Visual Basic for Finance workshop – is it for me?

These notes introduce Visual Basic for Applications and give a preview of topics and content in the Visual Basic for Finance workshop. After reading through these notes you can "self-assess" your Visual Basic knowledge on-line. Your understanding of and comfort level with these notes together with the test and its results will help you judge whether Visual Basic and the associated course are likely to meet your needs.

Introduction

Visual Basic for Applications is included in some Microsoft Office products like Excel, Word and Outlook. Visual Basic is a programming language and when you are developing Visual Basic solutions to problems you are programming. If you have programmed before (in a university course perhaps) you will know what it involves. If you haven't programmed before here is a short outline of what it is: In programming you break a large problem down into smaller parts and express those small parts in a specific and exact (and usually fairly complex) computer language. To be a good programmer one needs to be logical, precise, creative and able to understand and work with abstraction. So it is an intellectually challenging activity. It is not like a "cookbook" where you simply follow step 1, then step 2, etc. It is more like building a complex machine to perform a particular function. So, Visual Basic is not for everyone.

Like most skills it takes time to master. A two-day Visual Basic course gives you an introduction to the language and to some of the tools available and will expose you to "walk-throughs" of solving some simple problems. However, it won't make you into a Visual Basic master. Think about learning a sport: You can learn the fundamentals of tennis or skiing, say, in a couple of days but it takes sustained and serious work to become very proficient. And it's exactly the same with a technical skill like Visual Basic programming.

Walkthrough

In the next section of the notes we'll perform a walkthrough of a Visual Basic solution to a problem. The walkthrough will expose you to the look and feel of Visual Basic and give an indication of the type of thinking needed to build Visual Basic solutions.

One of the common applications of Visual Basic is in automating tasks. We'll look at an example of how to do that. We'll begin by introducing the task to be automated and then we'll show how the automation is done in Visual Basic.

Task to be automated

Each week we need to consolidate information from three separate spreadsheets. The individual spreadsheets are shown next.

Division_1.xls

	A	B
1	**Day**	**Volume**
2	Monday	23
3	Tuesday	43
4	Wednesday	56
5	Thursday	13
6	Friday	16

Division_2.xls

	A	B
1	**Day**	**Volume**
2	Monday	19
3	Tuesday	56
4	Wednesday	12
5	Thursday	17
6	Friday	8

Division_3.xls

	A	B
1	**Day**	**Volume**
2	Monday	4
3	Tuesday	6
4	Wednesday	8
5	Thursday	21
6	Friday	33

The resulting consolidated sheet should look as shown next.

	A	B	C
1	**Volume**	**Volume**	**Volume**
2	23	19	4
3	43	56	6
4	56	12	8
5	13	17	21
6	16	8	33

Each division's results have been copied and pasted into adjoining columns in the consolidated sheet.

We can automate this sequence of steps by turning on Visual Basic's macro recorder, opening a new workbook, opening each division spreadsheet in turn, copying and pasting, saving the resulting consolidated sheet and finally turning off the macro recorder. Then, when we want to redo those steps, we can simply play back the recorded macro.

Macro recording is a good solution to automation as long as we always do exactly the same thing. But often requirements change and the "recording and playing back" method becomes too inflexible. For example, suppose the number of divisions in the preceding scenario changed from 3 to 4. Our recorded macro would only process the first three divisions (because that's what we recorded). So we'd need to re-record the macro. Each time the number of divisions altered we'd need to re-do the macro. Obviously a better solution is to generalize the macro so that it works with however many divisions there are. That is what we'll do next – generalize. We begin by looking at the code that Visual Basic made when we recorded the macro. Then we'll edit the code to work with any number of divisions.

Following are the first few lines of the code that were recorded:

```
Sub Consolidate()
    Workbooks.Open Filename:="C:\tmp\example\Division_1.xlsx"
    Range("A1:B6").Select
    Selection.Copy
    Range("A1").Select
```

```
    Windows("Book3").Activate
    ActiveSheet.Paste
```

Each action we did (opening a file, selecting a cell, copying, pasting, etc) was translated into an equivalent "statement" in the Visual Basic Language.

Now we have to generalize the macro to make it work for any number of divisions. We begin by studying the macro that was recorded. The complete macro is shown next.

The recorded macro

```
Sub Consolidate()
    Workbooks.Open Filename:="C:\tmp\example\Division_1.xlsx"
    Range("B1:B6").Select
    Selection.Copy
    Windows("Book3").Activate
    Range("A1").Select
    ActiveSheet.Paste

    Workbooks.Open Filename:="C:\tmp\example\Division_2.xlsx"
    Range("B1:B6").Select
    Selection.Copy
    Windows("Book3").Activate
    Range("B1").Select
    ActiveSheet.Paste

    Workbooks.Open Filename:="C:\tmp\example\Division_3.xlsx"
    Range("B1:B6").Select
    Selection.Copy
    Windows("Book3").Activate
    Range("C1").Select
    ActiveSheet.Paste

    ActiveWindow.Close
    ActiveWorkbook.SaveAs Filename:="C:\tmp\example\Consolidated.xlsm", _
    FileFormat:=xlOpenXMLWorkbookMacroEnabled, CreateBackup:=False
    Application.WindowState = xlMinimized
End Sub
```

Editing / generalizing the recorded macro

If we look closely at the preceding code we can see that sections are repeated. In the repeated sections some parts are repeated exactly but some parts (highlighted with a shaded background) vary from section to section. We now need to edit the code and add Visual Basic statements to "loop" through however many worksheets are in the folder. The restructured code is shown below.

It uses several Visual Basic constructs: Variables, text-manipulation, looping and error handling. In the restructured code following the new and changed parts are shown with a highlighted background.

```
Sub Consolidate()
    On Error goto err_no_files_left
    number = 1
    Do
        file = "Division_" & number & ".xlsx"
        Workbooks.Open Filename:="C:\tmp\example\" & file
        Range("B1:B6").Select
        Selection.Copy
        Windows("Book3").Activate
        Range("A1").Offset(0,number).Select
        ActiveSheet.Paste
        number = number + 1
    While (True)
    err_no_files_left:
    ActiveWorkbook.SaveAs Filename:="C:\tmp\example\Consolidated.xlsm", _
    FileFormat:=xlOpenXMLWorkbookMacroEnabled, CreateBackup:=False
    Application.WindowState = xlMinimized
End Sub
```

Visual Basic features used in the above code segment include:

Looping

The “Do” and “While” statements make Visual Basic repeat the sections of code between the two statements.

Variables

In the code segment above “file” and “number” are variables. These items contain numbers or text whose values can be set and reset during the running of the code.

Text manipulation

Consider the statement "Division_" & number & ".xlsx" above. The “&’s” join two pieces of text together so that if number is 3 then "Division_" & number & ".xlsx" is equivalent to "Division_3.xlsx"

The Object Model

Consider this statement from earlier: Range("A1").Offset(0,number).Select. That statement makes Visual Basic select a cell that is offset zero rows and number columns from the cell A1. So, for example, if number is 3 then the statement has the same effect as the statement Range(“D1”).Select. In Visual Basic terminology Offset is a property of the Range object.

Error handling

The “On Error” statement near the top of the code segment tells Visual Basic to “jump” to a particular line in the code when an error occurs. In our example the error will occur when the code

has run through all of the Division worksheets and tries to open another worksheet (e.g. Division_19.xls) that doesn't exist. In other words, the error will occur when Visual Basic has processed all the sheets and needs to save the resulting consolidated workbook.

Next step – on-line self-assessment

The purpose of the preceding discussion wasn't to give exhaustive detail above Visual Basic but rather to introduce its look and feel. Having read through the preceding notes you'll probably already know whether or not Visual Basic is for you. However, if you like, you can do also a self-assessment of your Visual Basic knowledge (relating to the content of these notes). At the end of the test you will be given the option of entering your email address and receiving an emailed copy of your test results.

To go to the web page containing this document and the on-line self-assessment: http://www.tykoh.com/vbfba-intro.php

To go to the on-line self-assessment: http://www.tykoh.com/vbfba-intro-assess.php

---------oooOooo---------