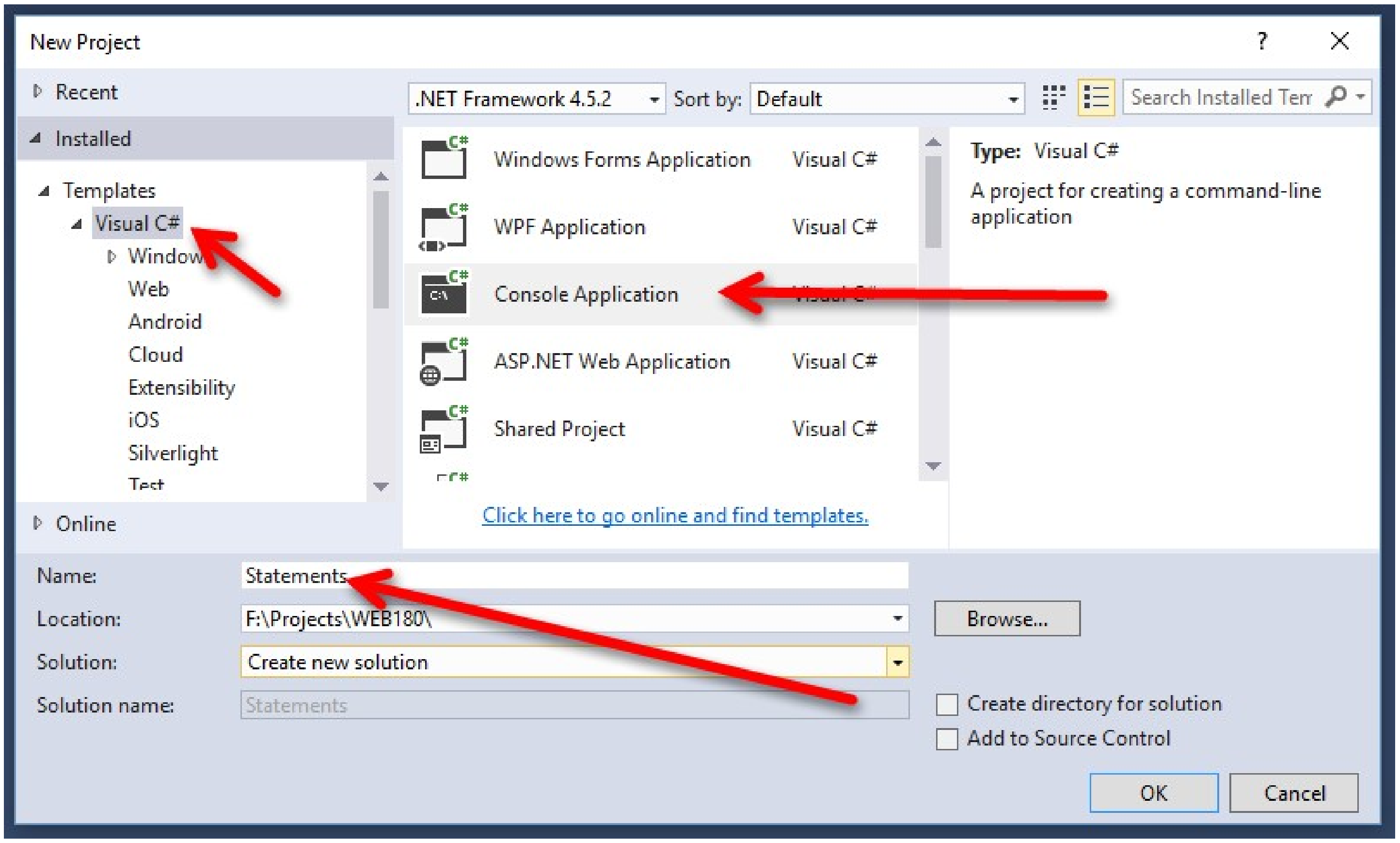
We are going to explore C# statements in this application. After reading through the statements tutorial, https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/statements, follow along with this walk through to create an application that does it all.

First, let's identify the different type of statements that we will cover in this document. Note that I am not going to ask you to code all of the statements presented just the ones that I feel, based on many years of programming, are the most useful.

From the tutorial we are going to look at the following statement types:

- Local declarations, both variable and constant
- If statement
- Switch statement
- While statement
- Do statement
- For statement
- Foreach statement

What follows will be an example of each statement type followed by a task that must include in your statements project. Speaking of which let's get started by creating a new project name Statements.



Type: Visual C#

A project for creating a command-line application

Name: Statements

Location: F:\Projects\WEB180\

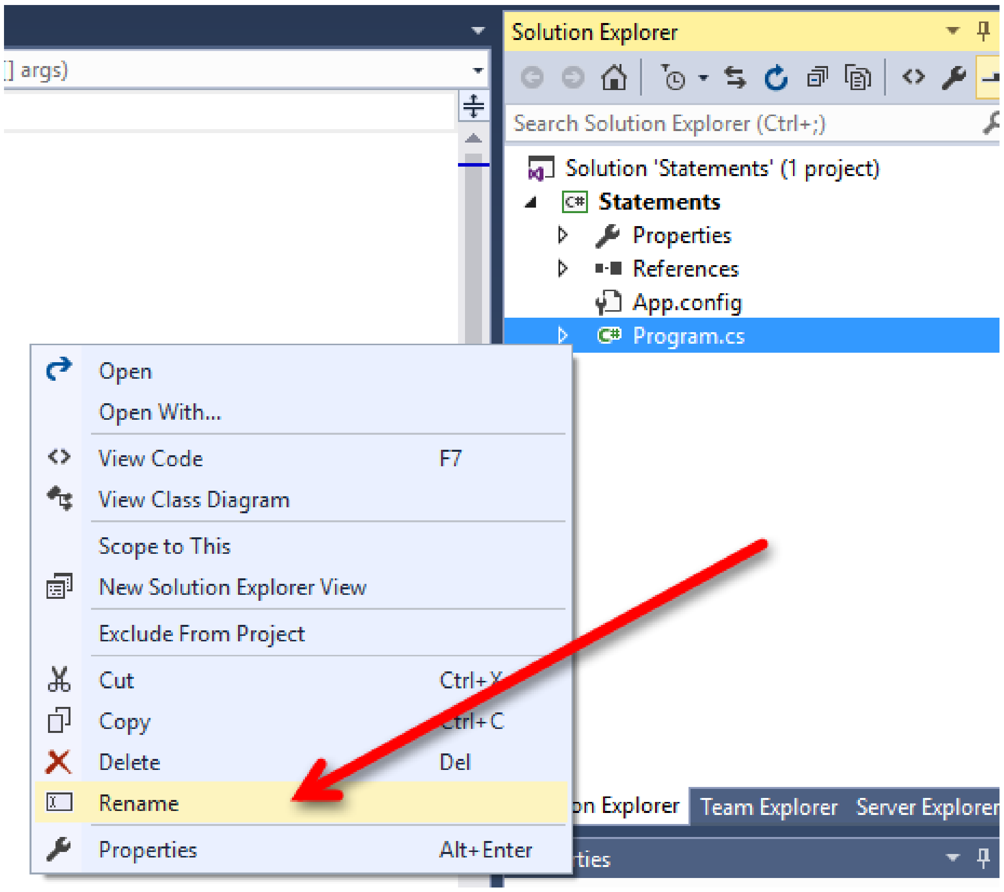
Solution: Create new solution

Solution name: Statements

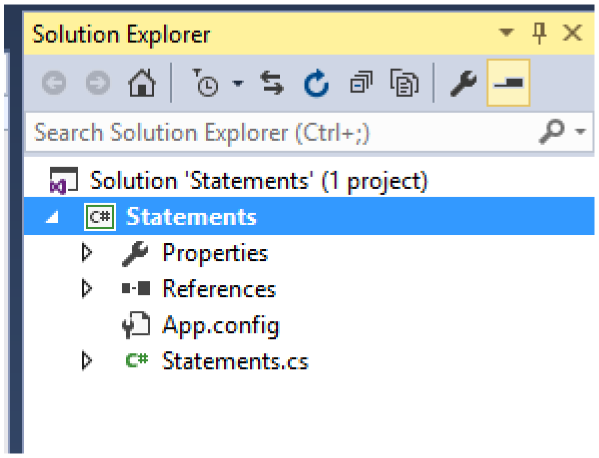
☐ Create directory for solution

☐ Add to Source Control

When the project gets created and opens let's rename the program.cs to statements.cs.



This results in



Statements

Now we can start writing some code (or at least I will be writing) and solving some tasks. We are going to start with local declarations.

From the tutorial we see local declarations look like:

```
    int a;
    int b = 2, c = 3;
    a = 1;
    const float pi = 3.1415927f;
    const  intr = 25;
```

Pretty straightforward and something we have already used in the Lesson 1 coding we did. For your task, add the following to the local declarations above:

- Add an int d with a values of 5
- Add a constant g with value of 9.80665 as a double
    - g, standard gravity, 9 .80665 m/s^2

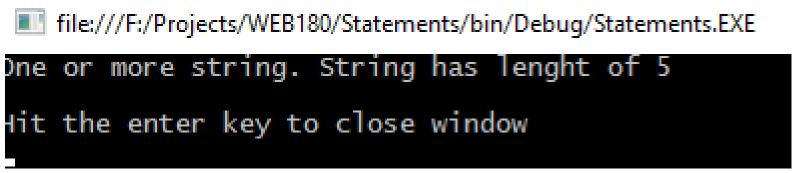
If Statement

One of the most powerful things you can do in programming is making decisions based on data values and events. While we will explore some varied methods of doing this we will start with the if statement.

Consider the tutorial code with the addition of a manual declaration of a string variable.

```
// Basic if statement
string s = "";
if (s.Length == 0)
{
    Console.WriteLine("No arguments");
}
else
{
    Console.WriteLine("One or more arguments");
}
Console.WriteLine();
```

For your task with the if statement, change the string declaration to a string and add the length to the output. It should look like the image below.



file:///F:/Projects/WEB180/Statements/bin/Debug/Statements.EXE

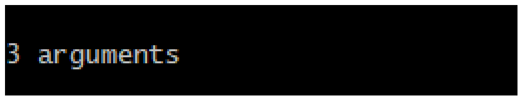
```
One or more string. String has lenght of 5

Hit the enter key to close window
_
```

Switch Statement

An improvement to the basic if statement, at least in some circumstances, is the switch statement. When you must check a variable against multiple values the switch statement is the go to statement. Consider the code below.

```
// switch statement
string[] t = { "one", "two", "three" };
int n = t.Length;
switch (n)
{
    case 0:
        Console.WriteLine("No arguments");
        break;
    case 1:
        Console.WriteLine("One argument");
        break;
    default:
        Console.WriteLine($"{n} arguments");
        break;
}
```

The result of running this code would cause the default statement to be executed. Let's see what happens.



```
3 arguments
```

Notice that each case must be terminated with a break statement even the default case. This causes the switch to only execute one of the cases. Also, the default case gives the switch someplace to go if none of the cases work.

Your task is to modify the code just given so that up to five arguments are checked. Copy and paste can really be your friend when creating simple switch statements. In complex environments, it gets more involved. The longest switch I can remember writing was several hundred lines long with a dozen case statements.

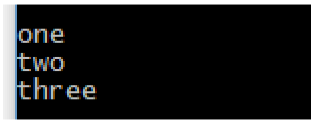
While Statement

Often during the course of solving a computing problem you need to take action while a value is outside of a control value. Consider one of the essential systems required to live in the South—an A/C system. While the temperature of the room is above a set point keep running the A/C. It is somewhat of an over simplification but I hope you get the idea.

Now consider the code in the MS tutorial.

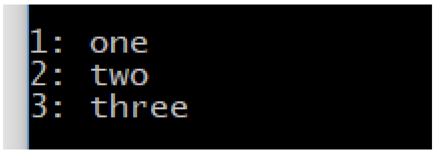
```
    int i = 0;
    while (i < t.Length)
    {
        Console.WriteLine(t[i]);
        i++;
    }
```

The output of this would look like this. See how the while loop controls the output flow.



```
one
two
three
```

Your task for the while statement is to include the number of I at the beginning of each line so that it looks like the following image.



```
1:  one
2:  two
3:  three
```

Do Statement

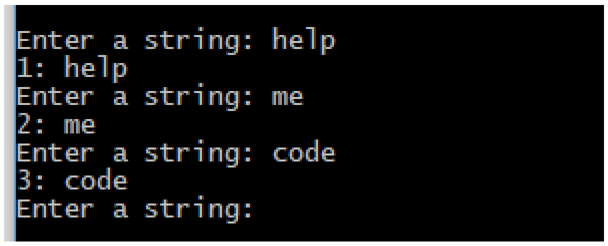
Occasionally, while solving a problem, you need to loop through values as we just did in the while statement example. However, what if you needed to accomplish something no matter what the check value is? That is where the so statement comes in to play. Consider the code shown below.

```
            // Do statement
            string u = "";
            do
            {
                Console.Write("Enter a string: ");
                u = Console.ReadLine();
                if (string.IsNullOrEmpty(u))
                    Console.WriteLine(u);
            } while (!string.IsNullOrEmpty(u));
            Console.WriteLine();
```

This code will continue to ask for user input as long as their input has some characters in it. Once the string is determined to be null then the program falls out of the loop.

Note: This construct is also known as a do-while loop.

Your task is to implement a do statement that keeps track of how many times the loop has been used and outputs the count in front of the string. The output should look like the following.



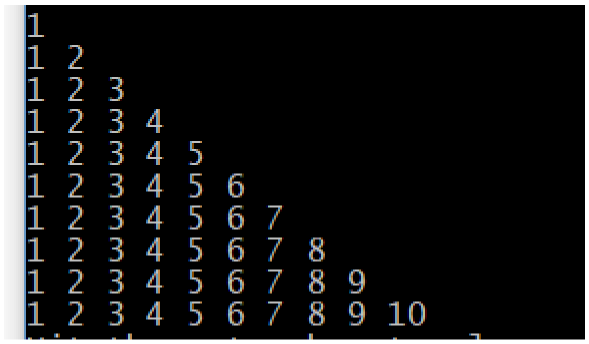
```
Enter a string: help
1: help
Enter a string: me
2: me
Enter a string: code
3: code
Enter a string:
```

For Statement

The for statement is very useful when you want to step through values with controlled starting point, increments and stopping conditions. Consider the code presented in the MS tutorial.

```
    for (int i = 0; i < args.Length; i++)
    {
        Console.WriteLine(args[i]);
    }
```

This code simply outputs the value of the variable i. In this example, i goes from 0 to whatever the length of args is. Your task if to write a for statement that outputs the variable values of the loop on one line as shown below. This will require you to use nested for statements.



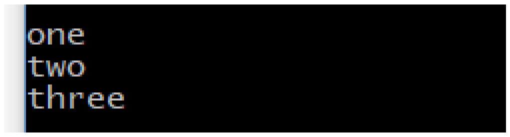
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
```

Foreach Statement

A special use of the for control structure is the foreach structure which uses an array of items as input and “walks” through them. What we do with the items is not important as we are looking at how to use the foreach statement. Consider the t array of strings that we have been using. Using the foreach statement we can unravel the array and output the values to the screen.

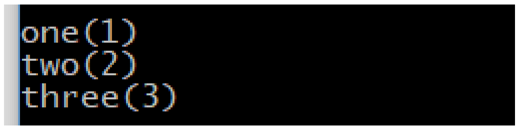
```
// Foreach statement
foreach (string v in t)
{
    Console.WriteLine(v);
}
Console.WriteLine();
```

The output of this code looks like the image below.



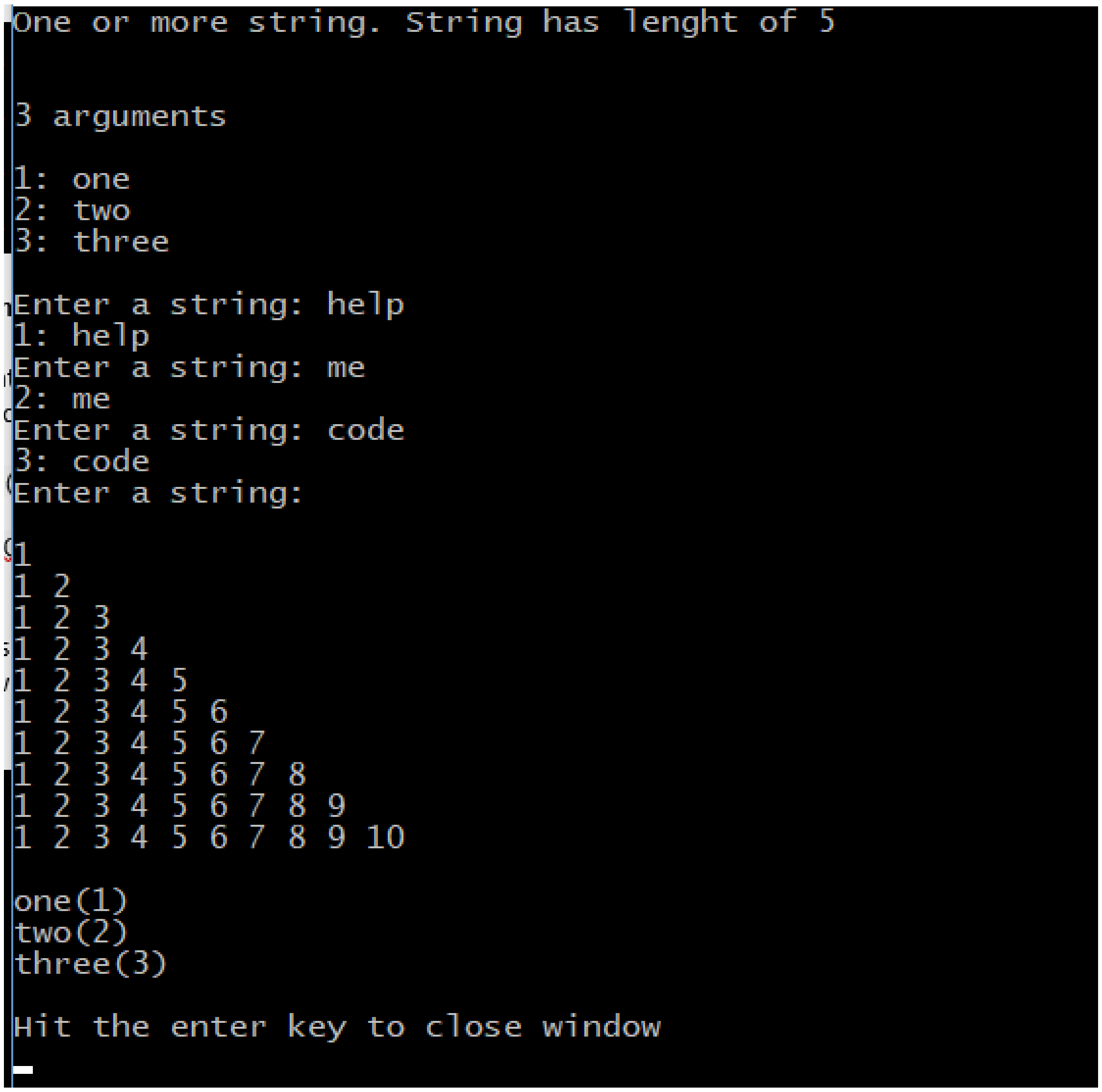
```
one
two
three
```

Your task is to modify this code so that we put the number of the array item is put in parenthesis after the word. It should look like the image below.



```
one(1)
two(2)
three(3)
```

When you are completed with your project for this lesson the output should look something like the image below. Once you have completed these programming tasks submit the statements.cs file to Blackboard.



```
One or more string. String has lenght of 5


3 arguments

1: one
2: two
3: three

Enter a string: help
1: help
Enter a string: me
2: me
Enter a string: code
3: code
Enter a string:

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10

one(1)
two(2)
three(3)

Hit the enter key to close window
```

As always, ask questions on the Discussion Forum.