What do we need to learn in order to write computer programs?

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

- A method is a self-contained block of code that does something.
- They are essential for two reasons:
  - makes program development more manageable – easy to write, easy to read, easy to modify
  - Facilitates better and easier testing
  - Allow work to be divided up among different people
  - they promote code reuse If you need to do the same thing more than once, you should wrap it up into a method that you can reuse.
    - avoid repeating code in a program
Methods

- There are two types of methods in VB
  - Subroutines – methods that do not return a value
  - Functions – methods that return a value
Subroutines

- Syntax:

  ```vba
  Private Sub subroutineName([OptionalParameterList])
  [Statements]
  End Sub
  ```
Subroutine calls:

No parameter needed

Room number

House Location

Specification of work

Architectural drawing
Subroutines *without* input Parameters:

Example:

```csharp
Private Sub DisplayMessage1()
    Console.WriteLine("Hello World")
End Sub
```

Note: This subroutine has **no input parameters**.
Passing Parameters to a Subroutine

- Parameters can be sent to a subroutine either ‘____’ or ‘By____’.

  - **Pass By** is used if the sub-program has **no reason to modify** the content of the input parameter.

  - **Pass By** is used if the sub-program is **expected to modify** the content of the input parameter.
Practical Example:

```vbnet
Private Sub DisplayMessage2(ByVal Message As String)
    Console.WriteLine(Message)
End Sub

Sub Main()
    DisplayMessage2("hello") 'Pass by value
End Sub
```
Option Explicit On
Option Strict On

Module Module1

Sub Main()
    Dim Name, Age As String
    Name = "John"
    Age = "20"

    DisplayUserInfo(Name, Age) 'Pass by Value

End Sub

Private Sub DisplayUserInfo(ByVal strName As String, ByVal strAge As String)
    Console.WriteLine("Your name is: {0} ", strName)
    Console.WriteLine("Your age is: {0}", strAge)
End Sub

End Module
Passing Parameters **By Reference (Read/Write):**

```vbnet
Private Sub GetUserUserName(ByVal strName As String)
    Console.Write("Enter your name: ")
    strName = Console.ReadLine()
End Sub

Sub Main()
    Dim Name As String
    GetUserUserName(Name) 'Pass by Reference
    Console.WriteLine("Name = {0}", Name)
End Sub
```
This subroutine will read the "User’s Name", and pass it back to the calling module.

```vbnet
Private Sub GetUser_Name(ByRef strName As String)
    Console.Write("Enter your name: ")
    strName = Console.ReadLine()
End Sub

Sub Main()
    Dim Name As String
    GetUser_Name(Name)        'Pass by Reference
    Console.WriteLine("Name = {0}", Name)
End Sub
```
This subroutine will read the “**User’s Name**”, and pass it back to the calling module.

```vbnet
Private Sub GetUserName(ByRef strName As String)
    Console.Write("Enter your name: ")
    strName = Console.ReadLine()
End Sub
```

```vbnet
Sub Main()
    Dim Name As String
    GetUserName(Name)        'Pass by Reference
    Console.WriteLine("Name = {0}", Name)
End Sub
```
A complete program (Pass By Reference)

Option Explicit On
Option Strict On

Module Module1

Sub Main()

    Dim Name, Age As String
    Name = ""
    Age = ""

    GetUserInfo(Name, Age)      'Pass by Reference

    Console.WriteLine("Name = {0}, Age = {1}", Name, Age)

End Sub

Private Sub GetUserInfo(ByVal strName As String, ByVal strAge As String)
    Console.Write("Enter your name: ")
    strName = Console.ReadLine()
    Console.Write("Enter your age: ")
    strAge = Console.ReadLine()
End Sub

End Module
Option Explicit On
Option Strict On

Module Module1

Sub Main()

    Dim Name, Age As String
    Name = "......"
    Age = "......"

    GetUserInfo(Name, Age) ' What will be the result?!!!

    Console.WriteLine("Name = {0}, Age = {1}", Name, Age)

End Sub

Private Sub GetUserInfo(ByRef strName As String, ByVal strAge As String)
    Console.Write("Enter your name: ")
    strName = Console.ReadLine()
    Console.Write("Enter your age: ")
    strAge = Console.ReadLine()
End Sub

End Module
Module Module1

Sub Main()

    Dim Name, Age As String
    Name = "......"
    Age = "......"

    GetUserInfo(Name, Age)

    Console.WriteLine("Name = {0}, Age = {1}", Name, Age)

End Sub

Private Sub GetUserInfo(ByRef strName As String, ByVal strAge As String)
    Console.Write("Enter your name: ")
    strName = Console.ReadLine()
    Console.Write("Enter your age: ")
    strAge = Console.ReadLine()

End Sub

End Module
Passing Parameters – another example

Option Strict On
Option Explicit On

Module CallByRefDemo

Sub Main()
    Dim firstNumber, secondNumber As Integer
    GetNumbers(firstNumber, secondNumber)
    SwapValues(firstNumber, secondNumber)
    ShowResults(firstNumber, secondNumber)
    Console.ReadLine()  ' Just to pause the program
End Sub
Passing Parameters – another example

Private Sub GetNumbers(ByRef input1 As Integer, ByRef input2 As Integer)
    Console.Write("Please enter the first number: ")
    input1 = CInt(Console.ReadLine())
    Console.Write("Please enter the second number: ")
    input2 = CInt(Console.ReadLine())
End Sub

Private Sub SwapValues(ByRef value1 As Integer, ByRef value2 As Integer)
    Dim temp As Integer
    temp = value1
    value1 = value2
    value2 = temp
End Sub

Private Sub ShowResults(ByVal result1 As Integer, ByVal result2 As Integer)
    Console.WriteLine("In reverse order the numbers are: {0,10} {1,10}", result1, result2)
End Sub

End Module
Passing Parameters: Pitfall

- Common Mistake:
  - Declaring parameter "again" inside method:

- Compiler error results
  _____________________________
  _____________________________
  _____________________________
Parameters and Arguments

- Confusing terms, often used interchangeably
- True meanings:
  - Formal parameters
    - In method definition
  - Arguments
    - Used to "fill-in" a formal parameter
    - In method call (argument list)
  - Pass-by-value & Pass-by-reference
    - Simply the "mechanism" used in plug-in process
Event Subroutines

- Also known as **Event**

```vbnet
Private Sub BtnQuit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnQuit.Click
    Application.Exit()
End Sub
```
Functions

- Functions are similar to subroutines, but in addition to performing a specific task, functions also return a value.
- Can be explicit without using pass by reference.
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
Functions

- Syntax:

```vbnet
Private Function FunctionName([OptionalParameterList]) As DataType

[Statements]

Return (Expression)

End Function
```
Functions:

A function can do anything that a subroutine can do …

It always return a value..

- No parameter needed
- The time
- Specification of work
- Architectural drawing
- Room number
- No Room Found
- House Location
- Fire is out

+
Passing Parameters to a Function

- Parameters can be sent to a function either ‘By ?’ or ‘By ?’.

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

Private Function Cube(ByVal Number As Integer) As Integer
    Return (Number * Number * Number)
End Function

Sub Main()
    Dim CalcResult As Integer
    CalcResult = Cube(5)  ' Pass by value
End Sub
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
Scope of Declarations

- **Block scope**
  - The scope of a variable declared in a block is from the point of the declaration to the end of the block.
  
  ```
  For i as integer = 1 to 10
  ......
  Next
  ```

- **Method scope**
  - The scope of a method’s local-variable declaration or parameter is from the point at which the declaration appears to the end of that method.

- **Class scope**
  - The scope of a member that is declared in the class’s body, but outside of the bodies of the class’s method, is the entire body of the class.
  - For example, the controls in an GUI applications have class scope.
Review:

Reasons for using Subroutines and Functions:

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