A201 Object Oriented Programming with Visual Basic .Net

By:

Dr. Hossein Hakimzadeh
Computer Science and Informatics
IU South Bend
What do we need to learn in order to write computer programs?

- Fundamental programming constructs:
  - Variables,
  - Arithmetic operators,
  - Input and output
  - Conditionals,
  - Loops,
  - Procedures and functions,
  - Arrays,
  - Structures, classes and objects,
  - Files
Functions

- Functions are similar to sub-procedures, but in addition to performing a specific task, functions also return a value.
Functions

VB .Net has many built in functions:

X = CDbl( )
X = CInt( )
X = CStr( )
X = InputBox( )
X = Pmt( )
X = IsNumeric( )
Functions

- Visual Basic allows the programmers to create their own functions
Functions

- Syntax:

Private Function FunctionName([OptionalParameterList]) As DataType

[Statements]
Return (Expression)

End Function
A function can do anything that a procedure can do ...

It always return a value..
Passing Parameters to a Function

- Parameters can be sent to a procedure either *By Value* or *By Reference*.

- Same as sending parameters to a procedure!!
Calling a Function:

Private Function Cube(ByVal Number As Integer) As Integer

    Return (Number * Number * Number)

End Function

Sub Main()

    Dim result As Integer
    result = Cube(5) 'Pass by value

End Sub
A complete program (Calling a Function)

Option Explicit On
Option Strict On

Module Module1

Sub Main()

    Dim result As Integer
    result = Cube(5)
    Console.WriteLine("The result is {0}", result)

End Sub

Private Function Cube(ByVal Number As Integer) As Integer

    Return (Number * Number * Number)

End Function

End Module
Function name: Cube()  
Description: Accepts an integer as a parameter, then calculates and returns the cube of that number.

Input: Integer value  
Output: NONE  
Preconditions: NONE  
Postconditions: No Side effect.

Private Function Cube(ByVal Number As Integer) As Integer

    Return (Number * Number * Number)

End Function
Pre and Post Conditions:

**Preconditions:**
- Indicates what is assumed to be true before the procedure or function is called.

**Postconditions:**
- Indicates the effect of the procedure or function on the rest of the program. (what should be true after the module is executed)
Review:

Reasons for using Procedures and Functions:

- Modularity (divide and conquer)
- Reusability (to eliminate repetition of code)
- Easier to manage
- Easier to understand