Deconstructing a Visual Basic Program

Real Estate 631, S. Malpezzi April 22, 2000

The Big Picture

Let's take a look at the components of my draft of the macro for Homework 3. I have a few things to fix, but let's look at it warts and all. It does work as is. First, here's the macro in its entirety:

```
'
' test Macro
' Macro As of 2/13/00 by Stephen Malpezzi
' Analyze Multiple Cases With 1 Cash Flow Model
'
Sub test()

'erase output from previous run
 Worksheets("prob 3").Range("OUTPUT").ClearContents

'Control Program
For nnn = 1 To Worksheets("prob 3").Range("XNOBS")
    Worksheets("prob 3").Activate 'DONT FORGET WE NEED THIS
    'copy a row of input into the buffer
    Range("XCASES").Rows(nnn).Select
    Selection.Copy
    ActiveSheet.Paste destination:=Worksheets("prob 3").Range("BUFFER")

    'calculate the model
    Worksheets("prob 3").Range("MODEL").Calculate

    'copy key results to the output range
    'note: copy value, not formula!
    Range("XOBS").Select
    Selection.Copy
    Worksheets("prob 3").Range("Z11.Z121").Rows(nnn).PasteSpecial _
        Paste:=xlValues

    Range("XIRR").Select
    Selection.Copy
    Worksheets("prob 3").Range("AA11.AA121").Rows(nnn).PasteSpecial _
        Paste:=xlValues

    Range("XNPV").Select
    Selection.Copy
    Worksheets("prob 3").Range("AB11.AB121").Rows(nnn).PasteSpecial _
        Paste:=xlValues

    'beep each time through loop for the hell of it
    Beep

 Next nnn 'End loop here

 'finish up at top of output area
 Application.Goto reference:=Worksheets("prob 3").Range("Z2"), _
    scroll:=True
```

```
'tell us we're through the macro
MsgBox "Cases Successfuly Completed=" & nnn - 1
End Sub
```

Breaking it Down

Got the above? Let's look at some of the pieces. `Code in Courier font`, discussion in Times Roman *after* the code.

```
'
' test Macro
' Macro As of 2/13/99 by Stephen Malpezzi
' Analyze Multiple Cases With 1 Cash Flow Model
'
```

Note the above comments. You can't have too many clear comments!

```
Sub test()
```

We name our macro "test." Of course, we can name it whatever we want.

```
'erase output from previous run
 Worksheets("prob 3").Range("OUTPUT").ClearContents
```

Note that we use "ClearContents" as our method to delete old output. We could have used "Clear" but that would have deleted formatting of the cells as well. ClearContents deletes what's inside the cells only.

Also note that we specified named range OUTPUT as the range to delete, in Worksheet called "prob 3." Use whatever names you used for the worksheet, and range containing all of your output variables (internal rate of return, net present value, whatever).

A minor point: I was a little inconsistent with how I named ranges. I suggested to you that you name your ranges with a common prefix, e.g. ZOUTPUT, ZNOBS, ZCASES, ZGPI, etc. I used X as my prefix rather than Z, and a few (OUTPUT, BUFFER) are missing prefixes. I'll fix.

```
'Control Program
For nnn = 1 To Worksheets("prob 3").Range("XNOBS")
```

This is key. We set up a "For/Next" loop, it starts here. NNN is the index variable. From context, VBA figures out that NNN is a variable, since that's what it expects after the word FOR. Also, we could say:

For nnn = 1 to 20

but then we'd have to fix an entry in our VBA code every time we changed the number of observations. So, in the worksheet "prob 3" we have a named range called XNOBS that contains just one cell, one number: the number of cases we want to run through the model. If you followed the naming convention I suggested and used the letter Z, you would call this range ZOBS.

```
    Worksheets("prob 3").Activate 'DONT FORGET WE NEED THIS
```

Why do we need this? To tell VBA our work will be in worksheet "prob 3." Sometimes needed, sometimes not. In fact, if we ran the macro while we were *in* the worksheet, it would already be activated, and VBA wouldn't need to be told which one to activate. (Check it out: comment this line out, and run from the spreadsheet, everything works fine; run from the VBA editor and it will fail, until you un-comment it).

A point about style: notice that I indented everything between "For" and "Next" to make it very easy to see what is in between the beginning and end of the loop. Judicious use of indenting helps program readability a lot.

```
'copy a row of input into the buffer
Range("XCASES").Rows(nnn).Select
Selection.Copy
ActiveSheet.Paste destination:=Worksheets("prob 3").Range("BUFFER")
```

Pretty straightforward, by now. Notice we select the range containing all our cases (you may have called it ZCASES) and each time through the loop we choose the nnnth case. In times like this it's handy that we can address the index variable directly.

Make XCASES, ZCASES, or whatever you called it, "deep" enough to handle all the cases we might try. I'd suggest 100 rows or so for now.

You probably called the buffer ZBUFFER.

Also, note that there are several ways to write equivalent code to copy and paste that work just as well. I let the macro recorder suggest this code to me.

```
'calculate the model
Worksheets("prob 3").Range("MODEL").Calculate
```

Notice we recalculate only the range MODEL (ZMODEL?) that contains the proforma. With this particular problem you could have just written `Calculate` and been done with it, but sometimes it is handy to limit calculation to one portion or another of a spreadsheet.

```
'copy key results to the output range
'note: copy value, not formula!
Range("XOBS").Select
Selection.Copy
Worksheets("prob 3").Range("Z11.Z121").Rows(nnn).PasteSpecial _
    Paste:=xlValues

Range("XIRR").Select
Selection.Copy
Worksheets("prob 3").Range("AA11.AA121").Rows(nnn).PasteSpecial _
    Paste:=xlValues

Range("XNPV").Select
Selection.Copy
Worksheets("prob 3").Range("AB11.AB121").Rows(nnn).PasteSpecial _
    Paste:=xlValues
```

Here we're writing out three columns of output: the number of the case, the IRR, and the present value. The three columns are Z, AA, and AB (the 26th, 27th and 28th column of the spreadsheet respectively).

Shame on me, and two points off, for not using named ranges as I should have!

Notice the use of index variable nnn once more. We read the first row of input above, we write the first row of output here. Read the 20th, write the 20th. And so on.

```
    'beep each time through loop for the hell of it
    Beep
```

What can I tell you, I like to make noise when I program.

```
 Next nnn 'End loop here
```

This is where we end the loop, and add one to the index variable nnn. Then we go back to the top and see if nnn has yet exceeded the number contained in XNOBS. If no, run through the loop again. If yes, come back here and continue with the rest of the program.

```
 'finish up at top of output area
 Application.Goto reference:=Worksheets("prob 3").Range("Z2"), _
    scroll:=True
```

This is the VBA-coded equivalent of the F5 key. Notice it says "Application.Goto", which tells us Goto is an Excel thing, not a VBA thing.

```
'tell us we're through the macro
MsgBox "Cases Successfuly Completed= " & nnn - 1
```

Just a nice message confirming we're done, and telling us how many cases we've run. Notice the concatenation operator, &. Also, do you see why we use nnn-1 instead of nnn?

```
End Sub
```

We're out of here! On to the next VBA challenge!