Introduction – Applications (Module 1)

a. Console.Write (Module 1)

1. Console.WriteLine() is used as a VB statement.

2. This is used if you want to create a sentence, add a word or two, or display the variables created/declared earlier in the program (whatever you declare early in the program needs to be shown/displayed later, which is where this type of statement comes in).

3. This can also be used to display the result of calculations or equations that are typed earlier in the program.

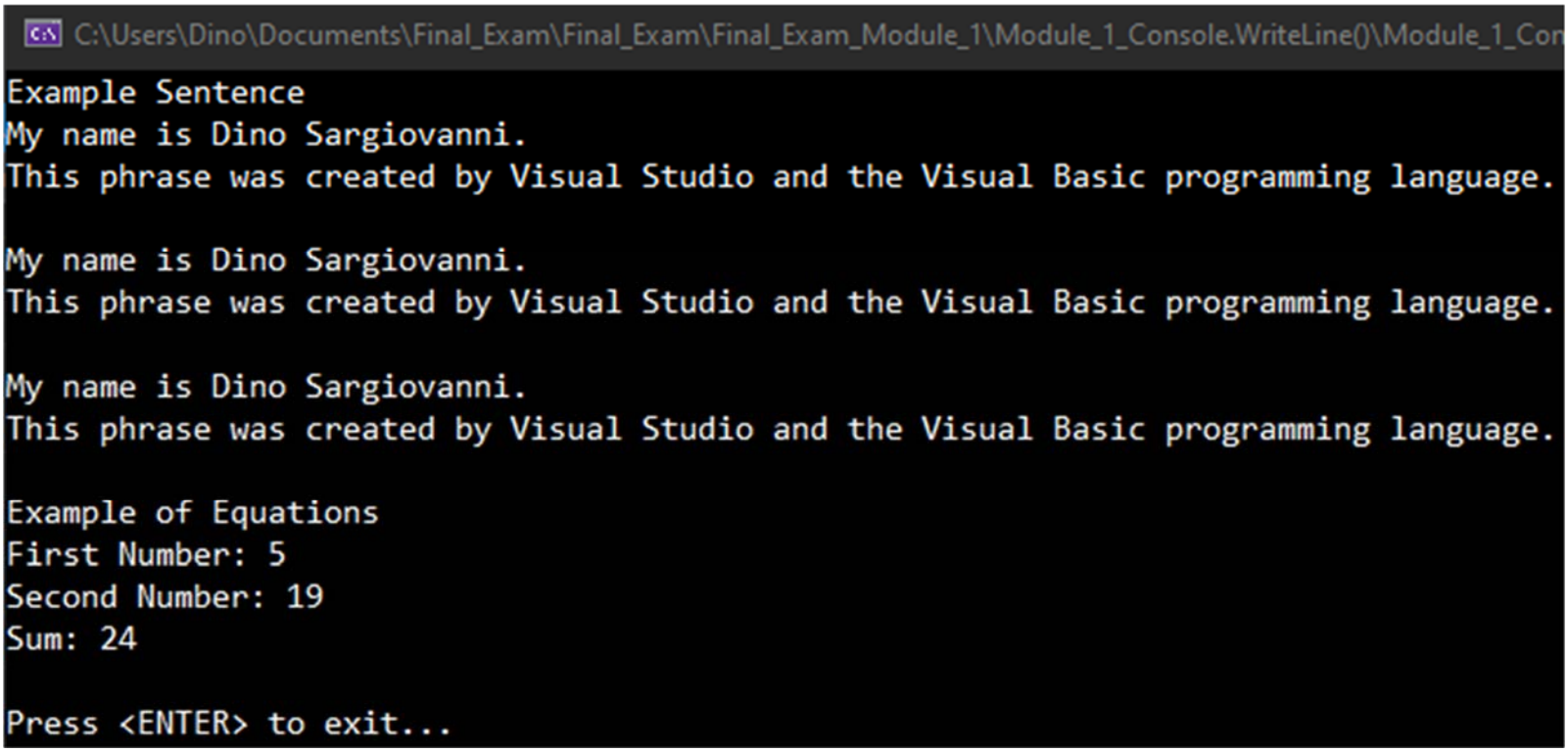
```
1   Imports System
2   Module Program
3
4       'Declare the variables and their data types
5       Dim firstSentence As String
6       Dim secondSentence As String
7
8       Sub Main(args As String())
9
10          'There are different ways you can use the Console.WriteLine() method.
```

```
11
12          'One way is with no variables used (Write the sentence as-is)
13          Console.WriteLine("Example Sentence")
14          Console.WriteLine("My name is Dino Sargiovanni.")
15          Console.WriteLine("This phrase was created by Visual Studio and the Visual Basic programming language.")
16
17          Console.WriteLine("")
18
```

```
19          'A second way is to declare variables and use them in the string.
20          firstSentence = "My name is Dino Sargiovanni"
21          secondSentence = "This phrase was created by Visual Studio and the Visual Basic programming language"
22
23          Console.WriteLine(firstSentence + "." +
24          Environment.NewLine + secondSentence + "." + Environment.NewLine)
```

```
25
26      'A third way is to use String Inpolation
27      '(a dollar symbol, quotes, and braces that isolate the variables from the rest of the sentence)
28      Console.WriteLine($"{firstSentence}.")
29      Console.WriteLine($"{secondSentence}.")
30
31      'You can use Console.WriteLine() to add a line to separate paragraphs or entries
32      Console.WriteLine()
33
34      'For equations, you can use Console.WriteLine to help with displaying your solution/calculation
35      Dim firstNumber As Integer
36      Dim secondNumber As Integer
37
```

```
38          firstNumber = 5
39          secondNumber = 19
40
41          'Using String Interpolation
42          Console.WriteLine("Example of Equations")
43          Console.WriteLine($"First Number: {firstNumber}")
44          Console.WriteLine($"Second Number: {secondNumber}")
45
46          Dim sum As Integer = (firstNumber + secondNumber)
47
48          'String Interpolation
49          Console.WriteLine($"Sum: {sum}")
50
51          Console.WriteLine()
52          Console.WriteLine("Press <ENTER> to exit...")
53          Console.ReadKey()
54
55      End Sub
56  End Module
```


C:\Users\Dino\Documents\Final_Exam\Final_Exam\Final_Exam_Module_1\Module_1_Console.WriteLine()\Module_1_Con
Example Sentence
My name is Dino Sargiovanni.
This phrase was created by Visual Studio and the Visual Basic programming language.
My name is Dino Sargiovanni.
This phrase was created by Visual Studio and the Visual Basic programming language.
My name is Dino Sargiovanni.
This phrase was created by Visual Studio and the Visual Basic programming language.
Example of Equations
First Number: 5
Second Number: 19
Sum: 24
Press <ENTER> to exit...

Variables & Data Types (Module 2)

a. Double Data Type (Module 2)

1. Doubles as a data type are used when you want to include numbers with decimal places with precision (at least one decimal place).

2. A double is different than an integer, because you can have the numbers as approximates and not as exact (or whole) numbers.

3. This data type can either convert to another data type, or another data type can convert to a double data type.

```vb
Imports System

Module Program
    Sub Main(args As String())
        'Facts about Ohio
        'Data types including Integers and Doubles
        Dim stateNumber As Integer
        Dim stateArea As Double
        Dim statePopulation As Double
        Dim countyNumber As Integer
        Dim averagePopulation As Double
        Dim averageCounty As Double

        'Define the variables
        stateNumber = 17
        stateArea = 44825
        statePopulation = 11690000
        countyNumber = 88
        averagePopulation = (statePopulation / countyNumber)
        averageCounty = (stateArea / countyNumber)

        'Print the variables
        Console.WriteLine("Facts about Ohio")
        Console.WriteLine($"Ohio is the {stateNumber}th state in the United States.")
        Console.WriteLine($"The area of Ohio in square miles is {stateArea}.")
        Console.WriteLine($"The population of Ohio as of 2019 was {statePopulation}.")
        Console.WriteLine($"In Ohio, there are {countyNumber} counties.")
        Console.WriteLine($"The average population of people per county is approximately {averagePopulation:F2}")
        Console.WriteLine($"The average area of a county in Ohio in square miles is {averageCounty}")

        Console.WriteLine()
        Console.WriteLine("Press <ENTER> to exit...")
        Console.ReadKey()
    End Sub
End Module
```

```
C:\Users\Dino\Documents\Final_Exam\Final_Exam\Final_Exam_Module_2\Module_2_Double_Data_Ty
Facts about Ohio
Ohio is the 17th state in the United States.
The area of Ohio in square miles is 44825.
The population of Ohio as of 2019 was 11690000.
In Ohio, there are 88 counties.
The average population of people per county is approximately 132840.91
The average area of a county in Ohio in square miles is 509.375

Press <ENTER> to exit...
```

Numbers & Operators (Module 3)

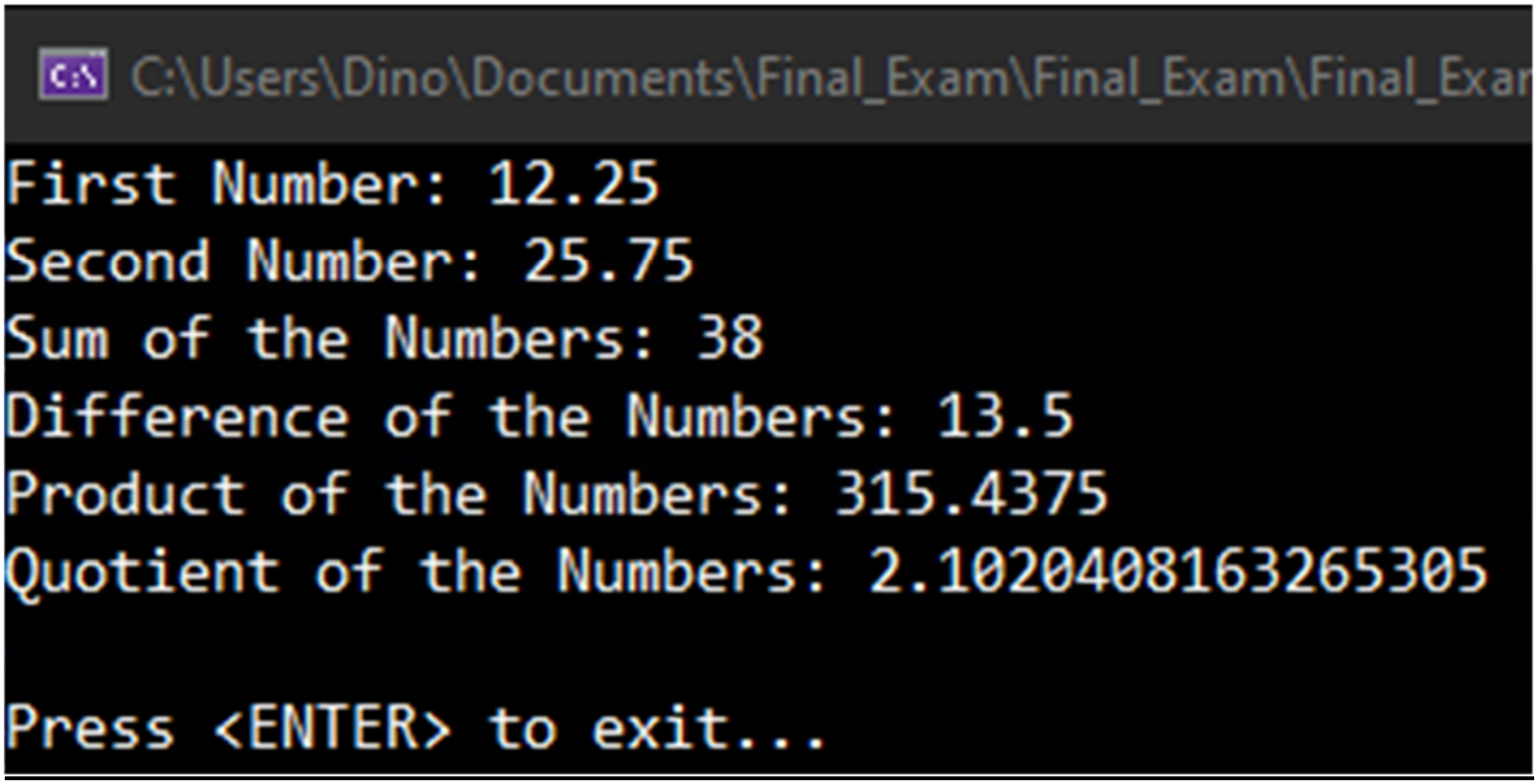
a. Equal Assignment Operator (Module 3)

1. An equal (=) assignment operator is typically used to assign a value to a variable or property.

2. In fact, this operator is considered a Compound Assignment Operator, which are used to perform an operation on the value of the variable before assigning an updated value to the variable.

3. Because the equal operator is also a compound assignment operator, you can perform two tasks in one step.

```
1   Imports System
2   Module Program
3       Sub Main(args As String())
4
5
6           'Example using Doubles
7           Dim firstNumber As Double = 12.25
8           Dim secondNumber As Double = 25.75
9
10          'Declare operations
11          Dim sum As Double
12          Dim difference As Double
13          Dim product As Double
14          Dim quotient As Double
15
16          'Define equations for operations
17          sum = firstNumber + secondNumber
18          difference = secondNumber - firstNumber
19          product = secondNumber * firstNumber
20          quotient = secondNumber / firstNumber
21
22          'Print the variables
23          Console.WriteLine($"First Number: {firstNumber}")
24          Console.WriteLine($"Second Number: {secondNumber}")
25          Console.WriteLine($"Sum of the Numbers: {sum}")
26          Console.WriteLine($"Difference of the Numbers: {difference}")
27          Console.WriteLine($"Product of the Numbers: {product}")
28          Console.WriteLine($"Quotient of the Numbers: {quotient}")
29
30          Console.WriteLine()
31          Console.WriteLine("Press <ENTER> to exit...")
32          Console.ReadKey()
33
34      End Sub
35  End Module
```


C:\Users\Dino\Documents\Final_Exam\Final_Exam\Final_Exar
First Number: 12.25
Second Number: 25.75
Sum of the Numbers: 38
Difference of the Numbers: 13.5
Product of the Numbers: 315.4375
Quotient of the Numbers: 2.1020408163265305
Press <ENTER> to exit...

Strings, Booleans, Date & Time (Module 4)

a. Strings (Comparison with String.Compare Method) (Module 4)

1. Strings are a data type that consist of letters, words, and numbers (otherwise known as characters).

2. With strings, they are enclosed with a set of double quotation marks (ex. “This is an example of a string.”).

3. You can compare different strings in your program with different methods, such as Option Compare Binary, Option Compare Text, StrComp, and String.Compare; String.Compare uses the Binary comparison, but it returns an integer.

```vb
Imports System

Module Program
    Sub Main(args As String())

        'Declare variables
        Dim string1, string2, string3 As String
        string1 = "Dino"
        string2 = "dino"
        string3 = "DINO"

        'String.Compare Method
        Dim comp12 As Integer = String.Compare(string1, string2)     ' 1 because D succeeds d
        Dim comp13 As Integer = String.Compare(string1, string3)     ' -1 because ino precedes INO (after d=d)
        Dim comp21 As Integer = String.Compare(string2, string1)     ' -1 because d precedes D
        Dim comp23 As Integer = String.Compare(string2, string3)     ' -1 because dino precedes DINO
        Dim comp31 As Integer = String.Compare(string3, string1)     ' 1 because INO succeeds ino (after D = D)
        Dim comp32 As Integer = String.Compare(string3, string2)     ' 1 because DINO succeeds dino

        'Print the varaibles
        Console.Write($"{comp12}" + vbCrLf + $"{comp13}" + vbCrLf + $"{comp21}" + vbCrLf + $"{comp23}" + vbCrLf + $"{comp31}" + vbCrLf + $"{comp32}")

        Console.WriteLine()
        Console.WriteLine("Press <ENTER> to exit...")
        Console.ReadKey()

    End Sub
End Module
```

```
C:\Users\Dino\Documents\Final_
1
-1
-1
-1
1
1
Press <ENTER> to exit...
```

Statements (Module 5)

a. If-Then-End If (Module 5)

1. This statement is a conditional statement with only one condition.

2. When the condition is met, then the statement executes a written/given task.

3. You can also use multiple **If-Then-End If** when there are more than two conditions, and when the conditions are uniquely different (this is a different way of writing another statement such as the **If-Then-ElseIf-Else** statement).

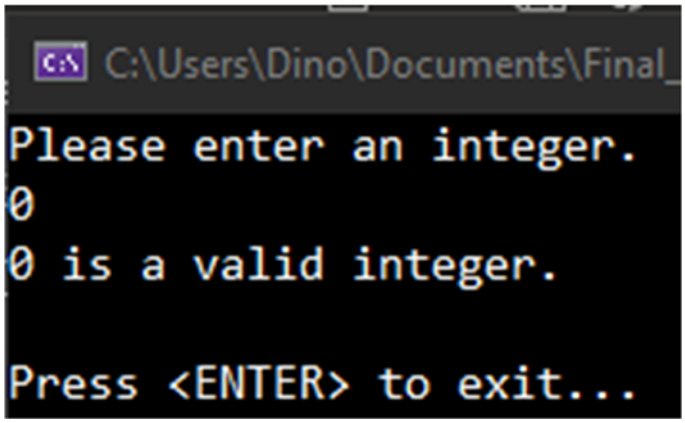
```
1   Imports System
2
3   Module Program
4       Sub Main(args As String())
5
6           'Declare a variable
7           Dim number As Integer
8
9           Console.WriteLine("Please enter an integer.")
10          number = Console.ReadLine()
11
12          'If-Then-End If statement
13          If number >= 0 Then
14              Console.WriteLine($"{number} is a valid integer.")
15          End If
16
17          'Another condition for the statement
18          If number < 0 Then
19              Console.WriteLine($"Sorry, but {number} is not a valid integer.")
20          End If
21
22          Console.WriteLine()
23          Console.WriteLine("Press <ENTER> to exit...")
24          Console.ReadKey()
25
26      End Sub
27  End Module
```

```
C:\Users\Dino\Documents\Final_E
Please enter an integer.
5
5 is a valid integer.

Press <ENTER> to exit...
```

```
C:\Users\Dino\Documents\Final_Exam\Final_Exam\
Please enter an integer.
-2
Sorry, but -2 is not a valid integer.

Press <ENTER> to exit...
```



b. If-Then-Else-End If (Module 5)

1. This statement is a conditional statement with only two conditions.
2. If the first condition is met, then one task is executed; otherwise, another task is executed.
3. The Else refers to another condition, and the condition of the Else statement is not specified, since it is the only other condition.

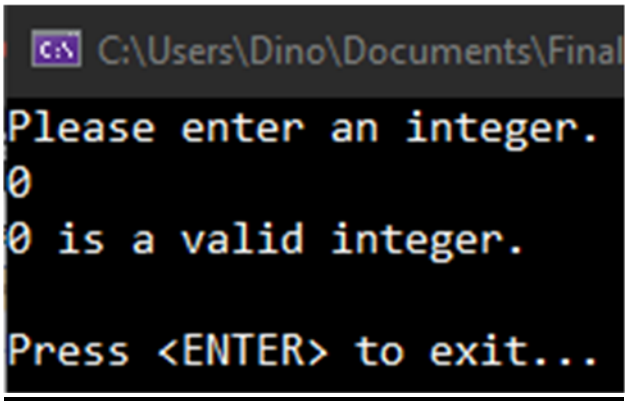
```
1   Imports System
2
3   Module Program
4       Sub Main(args As String())
5           'Declare a variable
6           Dim number As Integer
7
8           'Input for the user to type a number
9           Console.WriteLine("Please enter an integer.")
10          number = Console.ReadLine()
11
12          'If-Then-End If statements with more than one condition (two conditions)
13          'Prints the output as well
14          If number >= 0 Then
15              Console.WriteLine($"{number} is a valid integer.")
16          Else
17              Console.WriteLine($"Sorry, but {number} is not a valid integer.")
18          End If
19
20          Console.WriteLine()
21          Console.WriteLine("Press <ENTER> to exit...")
22          Console.ReadKey()
23      End Sub
24  End Module
```

```
C:\Users\Dino\Documents\Final_
Please enter an integer.
7
7 is a valid integer.

Press <ENTER> to exit...
```

```
C:\Users\Dino\Documents\Final_Exam\Final_Exam
Please enter an integer.
-3
Sorry, but -3 is not a valid integer.

Press <ENTER> to exit...
```


C:\Users\Dino\Documents\Final
Please enter an integer.
0
0 is a valid integer.
Press <ENTER> to exit...

Procedures (Module 6)

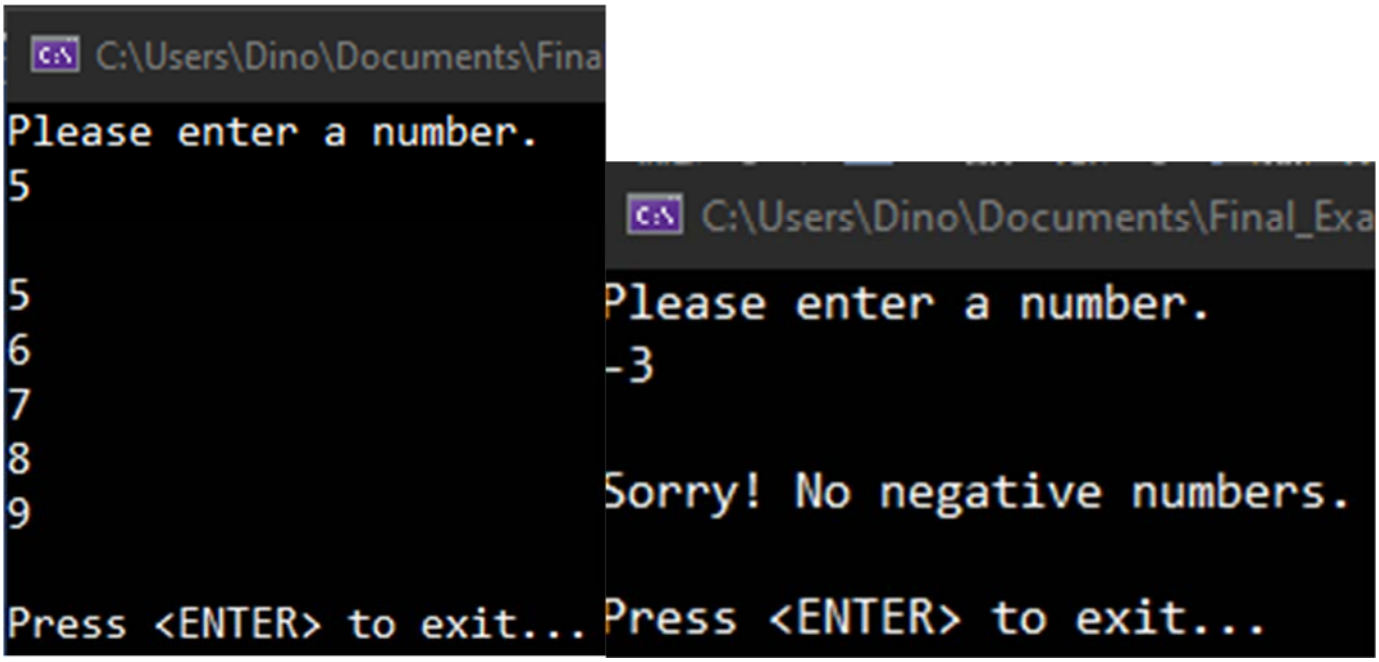
a. Sub Procedure (Module 6)

1. A Sub procedure is a series of VB statements that are enclosed by Sub and End Sub statements.

2. It performs a task, but it does not return a value; this means it does not have any single return data type, but it can return a statement that contains the return types in a String.

3. To create a Sub procedure, be sure to follow the Sub keyword with the name of the procedure (in the Sub Statement), then list the parameter in parentheses, and place the code statements between the Sub and End Sub statements.

```
1   'Inspired from DB5 posts from Module 6
2   Imports System
3   Module Program
4       Sub NumberList(number1 As Integer, number2 As Integer)
5           For numberGroup As Double = number1 To number2
6               If number1 > 0 Then
7                   Console.WriteLine($"{numberGroup} ")
8               End If
9           Next
10      End Sub
11      Sub Main(args As String())
12
13          'Declare the variables
14          Dim firstNumber As Integer
15          Dim lastNumber As Integer
16
17          'Asks for the input from the user
18          Console.WriteLine("Please enter a number.")
19          firstNumber = Console.ReadLine()
20          lastNumber = (firstNumber + 4)
21          Console.WriteLine()
22          NumberList(firstNumber, lastNumber)
23
24          'Output for a different condition
25          If firstNumber < 0 Then
26              Console.WriteLine("Sorry! No negative numbers.")
27          End If
28
29          Console.WriteLine()
30          Console.WriteLine("Press <ENTER> to exit...")
31          Console.ReadKey()
32
33      End Sub
34  End Module
```



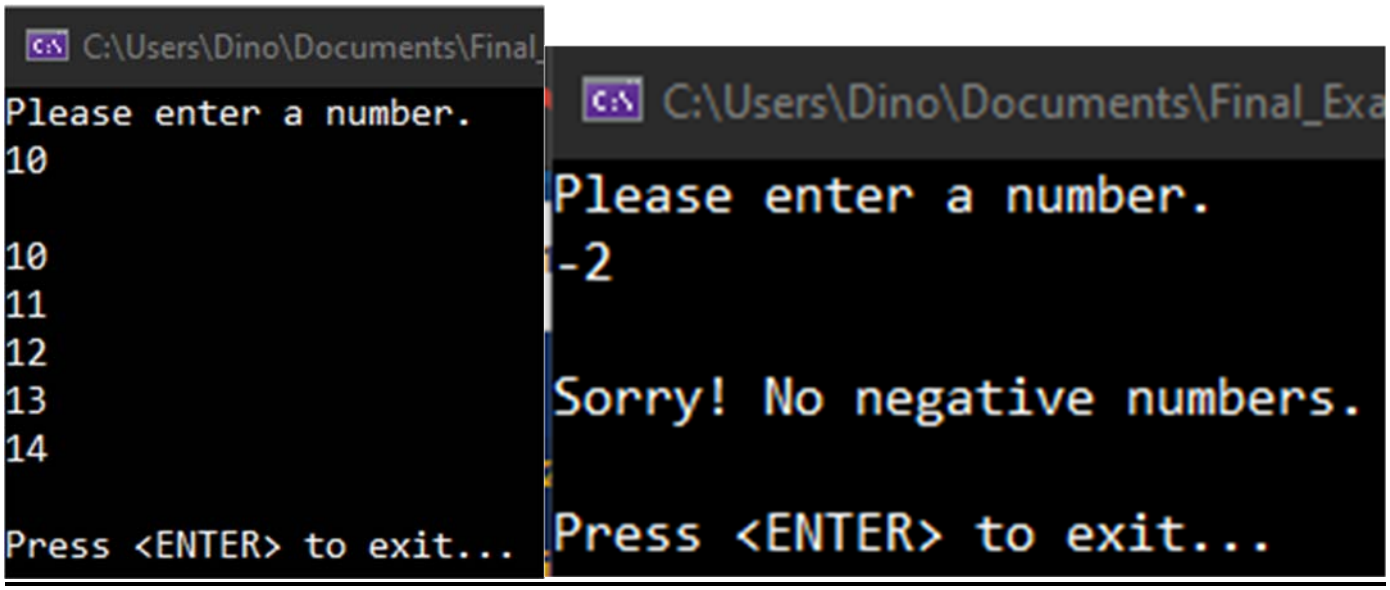
b. Function Procedure (Module 6)

1. A Function procedure that is enclosed by the Function and End Function statements.

2. It performs a task, and it returns a value or an array of values of a specific return data type.

3. To create a Function Procedure, follow the Function keyword with the name of the procedure (in the Function statement), list the parameters in parentheses, add an As clause specifying the data type of the return value, and place the code statements between the Function and End Function statements; you can also use a Return statement to return the value to the callin code.

```
1   'Inspired from DB5 posts from Module 6
2   Imports System
3   Module Program
4       Function NumberList(number1 As Integer, number2 As Integer)
5           For numberGroup As Double = number1 To number2
6               If number1 > 0 Then
7                   Console.WriteLine($"{numberGroup} ")
8               End If
9           Next
10      End Function
11      Sub Main(args As String())
12
13          'Declare the variables
14          Dim firstNumber As Integer
15          Dim lastNumber As Integer
16
17          'Input from the user
18          Console.WriteLine("Please enter a number.")
19          firstNumber = Console.ReadLine()
20          lastNumber = (firstNumber + 4)
21          Console.WriteLine()
22          NumberList(firstNumber, lastNumber)
23
24          'Output for a different condition
25          If firstNumber < 0 Then
26              Console.WriteLine("Sorry! No negative numbers.")
27          End If
28
29          Console.WriteLine()
30          Console.WriteLine("Press <ENTER> to exit...")
31          Console.ReadKey()
32      End Sub
33  End Module
```


C:\Users\Dino\Documents\Final_
Please enter a number.
10
10
11
12
13
14
Press <ENTER> to exit...
C:\Users\Dino\Documents\Final_Exa
Please enter a number.
-2
Sorry! No negative numbers.
Press <ENTER> to exit...

Arrays & Tuples (Module 7)

a. For Each…Next Loop (Module 7)

1. The For Each…Next Loop is a type of iteration statement for arrays and declaring variables for arrays (Iteration statements (loops) performs/repeats the same set of statements until a condition is met).
2. This is one of the main loops used to help display values in arrays and loop through collections such as arrays.
3. What it does is it loops through each array element without using a counter, and the length of the array is not needed, unlike the other loops in the program (ex. For… Next Loop, While…End Loop, Do-While Loop, and Do-Until Loop).

```
Imports System
Module Program
    Sub Main(args As String())

        'Declaring the array with a raw, ungrouped dataset
        Dim data() As Double = {24, 48, 22, 9, 45}

        'Declaring the length of the array
        Dim dataLength As Integer
        dataLength = data.Length()

        'For-Each... Next Loop for this array
        Console.WriteLine($"List of Numbers")
        For Each unit As Double In data
            Console.WriteLine($"{unit} ")
        Next
        Console.WriteLine()

        'Sort the dataset in ascending order
        Array.Sort(data)
        Console.WriteLine($"Ascending Order")
        For Each sort As Double In data
            Console.WriteLine($"{sort }")
        Next
        Console.WriteLine()

        Dim mean = data.Average()

        Dim variance
        Dim standardDeviation

        variance = 0

        For Each unit As Double In data
            variance += (unit - mean) ^ 2
        Next

```

```
38          variance /= dataLength - 1
39
40          standardDeviation = Math.Sqrt(variance)
41
42          Console.WriteLine($"Mean: {mean}")
43          Console.WriteLine($"Variance: {variance}")
44          Console.WriteLine($"Standard Deviation: {standardDeviation}")
45
46          Console.WriteLine()
47          Console.WriteLine("Press <ENTER> to exit...")
48          Console.ReadKey()
49
50      End Sub
51  End Module
```

```
C:\Users\Dino\Documents\Final_Exam\Final_Exam\F
List of Numbers
24
48
22
9
45

Ascending Order
9
22
24
45
48

Mean: 29.6
Variance: 272.3
Standard Deviation: 16.501515081955354

Press <ENTER> to exit...
```

Multidimensional Arrays & Jagged Arrays (Module 8)

a. Multiple Dimensions of Arrays (Module 8)

1. Multidimensional arrays are arrays that have more than one dimension.

2. The structure is different for multiple dimensions, including 2 and 3 dimensions, which represent different counts for a combination of variables, which are held in spots called indexes.

3. The indexes represent the “dimensions” or the “width” and “height” of a dataset, and this helps determine their “position” on the array, as a format like a table (2-dimension) or like a cube (3-dimension).

Before Switch in Array

```
1   Imports System
2
3   Module Program
4       Sub Main(args As String())
5
6           Dim array() As Double = {24, 48, 88, 5}
7
8           Console.WriteLine("Number List")
9           For Each number As Integer In array
10              Console.WriteLine($"{number} ")
11          Next
12          Console.WriteLine()
13
14          Dim multi(2, 2) As Double
15          multi(0, 0) = 5
16          multi(0, 1) = 24
17          multi(1, 0) = 48
18          multi(1, 1) = 88
19
20          Console.WriteLine("Multidimensional Array")
21          Console.WriteLine(multi(0, 0))
22          Console.WriteLine(multi(0, 1))
23          Console.WriteLine(multi(1, 0))
24          Console.WriteLine(multi(1, 1))
25
26          Console.WriteLine()
27          Console.WriteLine("Press <ENTER> to exit...")
28          Console.ReadKey()
29
30      End Sub
31  End Module
```

```
C:\Users\Dino\Documents\Final
Number List
24
48
88
5

Multidimensional Array
5
24
48
88

Press <ENTER> to exit...
```

After Switch in Array (changed indexes to change positions in array)

```
1   Imports System
2
3   Module Program
4       Sub Main(args As String())
5
6           Dim array() As Double = {24, 48, 88, 5}
7
8           Console.WriteLine("Number List")
9           For Each number As Integer In array
10              Console.WriteLine($"{number} ")
11          Next
12          Console.WriteLine()
13
14          Dim multi(2, 2) As Double
15          multi(1, 0) = 5
16          multi(0, 1) = 24
17          multi(0, 0) = 48
18          multi(1, 1) = 88
19
20          Console.WriteLine("Multidimensional Array")
21          Console.WriteLine(multi(0, 0))
22          Console.WriteLine(multi(0, 1))
23          Console.WriteLine(multi(1, 0))
24          Console.WriteLine(multi(1, 1))
25
26          Console.WriteLine()
27          Console.WriteLine("Press <ENTER> to exit...")
28          Console.ReadKey()
29
30      End Sub
31  End Module
```

```
C:\Users\Dino\Documents\Final
Number List
24
48
88
5

Multidimensional Array
48
24
5
88

Press <ENTER> to exit...
```