Introduction to Visual Basic Application (VBA Macro Language)

Topics

- Visual Basic Application Macro Language
  - What You Can Do with VBA macro
  - Types of VBA macro
  - Recording VBA macros
    Example: MyName()
  - If-Then statement
    Example: CheckCell()
  - For-Next Loops
    Example: SquaredSum()
  - Select Case statement
    Example: CheckCell2()
  - VBA Functions
    Example: Commission (Sales)
    Example: Commission3 (Sales, Years)

What You Can Do with VBA macro

1. ______________________ a procedure that you perform frequently.
2. Create a ______________________ command.
3. Create a new ________________.
4. Create custom ________________ for Excel.

Note 1: Saving Workbooks with Macro.

Note 2: Security.
Types of VBA macro

VBA Sub Procedure

A ________________ that either the user or another macro can execute. You can have __________ number of Sub procedures in an Excel workbook.

Begins with the keyword Sub and ends with an End Sub Statement.

VBA functions

A function always returns a ________________ (just as a worksheet function always returns a single value).

Begins with the keyword Function and ends with an End Function Statement.

Recording VBA macros

1. Activate an empty cell.
2. Choose View → Macros → Record Macro. Excel displays the Record Macro dialog box.
3. Enter a new single-word name for the macro, to replace the default Macro1 name. A good name is MyName.
4. Assign this macro to the shortcut key Ctrl+Shift+N by entering uppercase N in the edit box labeled Shortcut Key.
5. Click OK to close the Record Macro dialog box.
6. Type your name into the selected Cell.
7. The macro is finished, so choose View → Macros → Stop Recording.
Introduction to Visual Basic Application (VBA Macro Language)

Examining the macro

Alt + F11

Testing the macro

Alt + F11

View → Macros → View Macros → (Select MyName) → Run

Editing the macro

Alt + F11

Insert the following statements before the End Sub statement.

ActiveCell.Font.Bold = True

ActiveCell.Columns.AutoFit

If-Then statement

If condition Then statements [Else else statements]

The following example checks the active cell. If it contains a negative value, the font color is changed to red. Otherwise, the color changes to green.

Sub CheckCell ()
    If ActiveCell.Value < 0 Then ActiveCell.Font.Color = vbRed Else ActiveCell.Font.Color = vbGreen
End Sub
**Introduction to Visual Basic Application (VBA Macro Language)**

**For-Next loops**

For-Next loop is useful to execute one or more statements a number of times.

The following example has one statement between the **For** statement and the **Next** statement. This single statement is executed 10 times.

```vba
Sub SquaredSum ()
    Total = 0
    For Num = 1 to 10
        Total = Total + (Num ^2)
    Next Num
    MsgBox Total
End Sub
```

The variable **Num** takes on successive values of 1, 2, 3, and so on, up to 10. The variable **Total** stores the sum of **Num** squared, added to the previous value of **Total**. The result is a value that represents the sum of the first 10 integers squared. This result is displayed in a message box.

**Select Case statement**

The Select Case statement is useful for choosing among two or more options.
In the following example, the active cell is checked. If the value is less than 0, it is colored red. If it’s equal to 0, it’s colored blue. If the value is greater than 0, it’s colored brown.

Sub CheckCell2()
    Select Case ActiveCell.Value
        Case Is < 0
            ActiveCell.Font.Color = vbRed
        Case 0
            ActiveCell.Font.Color = vbBlue
        Case Is > 0
            ActiveCell.Font.Color = vbGreen
    End Select
End Sub

VBA Functions

To create a custom function, follow these steps:

1. Activate the VB Editor (press Alt+F11)
2. Select the workbook in the project window.
3. Choose Insert → Module to insert a VBA Module. (Or use an existing Module).
4. Enter the keyword **Function** followed by the function’s name and a list of the arguments (if any) in parenthesis.

5. Insert the VBA code that performs the work – and make sure that the variable corresponding to the function’s name has the appropriate value when the function ends. (This is the value that the function returns.)

6. End the function with an **End Function** statement.

The following function is designed for a sales manager who needs to calculate the commissions earned by the sales force. The commission rate is based on the amount sold – those who sell more earn a higher rate. The function returns the commission amount, based on the sales made (which is the function’s only argument – a required argument). The calculations in this example are based on the following table:

<table>
<thead>
<tr>
<th>Monthly Sales</th>
<th>Commission Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - $9,999</td>
<td>8%</td>
</tr>
<tr>
<td>$10,000 - $19,999</td>
<td>10.5%</td>
</tr>
<tr>
<td>$20,000 - $39,999</td>
<td>12%</td>
</tr>
<tr>
<td>$40,000 +</td>
<td>14%</td>
</tr>
</tbody>
</table>
Function Commission (Sales)

    Tier1 = 0.08
    Tier2 = 0.105
    Tier3 = 0.12
    Tier4 = 0.14

    Select Case Sales
    Case 0 To 9999.99
        Commission = Sales * Tier1
    Case 10000 To 19999.99
        Commission = Sales * Tier2
    Case 20000 To 39999.99
        Commission = Sales * Tier3
    Case Is >= 40000
        Commission = Sales * Tier4
    End Select

    End Function
**Function with two arguments**

The following function increases commission paid by 1 percent for every year that the salesperson has been with the company. For this example, the function has been modified so that it takes two arguments, both of which are required arguments. Call this new function Commission3:

Function Commission3 (Sales, Years)

    Tier1 = 0.08
    Tier2 = 0.105
    Tier3 = 0.12
    Tier4 = 0.14

Select Case Sales

    Case 0 To 9999.99
        Commission3 = Sales * Tier1
    Case 10000 To 19999.99
        Commission3 = Sales * Tier2
    Case 20000 To 39999.99
        Commission3 = Sales * Tier3
    Case Is >= 40000
        Commission3 = Sales * Tier4

End Select
Commission3 = Commission3 + (Commission3 * Years/100)

End Function