Passing Parameters to Functions and Sub Routines

Introduction

One of the most useful features of a procedural programming language is the ability to write and manipulate subroutines (or subs, or functions, all the same here). However, one must understand how the data moves in and out of the subroutines to fully exploit their usefulness.

The diagram above shows three levels of subroutines, typically each level performs a simpler task. This is call “Top-Down” programming and allows you to break down tasks conceptually to more simple tasks. You can then work on a smaller task then incorporate it into a larger one after you are sure it’s working.

Parameters can be passed in a variety of ways, the fundamental difference is

1. Pass by value: A copy of the number is made and passed onto the routine. The original value is left untouched
2. Pass by reference: The address of the value is passed along. Any changes to the variable will be reflected later on after control is passed back to the calling program. This is default in VBA

So passing by value is a good way to send numbers that you won’t want to bring back (like printing out those numbers or using for input for other calculations)
Passing by reference is used when you really want to change things and send them back to the calling routine

Shown below is a short, incomplete example of parameter passing

```vba
Dim x As Single, y As Single, z As Single
Dim i As Integer, j As Integer, k As Integer

Sub MainMacro()
    Dim xmm As Single, ymm As Single, zmm As Single
    Dim imm As Integer, jmm As Integer, kmm As Integer

    xmm = 10#
    ymm = 20#

    Call add2numbers(xmm, ymm, zmm)
    MsgBox ("z=" + CStr(zmm))
    imm = 5
    jmm = 6
    Call add2numbers(imm, jmm, kmm)
    MsgBox ("kmm=" + CStr(kmm))
End Sub

Sub add2numbers(xdummy, ydummy, zdummy)
    zdummy = xdummy + ydummy
End Sub
```
, and most challenging to understand