Tutorial: Programming in Visual Basic 6.0

This tutorial contains a beginner's guide to Visual Basic 6.0, introducing the programming environment, defining key terms and introducing exercises to demonstrate the five control structures (sequence, selection: binary and multiway, iteration: pre and post test).

Syllabus outcome

H5.3 A student selects and applies appropriate software to facilitate the design and development of software solutions

Students learn to implement a fully tested and documented software solution in a methodical manner. (SDD syllabus, p.51)

The following headings may help you navigate:

Organising your first project

The first step is to create a project template within VB, to organise and store your work. This will consist of a menu structure with headings that will let you access the many exercises and examples you complete.

Activity 1

- Open VisualBasic 6.0
- Use the file menu to open a new project with a blank form.
- Use the **properties window** to set
  - Main.frm as the form name.
  - My programs as the caption.
  - BackColor to White.
  - BorderStyle to Fixed Single.
  - WindowState to Maximised.
- Find the Menu icon and click on it to select it. Enter the following menu headings:

Quit
Introduction with indented subheadings of
Example1
Example2

Click OK after each menu caption and name are typed.

- Click on Quit menu heading and enter the following code. This procedure is used to exit from running the project display and return to the design screens.

```
Private Sub Quit_Click()

    Unload me
    End

End Sub
```

- Use the <F5> function key to run the application to verify that the Menu structure is correct and that the Quit procedure is free from error.
- Use the File menu to save your work as **Main.frm** and ***(your intitials)*Project1.vbp**
- Use the file menu to open a new blank form (or the properties window)
- Set the following form properties:
  form name as **Welcome**
  caption to **Example1**
  BackColor to White
  BorderStyle to Fixed Single
  WindowState to Maximised
- Click on the Example 1 main menu heading and enter the following code:

```
Private Sub Example1_Click()
    Welcome.Show
End Sub
```

- Save your work and run <F5> to ensure that it is free of errors.

- Add two labels, an image and a command button to create a welcome screen. To do this

Select label icon from the toolbox. Click towards the centre-top of your form and position and resize your label as required.
With the label selected, use the properties window to
Change the caption to WELCOME TO VISUAL BASIC
Choose a bright back colour
Set the font (Arial, underline, alignment centred, size 24 point, forecolour blue)
Repeat to add the *Enjoy* label.
Use the image icon on your toolbox to add the image to your form. Use the properties window of the image to select a picture.
Use the command Button icon to add the button. Change its caption to RETURN. Then double-click the button and add the following line of code after the Command1_Click() procedure.

- `Unload Welcome`

- Use the file menu to save your work and use <F5> to run the application.

- DON'T FORGET TO SAVE (AND BACK UP TO FLOPPY) ALL YOUR WORK.

Event handlers and scroll bars

Some definitions to learn

- An **object** is a thing — anything that appears on the screen. An object has a set of **properties**. Each property of the object has a **value**.
  e.g. Label.Caption = "Welcome to Visual Basic" where

  Label is an object
  Caption is a property of label
  "Welcome to Visual Basic" is a value of the Caption property.

- **Events** are things that happen on the screen. **Event handlers** transfer data to **procedures** that complete the task. The results of these procedures are returned back to other screen objects, e.g. onChange onClick

- A **procedure** is a group of statements designed to perform a specific task. A procedure attached to an object, such as a button, is a **command** used to make something happen, e.g.

```
Public Sub Command2_Click()
        Text1.Text = "This is a procedure."
End Sub
```

Add new form to menu

As each new example and exercise solution is to be added to your project you will need to:

- add a new form
- set the form properties using the properties window
- click on the main menu icon with the main form displayed to show the menu designer
- add a new menu heading
- click on the menu heading to show the procedure code
- to the procedure code, add the statement

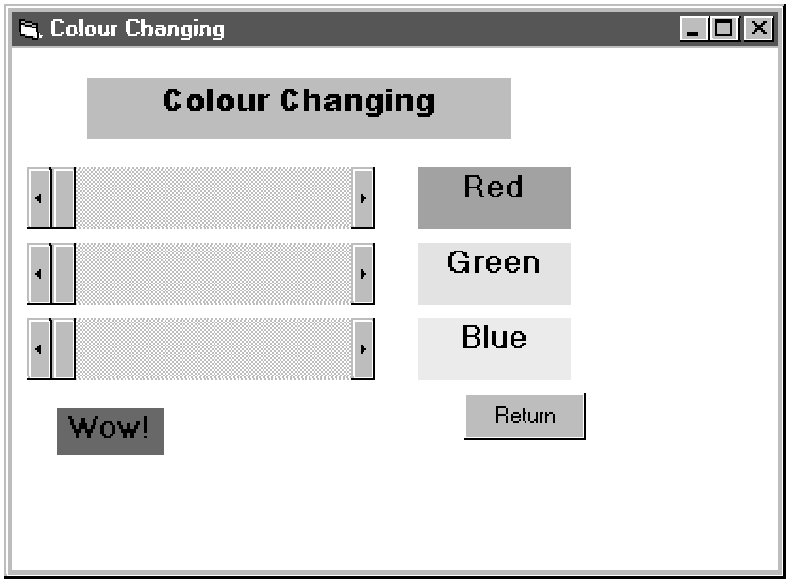
  `FormXX.Show`

  where FormXX is the new form name.

Activity 2

1. Open a new form and change its name to ColourChanger. Place the following objects on this form.

- A heading label2 (Caption = Colour Changer)
- 3 horizontal scroll bars (Set the max value property of all three to 255)
- 3 other labels (2red, 3Green, 4Blue)
- a command button to quit the form (Caption = Return)
- another small label5 under the button with its visible property set to false.



2. Double click each scroll bar and add the following code to its _onChange() event. Use cut and paste to make the task easier.

```
Label1.BackColor = RGB(HScroll1.Value, HScroll2.Value, HScroll3.Value)
Label5.BackColor = RGB(HScroll1.Value, HScroll2.Value, HScroll3.Value)

Label1.ForeColor = RGB(255 - HScroll1.Value, 255 - HScroll2.Value, 255 -
HScroll3.Value)
Label5.ForeColor = RGB(255 - HScroll1.Value, 255 - HScroll2.Value, 255 -
HScroll3.Value)

Label5.Visible = True
Label5.Caption = “WOW!”

Label2.BackColor = RGB(HScroll1.Value, 0, 0)
Label3.BackColor = RGB(0, HScroll2.Value, 0)
Label4.BackColor = RGB)0, 0, HScroll3.Value)
```

3. Double click the return button and add the following code to its _onClick() event

```
Unload Me
```

4. Use the Project Explorer window to return to your main form and double click example 2 in your menu to add the appropriate code.

5. Use <F5> function key to test your project. Save and backup.

Naming conventions

Up till now, we have often accepted default names, Text1, Label1, etc. In a big project, this is not good practice as it makes the code harder to read or maintain. Naming conventions use a prefix of three lower**Case** letters to identify the type of control, followed by a meaningful name. eg. lblTitle

Prefix Abbreviations for Control names

Prefix	Control	Prefix	Control
cbo	combo box	chk	check box
cmd	command button	dir	directory list box
drv	drive list box	fil	file list box
fil	file list box	fra	frame
frm	form	grd	grid
hsb	horizontal scrollbar	img	image
lbl	label	lin	line
lst	list box	mnu	menu
ole	OLE client	opt	option button
pic	picture box	shp	shape
tmr	timer	txt	text box
vsb	vertical scrollbar		

Data types in VB

A **variable** is a named location that holds data. A variable can only hold one datatype. A program can have as many variables as you need but before you can use a variable it must be declared.

You use a **DIM** statement to declare variables (where DIM stands for dimension). Here is the format of the DIM statement:

Dim VarName As Datatype

e.g. Dim curCost As Currency, Dim strSurname As String

Datatype	Description and Range	Prefix
Boolean	One of two values only. e.g. True or False e.g. blnIsOverTime	bln
Byte	Positive numeric values without decimals from 0-256 e.g. bytAge	byt
Currency	Data that holds dollar amounts from -$922,337,203,685,477.5808 to +- $922,337,203,685,477.5807 e.g. curHourlyPay	cur
Date	Date and time values from Jan 1, 100 to Dec 31, 9999 e.g. dteFirstLesson	dte
Double	Numeric values from –1.79769313486232E+308 to +1.79769313486232E+308. Often called double-precision. e.g. dblMicroMeasurement	dbl
Integer	Numeric values with no decimal point or fraction from –32,768 to 32,767 e.g. intCount	int
Long	Integer values beyond the range of Integer datatype from	lng

	−2,147,483,648 to 2,147,483,647 e.g. lngStarDistance	
Object	A special datatype that holds and references objects such as controls or forms. e.g. objSoundClip	obj
Single	Numeric values that range from −3,402823E+38 to 3,402823E+38. Often called single-precision. sngYearSalesFigures	sng
String	Data that consists of 0 to 65,400 characters of alphanumeric data including special characters such as @, ^, ½ e.g. strFirstName	str
Variant	Data of any datatype used for control and other values for which the datatype is unknown. e.g. vntControlValue	vnt or var

A **function** is a segment of code that accepts zero, one or more arguments and returns a single result. Visual Basic includes many built-in functions (intrinsic functions). Some perform basic mathematical tasks. Others manipulate string data such as converting text to upper**Case** or lower**Case** letters.

An **argument** is a value you pass to a function so the function has data to work with.

Function names have parentheses at the end to hold the function arguments. Even if a function has no arguments, the parenthesis are required.

Two intrinsic functions include message boxes and input boxes.

Activity 3: Message and input boxes

Message and input boxes are intrinsic functions in Visual Basic 6.0 which allow the end user to interact with the program.

Follow the instructions Add new form to menu at the end of Activity 1 to create a new form with a menu heading on the main form. Call this "Message and Input Boxes"

- Make the Form.Caption = "Message and Input Boxes"
- Put a label on the top of the form "Computer Conversation". Underneath have a command button with the caption "Talk to me!" Name the command button cmdTalk.
- Double click the command button to add the following code sequence.

```
Private Sub cmdTalk_Click()
     Dim strQuestion As String    'First you must declare your variables'
     Dim intAnswer As Integer

     'Then use the input and message box functions'

     strQuestion = InputBox("Type in your name!", "Ebeneezer")
     intAnswer = MsgBox("Hello there" & strQuestion, vbOKCancel, "Chat")

End Sub
```

- Add a return button, called cmdBack as you did in the ColourChanger, using the code

```
Private Sub cmdBack_Click()
     Form1.Show
End Sub
```

- Run your program using <F5>. Don't forget to save your work.

Here are some handy **literals** (values that don't change). You don't have to learn them as the help prompt supplies a drop down list as you start to type.

Buttons in Message Boxes

Named Literal	Value	Description
vbOKOnly	0	Displays the OK button
vbOKCancel	1	Displays the OK button and Cancel buttons
vbAbortRetryIgnore	2	Displays the Abort, Retry and Ignore buttons.
vbYesNoCancel	3	Displays the Yes, No and Cancel buttons.
vbYesNo	4	Displays the Yes and No buttons.
vbRetryCancel	5	Displays the Retry and Cancel buttons.

Icons in Message Boxes

Named literal	Value	Description
vbCritical	16	Displays Critical Message icon
vbQuestion	32	Displays Warning Query icon.
vbExclamation	48	Displays Warning Message icon.
vbInformation	64	Displays Information message icon.
vbSystemModal	4096	Displays a System Modal dialog box. The user must acknowledge this box before doing anything else.

Remarks are added in code to explain the purpose of a section of code or to add information for code maintenance. If Rem or ` is placed in front of the remark, that line of code is ignored completely.

Activity 4

Create a calculator that can add, subtract, multiply and divide two numbers given by the user.
[A possible solution might use two input boxes (txtOne and txtTwo) and a label to display the answer (lblAnswer).

Declare variables:

```
dblNo1 As Double
dblNo2 As Double
dblAnswer As Double
intError As Integer
```

Use Val function to change string from input box to a number, then an assignment statement to put that value into the variable.

```
dblNo1 =  Val (txtOne.text)
```

Repeat for second number.

Use a Format function to ensure answer is rounded off to two decimal places.

```
lblAnswer.Caption = Format (dblAnswer, "#,##0.00")
```

If you are very clever, this might be an option for the user.

Ensure that it is not possible to divide by zero, either by entering nothing or by entering zero. Use the MsgBox() function to indicate the problem to the user.

```
If Val (txtTwo.Text) = 0  Then
     IntError = MsgBox ("You cannot divide by 0!", VbOkCancel, "Whoops!)
Else …

End If
```

Add a **clear** command button with the following code to allow the user to do another calculation.

```
txtOne.Text = ""
txtTwo.Text = ""
lblAnswer.Caption = ""
txtOne.SetFocus
```

The SetFocus method returns the cursor to the first input box.

Set properties to pretty it up.
Check that it works. Use integers, very big numbers, very small numbers, negative numbers, zeros, etc. Is your label big enough for all values? If you set the label's autosize property to true it will stretch to fit.

Add a **remark** (put ` at the beginning of the line) at the top of your code which includes your name and the date.

Connect to a *Binary Selection* menu heading on the main form.

Sequence algorithms

The programs in Activities 1 — 3 were all constructed from **sequence** algorithm constructs. Each line of code followed another with only one possible pathway for each event. So, for each **sub procedure,** the algorithm would consist of input, output and a series of process steps, e.g.

```
Private Sub cmdClear_Click()  `user input
    txtOne.Text = ""          `sequence of processes initializing variables
    txtTwo.Text = ""
    lblAnswer.Caption = ""
    txtOne.SetFocus
End Sub                       `output
```

Binary selection

The next group of programs you will write uses the second algorithm construct — **selection.** Selection allows multiple pathways for any event and allows for choices to be made. Selection constructs can be **Binary** (two way) or **Multiway** (multiple choices)

Binary selection uses the `If – End If` or the `If – Else – End If` statements. Here is the syntax in Visual Basic.

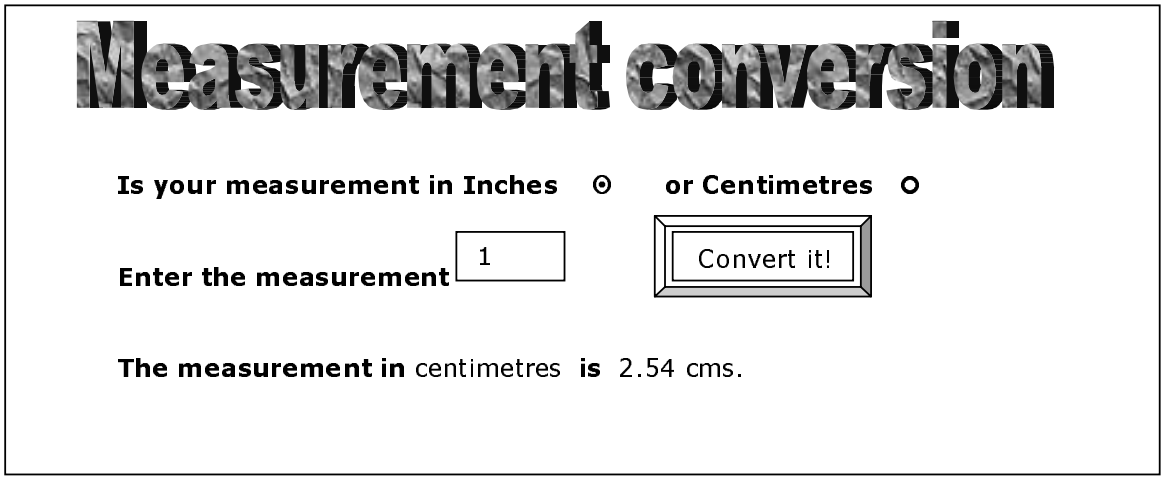
If *comparison test* **Then**
    One or more Visual Basic statements
End If

OR

If *comparison test* **Then**
    One or more Visual Basic statements
Else
    One or more Visual Basic Statements
End If

(You have used binary selection in your calculator to prevent a user dividing by zero.)

Activity 5



- Add a new menu heading *Selections* with two subheadings, *Binary Selection* and *Multiway Selection.*
- Write a program to convert inches to centimetres OR centimetres to inches (using the conversion 1 inch = 2.54 centimetres).
- Use option buttons (from the toolbox) for the user to indicate whether the conversion is inches to centimetres or centimetres to inches.
- Use IF statements to determine which formula to use based on which option button is selected. **Option buttons** are mutually exclusive, i.e. only one can be selected at a time.
- Connect to menu heading Binary Selection.
- Run the application to ensure that it is working correctly.
- Use your calculator to verify the results. Try it out with some **test data** including very large numbers, very small numbers, zero, negative numbers, 0.000000987654.

Multiway selection

In Activity 5 we looked at an example of binary selection. If the selection involves more than two alternatives, you can use nested If statements but this becomes complicated and leads to hard-to-read code. It is better to use **Case** statements. Here is the syntax for multiple selection through **Case** statements.

Select Case *Expression*
    Case *value*
        [One or more VB statements]
    Case *value*
        [One or more VB statements]
    Case *value*
        [One or more VB statements]
    Case Else
        [One or more VB statements]
End Select

For example:

Select Case intAge

```
        Case Is < 6
            lblTitle.Caption = "Preschool"
        Case 6 To 11
            lblTitle.Caption = "Primary School"
        Case 12 To 18
            lblTitle.Caption = "Secondary School"
        Case Else
            lblTitle.Caption = "Adult"
    End Select
```

Activity 6

1. The post office has the following charges for parcels based upon different weights.

Weight (gram)	Cost
0 – 50	$1.40
51–100	$2.70
101–250	$4.00
251–500	$7.50

Parcels which are heavier than 500 gms are calculated by weight*0.02

Design a project that allows a user to enter the weight in a text box and calculate the postage. Use **Case** statements in your code. Link this as CaseWeights under the Multiway menu heading in your main form.

2. Use a set of check boxes to allow a user to choose the noise level by the comments, then output the probable decibel level based on information in the following table.

Decibel level	Comfort level
140+	Extremely painful
90 –139	Deafening
60 – 89	Disturbing
30 – 59	Distracting
0 – 29	Relaxing

Again use **Case** statements and link this to the main form menu under the name NoiseLevels again connected to the **Multiway** menu heading.

Activity 7

A **control array** is a set of multiple controls of the same type with the same name (often created by using the Copy and Paste command). You may have encountered this already in adding radio buttons or check boxes to your form. Individual controls within the array are distinguished by having different Index property values. So, if you created a control array of option buttons called optChoice, the Click event procedure might look like this:

```
Private Sub optChoice_Click (Index As Integer)
    Select Case optChoice (Index)
            Case Index = 0        Label1.Caption = "Monday"

            Case Index = 1        Label1.Caption = "Tuesday"

            Case Index = 2        Label1.Caption = "Wednesday"

            Case Index = 3        Label1.Caption = "Thursday"
```

```
            Case Index = 4        Label1.Caption = "Friday"

            Case Index = 5        Label1.Caption = "Saturday"

            Case Else             Label1.Caption = "Sunday"

    End Select
End Sub
```

The code above would change the label caption as each different option button was selected. Try this out, then add code to change the label background colour (to something appropriate) for each different day. Link to the menu Multiway Selection heading in the main form menu under the heading Colour My Days.

Before we continue with the last structures — **iterations** — check that the menu headings on Main are all correct and linked (by code) to the correct forms. Check the list below and change any that need to be changed.

Menu
Quit
Introduction (Sequences)
    Example 1 *(– Welcome to VB)* *from Activity 1*
    Example 2 *(– Colour changer)* *from Activity 2*
Message and Input boxes
    Sequence *(– Computer Conversation)* *from Activity 3*
Selection
    Binary *(- The calculator)* *from Activity 4*
    Binary *(- Measurement Converter)* *from Activity 5*
    Multiway *(- Parcel weights)* *from Activity 6*
    Multiway *(- Noise levels)* *from Activity 6*
    Multiway *(-Colour my days)* *from Activity 7*
Iteration

Iterations

Iterations or loops are structures that allow a statement or group of statements to be carried out repeatedly while some condition remains true (or for a counted number of times). **Each iteration MUST contain a way of stopping the looping.** There are 2 basic iteration structures:

- **Pre-test iterations:** in these loops, the condition to be met occurs at the beginning of the loop and ***the code will not run at all if the condition is never met***.

- **Post-test iterations:** in these loops, the condition to be met is at the end of the loop so that ***the code always runs at least once***.

Activity 8

Create a new form and link to iteration — pre-test (- Count the beeps) on the Main menu. Add a text box and a command button. Put a label above the text box asking the user to input a number between 1 and 10. When the user clicks the command button, check the textbox for a valid number and issue an error message if the number isn't inside the expected range. If the number is valid, use **Do While ..... Loop** to issue that number of beeps. Name your form frmBeeper.

```
Private Sub Command1_Click()
    Dim intNumber As Integer                          'Declare variables
    Dim pause As Integer

    intNumber = Val(Text1.Text)                       'Input user number

    If (intNumber > 0) And (intNumber < 11) Then      'Validate number
        Do While intNumber > 0
            Beep
                For pause = 0 To 2500                 'Slow down the beeps
                 frmBeeper.Refresh
                Next pause
            intNumber = intNumber - 1                 'Count down
        Loop
    Else
        Text1.Text = ""
        Text1.SetFocus
    End If
End Sub
```

Write an IPO Chart for this program.

Then write the pseudocode algorithm for this program.

Activity 9

In this tutorial, we look at writing code containing the 5 basic control structures.

- sequence
- binary selection
- multiway selection
- pre-test iteration
- post-test iteration

The IPO chart and pseudocode algorithm for finding the average of some numbers entered from the keyboard, using a post-test loop, might look like this:

I	number, counter
P	add number to sum add 1 to counter average = sum/counter
O	average

```
BEGIN Main Program
    counter = 0
    sum = 0
    REPEAT
          get number
          sum = sum + number
          counter = counter + 1
    UNTIL counter = 10
    average = sum / counter
    display average
END MAINPROGRAM
```

Write this algorithm as a flowchart.

Open a new form and create a link to the program from the menu form (Main) under Iterations … Post-test Loops … Averages. Write an application to display the average of 10 numbers entered by the user using a post-test loop with the syntax `Do …..Loop Until` to ask for each number. Remember to use the `Val ()` function to convert strings to numbers. Each number might be asked for with an input box using the syntax:

```
strNumber = InputBox("Enter the next number.", "Enter your numbers.")
intNumber = Val(strNumber)
```

Don't forget to declare all your variables with Dim statements.
e.g. `Dim strNumber As String`

Initialise your variables to 0 at the beginning of the procedure.

```
intNumber = 0
intCounter = 0
```

Can you be sure that the counter will never cause division by zero and crash your program? Explain your answer.

Activity 10

Use a new function Rnd() **to generate a random number**. Write an application on a new form to generate a random number between 1 and 6 to simulate the rolling of a dice. Your form will need a large picture box with the Autosize property set to true and a command button with the following code behind it.

```
Private Sub Command1_Click()
    Dim x As Integer
    x = 0

    Picture1 Print

    Randomize
    ' Ensures the start of each sequence of random numbers is also randomised

    Do
        x = Int(Rnd * 6 + 1)
        Picture1 Print x;
    Loop Until x <> 6
End Sub
```

Run your program clicking the command button several times (until the picture box is full). Create a link to the program from the menu form (Main) under Iterations … Post test Loops … Roll the dice.

To produce a random integer x where 0 <= x < N, use the following syntax

```
x = Int(Rnd * N)
```

The following statement produces random integers in the range from 51 to 150.

```
x = Int (Rnd*100 + 51)
```

The Randomize statement ensures that the start of each sequence of random numbers is also random.

You will notice that Visual Basic has two different syntax statements for post-test iterations:

- `Do ……. Loop Until` *(comparison test)*
- `Do ……. Loop While` *(comparison test)*

Both are post-test loops where the comparison test appears at the bottom of the loop meaning that the code in the loop must execute at least once. These are both equivalent to the pseudocode syntax

REPEAT ……. UNTIL

Write the algorithm for this application in pseudocode that explains what every line in the program does.

Activity 11

Create a program to generate the first 20 Fibonacci numbers. This time use a counter to control the number of iterations. Add a Picture Box to print your results to. Make sure the PictureBox.Font Transparent = False, AutoRedraw = True and that the BackColor is different from the ForeColor. Use the following code to help you.

```
Dim FibNumber As Integer, FibNext As Integer, Counter As Integer

'Initialise
FibNumber = 0
FibNext = 1

Picture1 Print "Ist 20 Fibonacci Numbers"

Do While Counter < 20
     Picture1 Print FibNumber & ", ";
     'trailing semicolon stops the Print method going to the next line.
     Picture1 Print FibNext & ", "
     FibNumber = FibNumber + FibNext
     FibNext = FibNext + FibNumber
     Counter = Counter + 2
     If Counter = 10 Then
          Picture1 Print
          'This creates a new line
     End If
Loop
```

Write an IPO chart and the algorithm in pseudocode and as a flowchart.

Activity 12

Write a program that uses nested **For** loops to fill a 2 dimensional array and then to print out the times tables from 1 – 12 into a picture control using the syntax:

picDisplay **Print** (variable, [variable] ...]

with a new line for each new times table on the display.

You will definitely need to plot this one out on paper first, writing your algorithms and checking them. To declare your variable, you need the statement:

Dim arrTables[12,12] **As** Integer

Bibliography

Lynch, I (1999). *An introduction to Visual Basic.* Mansfield, QLD: Lynformation

Further Resources

Kerman, M. and Brown, R. (2000). *Computer programming fundamentals with applications in Visual Basic 6.0.* Reading, Massachusetts: Addison-Wesley.

searchVB.com, The VB Specific Portal and Search Engine presented by TechTarget.com: http://www.searchvb.com/

Free-Ed Net Course Catalog: http://www.free-ed.net/catalog.htm

This work was prepared by

Beverley Sampford