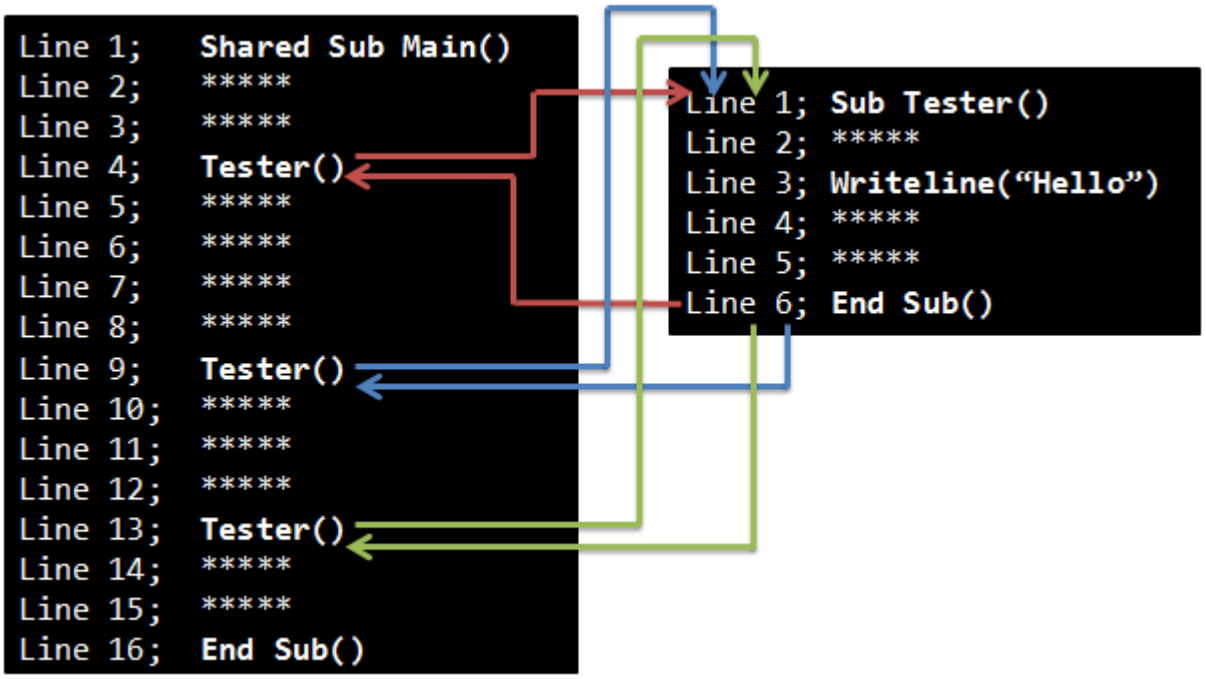
Getting Started

Procedural Programming

Procedural programming is a method of developing computer programs – programs that are sequential and linear in fashion, with specific focus on the order in which tasks are completed. Consider procedural programming as A before B and B before C. Procedural programs are normally command-line based and thus are not normally graphical in appearance. The example below shows the flow of a procedural program; each instruction is executed in order and one at a time.



My First Program

Open Notepad or Notepad++ and type the following code (pay attention to the indentation used):

Code:	**Explanation:**

Imports System.Console
Public Class MyFirstProgram
  Shared Sub Main()
  End Sub
End Class
	1. `Imports System.Console` – this is an imported library
	2. `Public Class MyFirstProgram` – a class is a 'blueprint' for an object. However, for now consider this as indicating the start of the program, as well as the name.
	3. `Shared Sub Main()` – every program has to have a Sub Main() (an initial starting method)
	4. `End Sub` – end of Sub Main()
	5. `End Class` – end of the program

Next, add the following programming instruction to Sub Main():

```
Imports System.Console
Public Class MyFirstProgram
    Shared Sub Main()
        WriteLine("Hello World")
    End Sub
End Class
```

Once the above line of code has been added to Sub Main(), save the file as 'first.vb'. Be sure to change the 'Save As Type' to 'All Files'. Now open the Visual Studio Command Prompt (this is a utility program that is packaged with visual studios – Microsoft's software development environment) and type the following instructions (note that the 'c:\student temp' directory should be replaced with the location and folder name of where your VB file has been saved):

```
cd c:\student temp
vbc first.vb /t:exe
first.exe
```

Program Output:

Test Your Skills

- ✓ *Now modify the code so that the console prints your full name, address and phone number on a separate line each. Remember you will need to compile your program each time you change it.*

  *Use the '**vbc first.vb /t:exe**' command to achieve this. You can also use the up arrow key to reprint previous commands typed.*

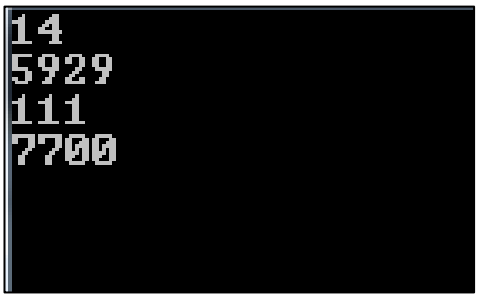
My Lucky Seven

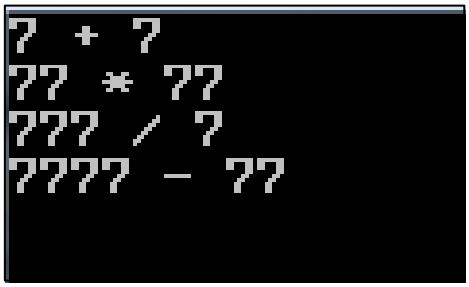
Open Notepad or Notepad++ and type the following code (pay attention to the indentation used):

```
Imports System.Console
Public Class MyLuckySeven
    Shared Sub Main()
        WriteLine(7 + 7)
        WriteLine(77 * 77)
        WriteLine(777 / 7)
        WriteLine(7777 - 77)
    End Sub
End Class
```

Once the code has been typed up, compile and run the program. See previous steps if you cannot remember how to achieve this. Try this code both with and without double quotes, i.e. 7+7 and "7+7".

Program Output:





Concatenation can be used to join 'parts' together. This can consist of both 'string' (text) and program instructions. Modify the original code so that it appears the same as the code shown below. Then compile and run the program again.

```
Imports System.Console
Public Class MyLuckySeven
    Shared Sub Main()
        WriteLine("7 + 7 = " & 7+7)
        WriteLine("77 * 77 = " & 77 * 77)
        WriteLine("777 / 7 = " & 777 / 7)
        WriteLine("7777 + 77 = " & 7777 - 77)
    End Sub
End Class
```

Test Your Skills

- ✓ *Create a short program that displays the first five sums of the five times table.*

Using Variables

A variable is an allocation in memory. Variables are very useful as a programmer can use them to store user input, answers from calculations and/or values for later use. There are many types of variable; one such variable type is Integer, which is used to store whole numbers. To declare a variable, the following syntax is used (note X is the name of the variable and As Integer is the type):

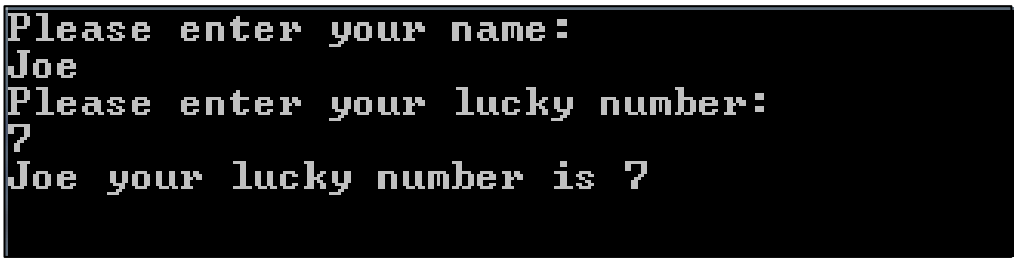
```
Dim X As Integer
```

Open Notepad or Notepad++ and type the following code (pay attention to the indentation used):

```
Imports System.Console
Public Class UsingVariables
    Shared Sub Main()
        Dim X As Integer
        Dim Y As String
        WriteLine("Please enter your name:")
        Y = ReadLine()
        WriteLine("Please enter your lucky number:")
        X = ReadLine()
        WriteLine(Y & " your lucky number is " & X)
    End Sub
End Class
```

Once the code has been typed up, compile and run the program. See previous steps if you cannot remember how to achieve this. Note that the ReadLine command is the opposite of WriteLine; ReadLine allows a programmer to gain input from the user and store the value entered into a variable ready for processing.

Program Output:



Test Your Skills

- ✓ *Create a short program that allows the user to enter two numbers. The program must then calculate the answer of those two numbers added together, multiplied together, divided together and subtracted by each other. The program must then display all the calculations and all of their respective answers.*

Applying Comments

Comments can be added to programming code; comments are not executed by the computer, but are instead visual aids for the programmer. For example, comments can be used for:

- Specifying the author and version of the software
- 'Commenting out' unwanted or old code
- Testing purposes (commenting out segments of code)
- Recording changes made to the software
- Leaving notes for other programmers

Modify the program that you wrote earlier (the 'Using Variables' program) to include your name, date and software version in the comments of the code. Comments are added by using the apostrophe symbol (') at the start of a line.

For example:

```
'*************************
'Joe Smith
'Version 1.1
'*************************
```

Using Procedures

In procedural programming, code is broken down into 'procedures'. In VB.NET this is achieved using Subs. For clarification purposes, a method, sub, procedure and function refer to the same thing. They refer to an executable block of code (with a specific purpose) that can be called by a program. Procedures can be used to 'break up' programs, making them easier to build and maintain.

Open Notepad or Notepad++ and type the following code (pay attention to the indentation used). Once the code has been typed up, compile and run the program.

```
Imports System.Console

Public Class UsingProcedures
        Shared Sub Main()
            WriteLine("Sub Run 1")
            Tester()
            WriteLine("Sub Run 2")
            Tester()
        End Sub
        Shared Sub Tester()
            WriteLine ("I am Sub Tester")
        End Sub
End Class
```

Program Output: