Naming and Naming Conventions - Excel project

Naming Is Hard

Before we get into Naming Conventions, let's talk about naming at a more general level, devoid of complexities of prefixes and capitalisation.

Picking appropriate names for the likes of variables, constants, functions, classes etc is not easy. In fact, a well-known (to some) quote from Phil Karlton is "There are only two hard problems in Computer Science: cache invalidation and naming things."

The fundamental reason to name carefully is to help the coder and those maintaining the code to understand what your program is doing. The time taken to cut corners here is simply wasted later down the line when trying to figure out what some cryptic name actually means. Remember that code is read far more often than it is written.

Firstly, there is no benefit having short names, convoluted names or names containing abbreviations that can only be understood by the original author. The only exceptions I think are valid for very short names are i when used in a small and straightforward loop as this use is so common, and xand y when used in relation to a co-ordinate system. Their use is so prevalent that everybody knows what they are. There are no performance benefits of using short names.

Names should be meaningful and aid the reader. Functions called DoIt() aren't useful. Don't be afraid of long names either, but don't go mad. Careful thought can yield meaningful names that are reasonably short too. Look at the names below for a function that gets the mean spending average for a year. Which name conveys to a reader what the function really does? Hopefully one of the last two will be your choice.

```
GetAverage(year As Integer) As Double
GetSpendingAverage(year As Integer) As Double
GetSpendingAverageForYear(year As Integer) As Double
GetSpendingMeanAverageForYear(year As Integer) As Double
GetMeanSpendingForYear(year As Integer) As Double
```

Try also not to have similarly spelled or sounding names that have totally different meanings. Oh, and whilst we are at it, variables randomly inserted with names

like fancySomeBeer may amuse you as you create them, but aren't appreciated when the next person is trying to fix your code just before a client deadline. I don't think adding digits to names generally helps either such as name1 and name2, unless there is really some logic to it and not just lazy naming.

Human nature being what it is, we all like to abbreviate things. As long as the abbreviations are well known and used carefully and consistently I can't see too much of a problem, so long as you don't go crazy and you take in to account potential readers of the code who may not be so versed in the same technical lingo as you. One of my pet hates is removing just one or two letters from a word to abbreviate it, for example using GetSpnd instead of GetSpend, or cellClr instead of cellColor. That last example brings up a another question — VBA objects and libraries use American English, so personally for consistency I adapt my spellings to match, so I would name a variable borderColor rather than borderColour as it is likely to be applied to a property of an object such a Range.Borders.Color and you end of with a mix.

Context is important — for example if you are writing code relating to weather, then tmp and temp could be useful contractions of temperature but for many people tmp and temp are common abbreviations for 'temporary'. So perhaps using the full word temperature is better. Just don't go for tmpTemp meaning a temporary temperature. If you do go for a contraction, ensure you use it for one meaning only throughout your code and don't mix it up as below.

```
GetAverageTemperature() As Double
GetAvRainfall() As Double ' abbreviating Average to Av
```

Renaming is just as important as naming. With search and replace, taking the time to rename really isn't much effort. Even if you have been careful in selecting names, as your code develops you may well find what was fairly obvious when you first named it, is confusing now that extra functionality has been added. For example, you have a routine called DrawChart which draws a chart of costs; you then add functionality by adding routines called DrawSpendChart and DrawBreakevenChart. Now all of a sudden the name DrawChart isn't so meaningful. You would have to study the code in detail to see what it did. Taking the time to rename to DrawCostChart is going to make the code more comprehensible and that's what one should strive for.

I also like to have functions and sub routines start with a verb, such as GetMeanSpend or ColorCellsAccordingToValue. Functions that return

a Boolean should generally start with ‘Is’ or perhaps ‘Does’ with the following words in the name giving no doubt as to the true or false state to be returned, for example IsCellBackgroundRed().

Naming Conventions

So what is a naming convention? Well basically it’s a set of rules that dictate how you name variables of differnt types, classes, modules, user defined types, references to user interface elements such as buttons and list boxes etc. The idea behind a naming convention is to improve readability and understandability by creating consistency and hopefully leading to a reduction in errors.

The exact naming convention to use is a hot topic where opinions are strongly held and loudly voiced. I’ve seen and used a fair few over the years and I can honestly say, that by far and away the most important rule is consistency. Use one convention and stick with it. If you are editing existing code and it’s not your favourite, then just grit your teeth and stick with it.

‘Use one convention and stick with it.’

What happens if you don’t have, or don’t use a convention? In that case, you need something and below I will give a few pointers to get going. My views are very much my own and I know many will cringe or stick pins in a little effigy of me because they don’t agree — that’s fine, all I want you to do is find something that has some logic behind it and that can be followed easily and stuck to. It’s easy these days to search around on the Internet and find something that appeals to you.

What’s in a name

There are generally three main elements to a naming convention and they are

- the type (Integer, Boolean etc)
- capitalisation
- scope or how visibile a variable/function is to other parts of your code

The type and scope are usually determined by a prefix.

Type Prefix

It was at one stage very common to prefix variables with some kind of indicator as to the underlying type of a variable, such as bIsSpendAboveThreshold for

a Boolean or dblTemperature for a Double. Personally, I don't think this is necessary and just makes the majority of variables more unreadable. That is assuming you declare all your variables upfront (you should).

However, it is very common in VBA code to see prefixes representing the type of an object, particularly a user interface component such as a textbox or button. For example, a prefix txt and cmd in front of a variable name referencing a text box or command button respectively. If you continue to adopt this practice, then please use the well-established prefixes listed in the VB6 Object Naming Conventions. Note this is now archived and no longer maintained and also applies to VB6 rather than VBA, but it's close enough.

Capitalisation

My preference here is to use something called Camel Case where you capitalise the first letter of each word. For functions, sub routines, classes this would include the first letter of the name. For local variables that contain no prefix, then I prefer to start with a lower-case letter and then capitalise each word, as in the example function below.

```
Private Function GetAreaOfCircle(radius As Double) As Double
    Const PI As Double = 3.142
    Dim areaOfCircle As Double
    areaOfCircle = radius * radius * PI
    GetAreaOfCircle = areaOfCircle
End Function
```

For constants (those variables defined using the Const keyword), I prefer all capitals, with each word separated by an underscore, such as MAXIMUM_SPEND.

Scope

Here we are talking about some form of naming (usually a prefix) to indicate the scope or visibility of a variable. For example, a variable defined inside a function cannot be accessed outside of it: it's scope is local to the function. As this is the 'normal' case, I wouldn't advocate any specific naming for local variable to indicate scope.

If however, you have a variable defined outside a function, such as at the top of a module (or perhaps Worksheet/Workbook in Excel), that variable can be accessed and modified by multiple routines, and hence is much harder to know when and where it is modified. Indicating to the reader that this kind of behaviour is possible with a prefix is

a useful clear visual clue. I would tend to use something like m or m_, meaning member or module depending on your preferred terminology.

A future post will cover in more detail the differences between variables defined in various ways at the top of modules and classes
using Dim, Public, Private and Global and the importance of minimising the scope of a variable. It will also contain some more examples of naming variables.