

# Programming Style



Each programmer has their own style of writing. Good programming style makes the code easy to understand, not only for other people but also the programmer himself. There are several components of good style:

* [Naming Conventions](#gjdgxs) (variables, controls, and subroutine names)
* [Indented and properly spaced code](#30j0zll)
* [Documented code](#1fob9te)
* [Cohesive Modules](#3znysh7)
* [Minimum Coupling](#2et92p0)
* [Minimize data scope where possible](#tyjcwt)

## ***Naming Conventions***

By default, when placing new controls on a form, Visual Basic will name them Commmand1, Command2, Command3... (or label1, text1, etc). A large application might need to use more than 30 different command buttons in total, but without meaningful names it is difficult for the programmer to remember what command28 is for. Thus, as soon as a new control is added to a project it is good practice to rename it. As well as controls, variables and subroutines should also be given names which the programmer can identify easily. Not only are meaningful names encouraged, but also Microsoft encourages programmers to use various *prefixes* to identify the type of object. The first table below lists the prefixes for variable data types and the second table lists prefixes for control types.

| | Variable Data Type | Prefix | Example | | --- | --- | --- | | Boolean | bln | blnFull | | Byte | byt | bytDaysInMonth | | Currency | cur | curPoundsSterling | | Date (Time) | dtm | dtmStart | | Double | dbl | dblAstronomicalDistances | | Integer | int | intNumberOfEmployees | | Long | lng | lngProfits | | Single | sgl | sgl | | String | str | strSurname | | User-Defined Type | udt | udtStaffDetails | | Variant | vnt | vntChartData | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

| | Control Type | Prefix | Example | | --- | --- | --- | | Animated button | ani | aniEmptying | | Check box | chk | chkWriteOnly | | Combo box | cbo | cboLanguage | | Command button | cmd | cmdCancel | | Common Dialog | dlg | dlgSave | | Data Contol | dat | datStock | | Data-bound Combo | dbcbo | dbcboActicleType | | Data-bound Grid | dbgrd | dbgrdStockItems | | Data-bound List box | dblst | dblstAccountCodes | | Directory list box | dir | dirTarget | | Drive list box | drv | drvSource | | File list box | fil | filPick | | Form | frm | frmMainMenu | | Frame | fra | frmPrinters | | Graph | gra | graSharePrices | | Grid | grd | grdQuantities | | Horizontal Scroll Bar | hsb | hsbHueColor | | Image | img | imgBitMap | | Label | lbl | lblHelpUser | | List Box | lst | lstColorCodes | | MCI | mci | mciSoundEffects | | MDI Child Form | mdi | mdiJuly | | Menu | mnu | mnuFileOpen | | MS Tab | mst | mstDays | | OLE | ole | oleExcel | | Picture Box | pic | picMemoryLeft | | ProgressBar | prg | prgConverting | | Report | rpt | rptEndofYear | | RichTextBox | rtf | rtfDiary | | Shape | shp | shpSquare | | Slider | sld | sldWindSpeed | | Spin button | spn | spnTicketsRequired | | StatusBar | sta | staInformation | | Text Box | txt | txtInputText | | Timer | tmr | tmrStartAlarmCount | | Vertical Scroll Bar | vsb | vsbRatios | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Note:** Control names are used for referring to specific controls at run time, do not confuse these with control *properties* such as ‘text’ and ‘caption’.

## ***Indented and properly spaced code***

Indenting is used to visually improve the layout of a program. Indenting shows which lines are subordinate to other lines. For example, all the lines which make up the body of a loop should be indented. In the following example there is a nested repeating structure (one inside another). Everything inside the first FOR loop (week\_no) is indented and then everything inside the second loop (day\_no) is indented further. Again each CASE statement is indented between the containing Select Case and End Select keywords.

**For bytWeekNo = 1 To 52  
 Print "Week #"; bytWeekNo  
 For bytDayNo = 1 To 7  
 Select Case bytDayNo  
 Case 1  
 Print "Sunday"  
 Case 2  
 Print "Monday"  
 Case 3  
 Print "Tuesday"  
 Case 4  
 Print "Wednesday"  
 Case 5  
 Print "Thursday"  
 Case 6  
 Print "Friday"  
 Case 7  
 Print "Saturday"  
 End Select  
 Next bytDayNo  
 Next bytWeekNo**

Blank lines can also be inserted between lines of code to break a procedure up into smaller blocks of code.

## ***Documented Code***

The first comments to be entered into a Visual Basic program should be a section describing the overall features of the current application. The **(general) (declarations)** section on the first or main form is a good place for these comments. Information such as the author, date written, any known bugs and the purpose of the application should be listed. This makes the job of any other programmers looking at your code so much easier.

Example:

**' Author : Simon Wilkinson  
 ' Date : 14.04.1998  
 ' Version : 1.1  
 ' Language : MS Visual Basic 5.0  
 ' Purpose : Demonstrates how to create a grid, set fixed  
 ' rows and columns, set individual cells, and  
 ' add and delete rows.  
 ' Known Bugs : N/A**

The above example documents the Grid program example which can be downloaded from the [menu](http://docs.google.com/dvbhome.html). It should be noted that the ‘Language’ line is useful not only to document which version of Visual Basic is being used, but also so that the documentation for all programs (written in any language) can be standardised.

As well as a main comment block at the start of a program, comments should also document complicated or unusual sections of code. Ideally each variable and array should be commented where it is defined so that its function can be easily understood later. There are two ways to document code in VB, the Rem statement and the '. Remarks (Rem) must be on a separate line, but a ' followed by some text can be added to the end of any line. The ' tells VB to ignore all words from this point to the end of the line.

Example:

**Dim intSheepNo As Integer ' Holds the number of black-face sheep**

Example:

**Rem Centralise the form.  
 frmCurrent.Left = (Screen.Width / 2) - (frmCurrent.Width / 2)  
 frmCurrent.Top = (Screen.Height / 2) - (frmCurrent.Height / 2)**

## ***Cohesive Proceedures***

Each procedure should be designed to complete a single task. If a procedure handles too many tasks then its logic can be hard to comprehend and bugs can easily occur.

## ***Minimize Coupling***

Parameter passing is good programming practice, but too many parameters and the code again becomes difficult to manage. Proceedures with many parameters are highly coupled, they have lots of links to other procedures. Where possible try to minimize these links. However, global variables should not be used instead!

## ***Minimize data scope where possible***

Varaibles and arrays can be accessed by code in different parts of the program. Accessable anywhere is Global. This sounds ideal, but without care strange side effects can occur where code in other parts of a program ‘mess about with’ the contents of a variable by mistake. Restricting the access range, or scope, of a variable or array can solve this problem. An intermediate scope is access to a variable throughout a single module. This means any procedure in the module can access the variable, but procedures in other modules are denied access. And the most restricted scope is local. These are variables and arrays which can only be accessed from within the subroutine in which they were defined; even other subroutines within the same module cannot access these variables.

Where possible limit scope as much as possible. For more information about this please read [variable scope](http://docs.google.com/scope.html).

 [Menu](http://docs.google.com/dvbhome.html)