



Advanced MS Excel 2000 for PC

Course Description: The class is a continuation of the “Intermediate MS Excel 2000 for PC” class. Topics covered include macros, user-defined functions, and simple Visual Basic procedures.

Prerequisites: familiarity with Excel 2000 and/or completion of “Intermediate MS Excel 2000 for PC”.

The W&MF staff has prepared this document for you so that can familiarize yourself with some of the more advanced features of Excel 2000. This document is meant to serve as a future reference for you - covering some options that you might find useful. Not all the information mentioned in this document will be covered in the Advanced MS Excel 2000 class.

Automating Repeated Tasks

Macros are a series of commands that Excel automatically executes. Macros are best used when there is a series of tasks that are repeatedly performed using the same sequence of actions and commands. Macros in Excel are constructed using the Visual Basic programming language; however, it is not necessary to understand computer programming to begin. Excel includes a Macro Recorder which creates the Visual Basic code. Using the Macro Recorder is good for automating simple tasks, but often enough, macros need to be edited and customized. This documentation will cover the process of recording macros and begin the basics of tailoring the macro.

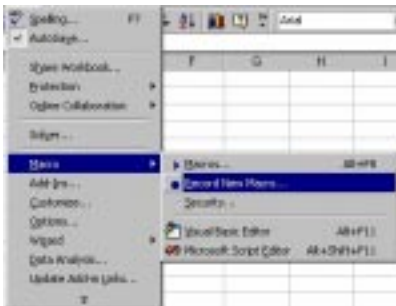
The Macro Recorder

The Macro Recorder is used to store any sequence of commands that are performed repeatedly. Once a macro has been recorded, run the macro to automatically repeat the commands that were recorded. The Macro Recorder works similar to a video camera. Just as a video camera records the performed actions, the Macro Recorder records the commands performed. Run the macro and the Macro Recorder plays back the commands just as videotape plays back the actions.

Steps to Recording a Macro

To record a macro:

1. Choose **Tools | Macro | Record New Macro**.



2. Type in a name for the macro in the **Macro name** field. The name must begin with a letter and can only contain letters, numbers, and underscores. The name cannot contain spaces or punctuation marks.
3. Type in a description for the macro in the **Description** field if desired.
4. To set options for the macro:

Create shortcut keys by setting the option in the **Shortcut key** field. To enable this option, enter in a shortcut key.

Use the **Store macro in** field to choose the location where the macro will be saved. Choosing **This Workbook** will save the macro in the current workbook. Choosing **New Workbook** will cause Excel to open a new workbook and save the macro inside. Choosing **Personal Macro Workbook** will cause Excel to save the macro in a special workbook that is always open and hidden. This option will be covered more in depth later.

5. Choose the **OK** button to start recording the macro.

The dialog box will disappear and the **Stop Macro** button will appear on the screen. In the **Status Bar**, the word **Recording** will also appear to indicate the Macro Recorder is running.

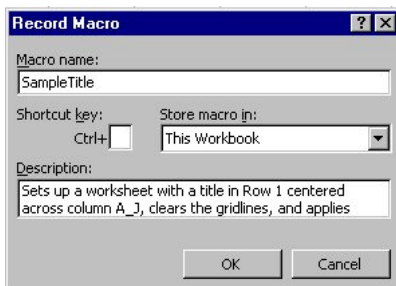
6. Perform the actions to record using keystrokes or the mouse to select menus or toolbar buttons. The Macro Recorder will record every action, mistakes included.

7. Click the **Stop Recording** button on the screen to stop the Macro Recorder.

Following is an example of a general macro that will set up a worksheet.

Begin Recording the Sample Macro:

1. Select **Tools | Macro | Record New Macro**.
2. In the **Macro name** field type "SampleTitle." Remember that the name must be one continuous word because Excel does not allow spaces.




3. In the **Description** field type "Sets up a worksheet with a title in Row 1 centered across columns A-J, clears the gridlines, and applies custom formats."

4. Choose the **OK** button.

5. The macro is now recording. From here on, all the actions performed are recorded until the macro is stopped.

Entering the title and applying custom formats:

1. Select cell A1, type "Sample Report", and then press the **Enter** key.
2. Select cell A1 again and change the font to "Times New Roman".
3. Change the font size to 16.
4. **Boldface** and underline the Title.
5. Highlight cells A1 through I1 and click on .
6. Click the Stop Recording Button.

Turning off the gridlines on the sheet:

1. Choose **Tools | Options** and click on the **View** tab.
2. Under **Window options** click on the **Gridlines** box to clear the gridlines.

xxxxx

3. Choose **OK**.

Running a Macro

After a macro has been recorded, run the macro at any time. Excel will repeat the commands inside the macro to perform the designated actions.

To run a macro:

1. Choose **Tools | Macro | Macros....**
2. In the **Macro name** field, select the macro name to run.
3. Choose the **Run** button.

Running the SampleTitle macro:

1. Switch to a new worksheet in the workbook.
2. Choose **Tools | Macro | Macros....**
3. In the **Macro** name field, select "SampleTitle."
4. Choose the **Run** button.

Making the Macro Accessible

Now that a macro has been recorded, it can be made more accessible by attaching it to graphical buttons.

Assigning a Macro to a Button on a Sheet

You can assign a macro to a button in your worksheet. When you click the button, your macro will run automatically.

1. Make sure the **Forms** toolbar is displayed.
2. On the **Forms** toolbar, click the picture of the button.
3. On the worksheet, drag the button to the size you want.
4. Select the macro you want in the **Macro name** box.
5. To assign the button to a new macro, choose the **Record** button and follow the steps outlined in **To Record a Macro** earlier in this section.

xxxxx

Removing the button or re-assigning the macro assignment:

1. Right-click on the button to select the button and view the quick menu.
2. To remove the button, select **Clear**.
3. To re-assign the button, select **Assign Macro** and select a new macro in the **Macro name / Reference** field. Click **OK** after a new macro has been selected.

Assigning a Macro to a Button on the Toolbar

You can also assign the macro to a button on the toolbar. This allows the macro to be accessible to every worksheet and workbook as long as the toolbar is displayed. If the workbook that stores the macro is not open at the time that the macro is selected, Excel will automatically open the workbook.

1. Choose **View | Toolbars**. The **Toolbar** dialog box will appear.
2. Choose the **Customize** button.
3. If the toolbar that contains the button is not visible, click the **Toolbars** tab, and then select the desired toolbar.
4. If the button you want to run the macro from is not on a toolbar, click the **Commands** tab, and then click **Macros** in the **Categories** list. In the **Commands** list, drag the **Custom** button onto a toolbar.
5. Right-click the toolbar button, and then click **Assign Macro** on the shortcut menu.

6. In the **Macro name** field, enter the name of the macro.

XXXXX

Removing a toolbar button or re-assigning the macro assignment:

1. Choose **View | Toolbars**. The **Toolbar** dialog box will appear.
2. To remove a button, click and drag the button from the toolbar into the **Toolbars** dialog box.
3. To re-assign the button, select the button and choose **Tools | Assign Macro**. Select a new macro in the **Macro name / Reference** field and select OK.

The Personal Macro Workbook

The workbook that stores the macro must always be open for the macro to be used. This makes it difficult to manage many macros used in a variety of workbooks. A solution exists for this by saving all of the macros in the Personal Macro Workbook. This workbook is hidden from view and is always opened whenever Excel is in use. Recorded macros can be saved into this Personal Macro Workbook by selecting the option in the Record New Macro dialog box that specifies a field called Personal.xls in the startup folder of the excel directory. In order to view the Personal Macro Workbook to edit or add new macros using Visual Basic, choose **Windows | Unhide** and select Personal.xls.

User-Defined Functions

User-defined functions are similar to creating a macro to improve efficiency. User-defined functions are like any other built-in Excel function such as the SUM function used in the Intermediate Excel class. They can be used to replace long formulae or even a set of formulae. To create user-defined functions, combine mathematical expressions, built-in Excel functions, and Visual Basic code in the Module worksheet of a workbook. Below is an example of a basic user-defined function:

```
Function Profit (UnitsSold, ProductionCost, SalePrice)
    Profit = UnitsSold * (SalePrice-ProductionCost)
End Function
```

Parts of a User-Defined Function

In creating a user-defined function, there are four basic parts. Every function created will begin with “Function” and end with “End Function.” These statements act as identifiers for the Excel program that mark the beginning and the end of the function. Every Function statement must be followed by a unique name that defines the function. In the previous example, the function name was Profit.

Following the function name, define the arguments / variables for the function. Each argument is a name that represents a value. The actual value can be defined in the list of arguments or it can be set to reference a cell or other user input. When defining the argument list, enclose the entire list in parentheses, separating multiple arguments with commas. Each individual argument must not contain spaces and should be names so that it reflects the value that it stores. In the example, the list contained three arguments: UnitsSold, ProductionCost, and SalePrice. Each argument is one continuous word and reflects the value that it references.

The final element in the function is the mathematical expressions. These are the actual calculations that Excel will perform. When writing formulae inside the Excel worksheet, formulae always begin with an “=” sign. This is done because Excel automatically places the calculated value in the active cell. Since the formula is now written as a function, a variable must be included before the equal sign so that Excel can store the calculated value. The general format for this is:

Variable = your mathematical expression

The expression can use any of the mathematical modifiers (+,-,*,/,^). Analyze the example where Profit=UnitsSold*(SalePrice-ProductionCost). Here the variable Profit is set to store the value of the expression. The expression multiplies the UnitsSold and the difference between the SalePrice and ProductionCost. It is important to remember that expressions do not have to be variable exclusive. It would be perfectly valid if the sample expression was

Profit = UnitsSold * 10

Creating a User-Defined Function

Writing a new user-defined function occurs in a Visual Basic module. The module exists as one of the Excel worksheets. If the module already exists as a sheet tab, access it by clicking on the tab. If the module does not exist, create a new one by choosing **Insert | Macro | Module**.

Creating a user-defined function:

1. Switch to the Visual Basic module worksheet by clicking on the sheet tab if it exists. If not, create a new one by choosing **Insert | Macro | Module**.
2. Type “Function” and the name of the function.
3. Type your argument list enclosing it within parentheses and separating arguments with commas.
4. Press Enter to move to a new line. When moving to a new line, Excel will check the previous line to ensure that the codes are correct. Visual Basic keywords such as “Function” will change to blue or a specified color. If there is an error associated with a keyword, Excel will automatically bring a dialog box on screen indicating the error.
5. Type “End Function” and press Enter.

Example:

1. Switch to the Visual Basic module worksheet by clicking on the sheet tab if it exists or create a new one by choosing **Insert | Macro | Module**.
2. Type “Function Profit” to begin the function and to name it.
3. Type the arguments list “(UnitsSold, ProductionCost, SalePrice)” and press Enter. Press the Tab key to indent the line and type in the expression “Profit=UnitsSold*(SalePrice-ProductionCost)”. Press Enter.

4. Type “End Function” and press Enter.

XXXXX

Removing a user-defined function:

1. Click on the sheet tab containing the code for the function.
2. Highlight and delete the function. Remember to delete the “Function” statement, the “End Function” statement, and any code in-between.

Using your Function

Once a function has been entered, it can be used anywhere in the workbook. The newly created function is referenced the exact same way that any other Excel function is used.

1. Select the cell to enter the function in.
2. Begin the formula by typing “=” and then type in the function name followed by an open parenthesis. Do not insert spaces between the function name and the parenthesis.
3. Type in the values for each argument. Separate multiple arguments with commas.
4. Close the argument list with a closing parenthesis.
5. Press Enter.

Example:

1. Go to any blank cell on the worksheet.
2. Type “=Profit(10,5,2)” and press Enter.
3. Excel will return the value 30.

XXXXX

What Visual Basic Is

Visual Basic is a powerful and easy programming language used in Microsoft Excel. Visual Basic can create custom commands, menus, dialog boxes, and even full-featured applications. The macros and functions that were created in the previous sections were all done using Visual Basic. The macro Recorder simplified the task by automatically generating the Visual Basic code from the actions performed. When creating the user-defined function in the previous example, it was accomplished by programming in Visual Basic. Visual Basic is an easy programming language to work with; however, its maximum extent is only unleashed through experience and practice.

Visual Basic Procedures

Everything inside Visual Basic is divided into procedures. A procedure is a block of code that is entered into a module and executed as a unit. Inside Visual basic there are three types of procedures: a Sub procedure, a Function procedure, and a Property procedure. A macro is a type of Sub procedure and a user-defined function is a type of Function procedure. In a Sub procedure, Excel performs an action but does not return a value. It is enclosed by the “Sub” and “End Sub” statements. A function procedure operates similarly to a Sub procedure but it returns a value. The user-defined function created earlier is only one type of Function procedure. A Function procedure can be made to change the Excel environment in addition to calculating and returning values. The last type of procedure, the Property procedure, is less common and will not be covered in this class. All Visual Basic code is stored in worksheets called Modules inside Excel.

Parts of a Procedure

Every procedure consists of five basic parts. When creating the user-defined function earlier, every part was used. The parts of a procedure are:

1. The “Function” and “End Function” or “Sub” and “End Sub” statements. These keywords mark the beginning and end of the procedure.
2. A name which identifies the procedure.
3. Arguments enclosed in parenthesis and separated by a comma. The arguments provide the procedure with the variables that it will work with as it executes the codes.
4. The Visual Basic code which provides Excel with the instructions that tell the procedure what to do.
5. The return value only exists in a Function procedure. A Sub procedure does not have a return value.

Working Together

When creating a procedure in Excel it is possible to have multiple procedures working together in a larger procedure. An example of this is generation of a report from raw data. Given the raw data, three separate procedures may be written. One may set up the new worksheet, another may calculate the data, and a third may plot the graph of the data. By creating a main procedure that calls upon the three sub procedures, all of the sub procedures are incorporated together.

Sub MakeReport	This is the main procedure
SetUpSheet	This is a sub procedure that sets up the worksheet
CalculateData	This is a sub procedure that calculates data
Make Graph	This is a sub procedure that makes the graph
End Sub	This ends the main procedure

Learning More about Visual Basic

The information given above is only a very slight introduction to the workings of Visual Basic. Further information falls beyond the scope of this class. To learn more, explore the variety of resources available on the World Wide Web or at the local bookstores.