VB.NET Functions

In VB.NET, the function is a separate group of codes that are used to perform a specific task when the defined function is called in a program. After the execution of a function, control transfer to the **main()** method for further execution. It returns a value. In VB.NET, we can create more than one function in a program to perform various functionalities. The function is also useful to code reusability by reducing the duplicity of the code. For example, if we need to use the same functionality at multiple places in a program, we can simply create a function and call it whenever required.

Defining a Function

The syntax to define a function is:

```
[Access_specifier ] Function Function_Name [ (ParameterList) ] As Return_Type
[ Block of Statement]
Return return_val
End Function
```

Where,

- **Access_Specifier:** It defines the access level of the function such as public, private, or friend, Protected function to access the method.
- **Function_Name:** The function_name indicate the name of the function that should be unique.
- **ParameterList:** It defines the list of the parameters to send or retrieve data from a method.
- **Return_Type:** It defines the data type of the variable that returns by the function.

The following are the various ways to define the function in a VB.NET.

```
Public Function add() As Integer
' Statement to be executed
End Function

Private Function GetData( ByVal username As String) As String
' Statement to be executed
End Function

Public Function GetData( ByVal username As String, ByVal userId As Integer) As String
' Statement to be executed
End Function
```

Example: Write a program to find the sum and subtraction of two numbers using the function.

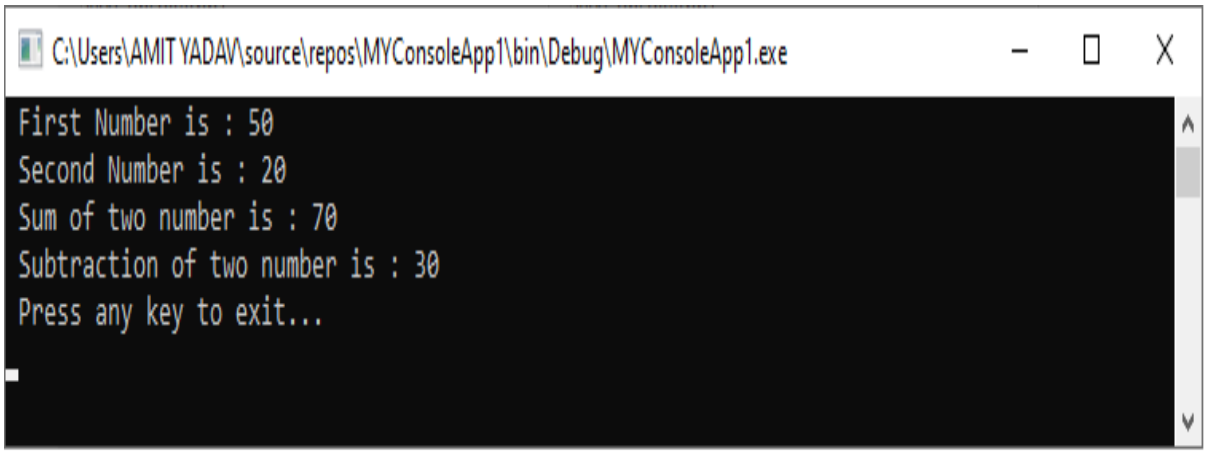
Find_Sum.vb

```
Imports System
Module Find_Sum
  ' Create the SumOfTwo() Function and pass the parameters.
  Function SumOfTwo(ByVal n1 As Integer, ByVal n2 As Integer) As Integer
    ' Define the local variable.
    Dim sum As Integer = 0
    sum = n1 + n2
    Return sum
  End Function
  Function SubtractionOfTwo(ByVal n1 As Integer, ByVal n2 As Integer) As Integer
    ' Define the local variable.
    Dim subtract As Integer
    subtract = n1 - n2
    Return subtract
  End Function
```

```
  Sub Main()
    ' Define the local variable a and b.
    Dim a As Integer = 50
    Dim b As Integer = 20
    Dim total, total1 As Integer
    total = SumOfTwo(a, b) 'call SumOfTwo() Function
    total1 = SubtractionOfTwo(a, b) 'call SubtractionOfTwo() Function
    Console.WriteLine(" Sum of two number is : {0}", total)
    Console.WriteLine(" Subtraction of two number is : {0}", total1)

  End Sub
End Module
```

Output:



In the above example, we have defined a **SumOfTwo() and SubtractionOfTwo()** function to add and subtract two predefined numbers. When the functions are called in the main() method, each function is executed and returns the sum and subtraction of two numbers, respectively.

VB.NET Sub

A Sub procedure is a separate set of codes that are used in VB.NET programming to execute a specific task, and it does not return any values. The Sub procedure is enclosed by the Sub and End Sub statement. The Sub procedure is similar to the function procedure for executing a specific task except that it does not return any value, while the function procedure returns a value.

Defining the Sub procedure

Following is the syntax of the Sub procedure:

```
[Access_Specifier ] Sub Sub_name [ (parameterList) ]
[ Block of Statement to be executed ]
End Sub
```

Where,

- **Access_Specifier:** It defines the access level of the procedure such as public, private or friend, Protected, etc. and information about the overloading, overriding, shadowing to access the method.
- **Sub_name:** The Sub_name indicates the name of the Sub that should be unique.
- **ParameterList:** It defines the list of the parameters to send or retrieve data from a method.

The following are the different ways to define the types of Sub method.

```
Public Sub getDetails()
' Statement to be executed
End Sub
  Private Sub GetData( ByVal username As String) As String
' Statement to be executed
End Sub
  Public Function GetData1( ByRef username As String, ByRef userId As Integer)
```

```
' Statement to be executed
End Sub
```

Example: Write a simple program to pass the empty, a single or double parameter of Sub procedure in the VB.NET.

Sub_Program.vb

```
Module Sub_Program
  Sub sample()
    Console.WriteLine("Welcome to JavaTpoint")
  End Sub

  Sub circle(ByVal r As Integer)
    Dim Area As Integer
    Const PI = 3.14
    Area = PI * r * r
    Console.WriteLine(" Area Of circle is : {0}", Area)
  End Sub

  ' Create the SumOfTwo() Function and pass the parameters.
  Sub SumOfTwo(ByVal n1 As Integer, ByVal n2 As Integer)
    ' Define the local variable.
    Dim sum As Integer = 0
    sum = n1 + n2
    Console.WriteLine(" Sum of two number is : {0}", sum)
  End Sub

  Sub SubtractionOfTwo(ByVal n1 As Integer, ByVal n2 As Integer)
    ' Define the local variable.
```

```
    Dim subtract As Integer
    subtract = n1 - n2
    Console.WriteLine(" Subtraction of two number is : {0}", subtract)
  End Sub

  Sub MultiplicationOfTwo(ByVal n1 As Integer, ByVal n2 As Integer)
    ' Define the local variable.
    Dim multiply As Integer
    multiply = n1 * n2
    Console.WriteLine(" Multiplication of two number is : {0}", multiply)
  End Sub

  Sub Main()
    ' Define the local variable a, b and rad.
    Dim a, b, rad As Integer
    sample() ' call sample() procedure
    Console.WriteLine(" Please enter the First Number : ")
    a = Console.ReadLine()
    Console.WriteLine(" Second Number is : ")
    b = Console.ReadLine()

    SumOfTwo(a, b) 'call SumOfTwo() Function
    SubtractionOfTwo(a, b) 'call SubtractionOfTwo() Function
    MultiplicationOfTwo(a, b) 'call MultiplicationOfTwo() Function

    Console.WriteLine(" Enter the radius of circle : ")
    rad = Console.ReadLine()
    circle(rad)
    Console.WriteLine(" Press any key to exit...")
    Console.ReadKey()
  End Sub
End Module
```

Output:

```
C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe
Welcome to JavaTpoint
 Please enter the First Number :
100
 Second Number is :
60
 Sum of two number is : 160
 Subtraction of two number is : 40
 Multiplication of two number is : 6000
 Enter the radius of circle :
5
 Area Of circle is : 78
 Press any key to exit...
```

In the VB.NET programming language, we can pass parameters in two different ways:

- Passing parameter by Value
- Passing parameter by Reference

<u>Passing parameter by Value</u>

In the VB.NET, passing parameter by value is the default mechanism to pass a value in the Sub method. When the method is called, it simply copies the actual value of an argument into the formal method of Sub procedure for creating a new storage location for each parameter. Therefore, the changes made to the main function's actual parameter that do not affect the Sub procedure's formal argument.

Syntax:

```
Sub Sub_method( ByVal parameter_name As datatype )
[ Statement to be executed]
End Sub
```

In the above syntax, the **ByVal** is used to declare parameters in a Sub procedure.

Let's create a program to understand the concept of passing parameter by value.

Passing_value.vb

```
Imports System
Module Passing_value
  Sub Main()
    ' declaration of local variable
    Dim num1, num2 As Integer
    Console.WriteLine(" Enter the First number")
    num1 = Console.ReadLine()
    Console.WriteLine(" Enter the Second number")
    num2 = Console.ReadLine()
    Console.WriteLine(" Before swapping the value of 'num1' is {0}", num1)
    Console.WriteLine(" Before swapping the value of 'num2' is {0}", num2)
    Console.WriteLine()

    'Call a function to pass the parameter for swapping the numbers.
    swap_value(num1, num2)
    Console.WriteLine(" After swapping the value of 'num1' is {0}", num1)
    Console.WriteLine(" After swapping the value of 'num2' is {0}", num2)
    Console.WriteLine(" Press any key to exit...")
    Console.ReadKey()
  End Sub

  ' Create a swap_value() method
  Sub swap_value(ByVal a As Integer, ByVal b As Integer)
    ' Declare a temp variable
    Dim temp As Integer
    temp = a ' save the value of a to temp
    b = a   ' put the value of b into a
    b = temp ' put the value of temp into b
```

```
    End Sub
End Module
```

Output:

```
C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe
 Enter the First number
30
 Enter the Second number
20
 Before swapping the value of 'num1' is 30
 Before swapping the value of 'num2' is 20

 After swapping the value of 'num1' is 20
 After swapping the value of 'num2' is 30
 Press any key to exit...
```

Passing parameter by Reference

A Reference parameter is a reference of a variable in the memory location. The reference parameter is used to pass a reference of a variable with **ByRef** in the Sub procedure. When we pass a reference parameter, it does not create a new storage location for the sub method's formal parameter. Furthermore, the reference parameters represent the same memory location as the actual parameters supplied to the method. So, when we changed the value of the formal parameter, the actual parameter value is automatically changed in the memory.

The syntax for the passing parameter by Reference:

```
Sub Sub_method( ByRef parameter_name, ByRef Parameter_name2 )
[ Statement to be executed]
End Sub
```

In the above syntax, the **ByRef** keyword is used to pass the Sub procedure's reference parameters.

Let's create a program to swap the values of two variables using the ByRef keyword.

Passing_ByRef.vb

```
Imports System
Module Passing_ByRef
    Sub Main()
```

```
        ' declaration of local variable
        Dim num1, num2 As Integer
        Console.WriteLine(" Enter the First number")
        num1 = Console.ReadLine()
        Console.WriteLine(" Enter the Second number")
        num2 = Console.ReadLine()
        Console.WriteLine(" Before swapping the value of 'num1' is {0}", num1)
        Console.WriteLine(" Before swapping the value of 'num2' is {0}", num2)
        Console.WriteLine()

        'Call a function to pass the parameter for swapping the numbers.
        swap_Ref(num1, num2)
        Console.WriteLine(" After swapping the value of 'num1' is {0}", num1)
        Console.WriteLine(" After swapping the value of 'num2' is {0}", num2)
        Console.WriteLine(" Press any key to exit...")
        Console.ReadKey()
    End Sub

    ' Create a swap_Ref() method
    Sub swap_Ref(ByRef a As Integer, ByRef b As Integer)
        ' Declare a temp variable
        Dim temp As Integer
        temp = a ' save the value of a to temp
        a = b   ' put the value of b into a
        b = temp ' put the value of temp into b
    End Sub
End Module
```

Output:

C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe

```
 Enter the First number
5
 Enter the Second number
3
 Before swapping the value of 'num1' is 5
 Before swapping the value of 'num2' is 3

 After swapping the value of 'num1' is 3
 After swapping the value of 'num2' is 5
 Press any key to exit...
```