Declaring Variables and the Variant – excel programming

Declarations

The most common way to declare a variable is with the keyword Dim, but there are others such as Static, Public and Private (the latter two will be covered in a future post).

```
Dim name As String
Dim age As Integer
```

The statements above declare variables name and age as
a String and Integer respectively. It specifies what is called the type of the variable. Other types
include Byte, Long, Date, Currency, Boolean, Single, Double, Object and Variant. There are two key advantages of indicating the type of your variable to the compiler (the underlyng tool that converts your VBA into instructions you computer understands). The first is safety — if you try and write code like age = name, the compiler will give an error, as it makes no sense to try and assign text (a string) to a number. The second is performance — picking the appropriate type to represent a quantity can save space and reduce processing time. Additionally, from a readability and comprehension point of view, picking the appropriate type for a variable gives the reader clues about its intended use.

It is possible to write code as shown below without the Dim

```
Public Sub DeclaringVariablesWithoutDim()
    age = 30
    name = "John"
End Sub
```

and it will compile. However, it may not be doing quite what you expect. The absence of the Dim statement means the age and name become Variants (more on these later). Another classic expression you see is the something like this.

```
Dim first_name, last_name As String
```

The common misunderstanding is that both first_name and last_name are of type String, but in fact the omission of the type after first_name makes it

a Variant and last_name a string. Similarly, arguments to a sub or function with a type e.g. Sub ProcessPerson(first_name, last_name As String) would also make first_name a Variant.

In general you should always explicitly declare variables with one of the keywords such as Dim. There is a way of getting the compiler to prevent you omitting the declaration by adding Option Explicit at the top of any code module. I suggest you always do this: you can also turn on 'Require Variable Declaration' in the Tools-Options dialog.

Variants

So what is a Variant? It's a special type that is effectively a wrapper for multiple types in a single unit. This may sound like a general purpose utopia, but unless needed leads to possibly performance issues and possibly unexpected behaviour.

The problem is that Variants have to store within them the type of current assigned value. So if you say age=30 it will have to store the fact currently there is an Integer assigned to it. Everytime you reference a Variant, there is an additional overhead of looking up what the underlying type is, as well as additional storage requirements.

```
temperature = 3  ' no Dim, so temperature is a Variant
Debug.Print TypeName(temperature)
temperature= temperature + 0.1
Debug.Print TypeName(temperature)
temperature = "Hello"
Debug.Print TypeName(temperature)
```

The above code will print Integer, Double and then String. Going back to readability and clarity, it's generally a good idea to know what type a variable is and have it fixed — that way you can rely on the compiler picking up some of your coding mistakes.

In general then, only use Variants when absolutely necessary.

Static Declarations

For completeness, a variable declared in a function on subroutine with the keyword Static instead of Dim retains its value between calls. Look at the example below.

```
Sub TestStaticAndDim()
    StaticVariable  ' prints 0
    DimVariable     ' prints 0
```

```
    StaticVariable  ' prints 1 - the 0 was incremented to 1 in the last call
    DimVariable     ' prints 0 - although the 0 was incremented in the last
                    ' call, the value was discared when the routine ended
End Sub

Sub StaticVariable()
    Static count As Long
    Debug.Print count
    count = count + 1
End Sub

Sub DimVariable()
    Dim count As Long
    Debug.Print count
    count = count + 1
End Sub
```

Constants

This sounds like an oxymoron, but if you have a variable whose value is fixed, then you need to declare your variable as a constant using the keyword const. Not only does it tell the compiler that it's not going to change and will therefore warn you if you attempt to reassign a value to it, but also it indicates to the reader of the code that this variable is never going to change its value.

```
const numWheels As Long = 4
' or if you prefer
const NUM_WHEELS As Long = 4
```

Where to Declare

Back in the day, it was not unusual for some languages to declare all the variables you wish to use in a function (or sub) at the top of it. My preference is to declare a variable as close to its first use as possible. This I feel increases readability.

The only thing to be aware of if you are familiar with other languages is that if you declare a variable inside a block (do, for, if etc) it is still valid (in scope) after then end of the block. It is as if it was declared above the block.

```
If isAboveThreshold Then
    Dim scopeTest As Long
```

```
    scopeTest = 123
End If
Debug.Print scopeTest    'prints 123, still in scope
```