Applications Programming

Procedure

Motivation

- Procedures in VBA include functions and subroutines.
- The advantages of using procedures include:
  - modularizing an application's code, which makes divide and conquer easy.
  - reducing the amount of duplicated code in a program.
- There are two types of procedures in VBA
  - Subroutine
  - Function

Subroutine

- A subroutine is a collection of statements that performs a task.
- To create a subroutine:

```
Sub ProcedureName([ParameterList])
   [Statements]
End Sub
```

- Example:

```
Sub DisplayNegErrorMsg(ByVal row As Integer)
  MsgBox "It is a negative number in row “ & row & “!”
End Sub
```

- Calling a subroutine:

```
Sub mainRoutine()
   Dim row As Integer
   For row = 1 to 100
      if Cells(row, 1) < 0 Then
         DisplayNegErrorMsg (row)
      End if
   Next row
End Sub
```

Parameter Passing

- Data can be passed into subroutines via parameters.
- Parameters are also called arguments.
- Syntax:

```
Sub ProcedureName(ByRef/ByVal var AS Type [, ParameterList])
   [Statements]
End Sub
```

- Example:

```
Sub DisplayNegErrorMsg(ByVal num As Double)
   MsgBox num & " is a negative number!"
End Sub
```

Calling Subroutine with Parameters

- Example of calling subroutine with parameters:

```
Sub mainRoutine()
   For row = 1 to 100
        if Cells(row, 1) < 0 Then
            DisplayNegErrorMsg(Cells(row,1))
        End if
    Next row
End Sub
```

- Example of passing multiple parameters:

```
Sub displaySmaller(x as Integer, y as Integer)
    If x < y Then
        MsgBox x & " is smaller."
    ElseIf x > y Then
        MsgBox y & " is smaller."
    Else
        MsgBox "They are the same."
    End If
End Sub

Sub mainRoutine()
   For row = 1 to 100
        displaySmaller(Cells(row,1), Cells(row,2))
    Next row
End Sub
```

Function

- A function returns a value to the part of the program that called the function.

- To create a function:
```
Function FunctionName([ParameterList]) As DataType
    [Statements]
    FunctionName = expression
End Function
```

- Example:
```
Function area(ByVal r As Double) As Double
    Const PI As Double = 3.14159
    area = PI * r * r
End Function
```

- Calling a function:
```
Sub mainRoutine()
    radius = InputBox("Enter the radius of a circle")
    Cells(1, 2) = area(radius)
End Sub
```

Parameter Passing

- Pass by value: This means that just the value (ie. a copy of the argument) is passed to the procedure and therefore, any changes that are made to the argument inside the procedure will be lost when the procedure is exited. For example:

  ```
  Function smallerOne(ByVal x as Integer, ByVal y as Integer) As Integer
  ```

- Pass by reference: This means that the actual address of the argument is passed to the procedure. Any changes that are made to the argument inside the procedure will be remembered when the procedure is exited. For example:

  ```
  Function getValue(ByRef x as Integer, ByRef y as Integer) As Integer
  ```

- Default type is pass by reference.

Parameter Passing

- Matching actual arguments and parameters
  - by position
  - by name

Array as Parameter

- Parameters can be singleton variables or arrays
- Use round brackets to indicate a parameter is an array parameter, for example:

```
Sub addElement(ByRef data() As Double, _
               ByRef size As Integer, _
               ByVal newData As Double)
```