VBA Programming in Excel/CALC: Procedural Programming and Object-oriented Programming

By Gilberto E. Urroz, November 2013

Procedural programming and Object-oriented programming

Modern electronic computers were originally developed with the purpose of solving complex numerical problems in an efficient manner. The first ever, high-level computer language, created in 1957, was FORTRAN. Its name stand for FORmula TRANslation. Thus, emphasizing the objective of solving numerical problems.

FORTRAN was the engineering language of choice from its insertion in 1957 through the 1980's, when other computer programming languages became available at a large scale, and computers became widely available. In the 1960's and 1970's, computer programming consisted in typing commands in cards or in a terminal, and obtaining results in text format. I'm going to refer to this type of programming as *procedural programming* in the sense that you code main programs, functions, and subroutines, with the purpose of processing, mainly, numerical data.

The emphasis in teaching programming to engineering students is, in most cases, procedural programming. In other words, engineering students need to learn programming with the purpose of manipulating numerical data in order to solve engineering problems. Procedural programming is the vehicle for achieving such objective. Thus, the emphasis of this course, CEE 2870, has been the teaching of basic procedural programming skills.

Since the 1990's, a new approach to programming has become popular: *object-oriented programming*. This approach is very useful in programming modern electronic computers particularly in what refers to handling elements in graphical user interfaces. Any component of a computer software, or the elements of the graphical user interface, is referred as an *object*. For example, in programming *Excel* or *CALC* with *VBA* (*V*isual *B*asic for *A*pplications), the application itself (*Excel* or *CALC*) is an object, each worksheet is an object, a range in a worksheet is an object, each cell is an object, etc.

Objects, in the context of computer programming, contain *properties* and *methods*. *Properties* are characteristics that distinguish an object from others. Thus, *Excel* or *CALC* worksheet cells have a property called *Value*, which corresponds to the numerical or string value contained in the cell. In programming *Excel* or *CALC* with *VBA* we used this property when reading a value from a cell, e.g.

```
x = Worksheets("Sheet1").Range("B2").Cells(2,1).Value
```

or, when placing a value to a cell, e.g.

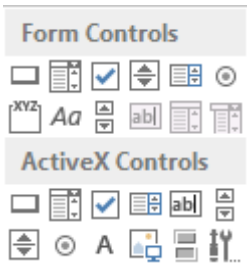
```
Worksheets("Sheet2").Range("F5").Cells(2,2).Value = y
```

Methods are programs that perform an action or calculation associated with an object. For example, In programming *Excel* or *CALC* with *VBA* we used worksheet functions to calculate numerical values, e.g.,

```
x = WorksheetFunction.Sin(2*y+z)
```


In this example, *WorksheetFunction* is a reference to a collection of functions (an object), each function in the collection being a *method* associated with that object. When invoked, as shown above, *WorksheetFunction.Sin*( ) calculates the sine of the argument. In *VBA* we used some similar pre-defined functions (*Abs, Sin, Exp*, etc.) to produce numerical calculations in our procedural programs. However, when invoking worksheet functions, as in *WorksheetFunction.Sin*( ), we were using object-oriented programming.

Thus, even though the emphasis of this course is on procedural programming, we also incorporated some object-oriented programming in our code. However, we kept object-oriented programming to a minimum. For example, out of all the objects available under the *Form Controls* or *ActiveX Controls* in *Excel* (or similar objects in *CALC*) we only used the *command button* with the purpose of activating *VBA* code in our applications. The many objects available to *Excel* are shown below:



Thus, we stayed away from using drop-down menus, check boxes, radio buttons, text boxes, etc. The incorporation of such objects in programming *Excel* or *CALC* with *VBA* would have required learning how to handle the properties and methods of such objects, and how to link them to our procedural programming code. This approach would have deviated us from the emphasis on numerical applications, i.e., away from procedural programming and into object-oriented programming.

The main reason why I developed my own set of class notes, in the form of a textbook for the class, is because I wanted to stick to the emphasis in procedural programming. Most of the textbooks that teach VBA programming emphasize object-oriented programming, and the optimal use of the graphical user interface. For example, you would find ways to add or remove worksheets in code, how to program all the objects from the *Form Controls*, how to produce a *User Form* to handle input and output, etc. I could not find a textbook that would emphasize procedural programming, and thus, I decided to write my own.

If you are interested in learning more about object-oriented programming with *VBA* in *Excel* (not *CALC*), there is a variety of textbooks out there. The way to handle objects in *CALC* is different than the way you do it in *Excel*. There aren't too many books addressing the use of controls in *CALC*, but the amount of information for the programming of *CALC* is increasing. Many examples of the use of these objects in *CALC* can be found online.

In summary, in this course we emphasized procedural programming in *Excel* or *CALC* with the purpose of learning how to solve numerical engineering problems. Some minimal object-oriented programming was included for input/output utilizing the interface, and to take advantage of pre-defined functions in the *Excel* or *CALC* application.