Visual Basic 6 Made Easy

Dr. Liew Voon Kiong
Disclaimer

Visual Basic ® Made Easy- A Complete Tutorial for Beginners is an independent publication and is not affiliated with, nor has it been authorized, sponsored, or otherwise approved by Microsoft Corporation.

Trademarks

Microsoft, Visual Basic, Excel, Access and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks belong to their respective owners.

Liability

The purpose of this book is to provide a basic guide for people interested in Visual Basic® programming. Although every effort and care has been taken to make the information as accurate as possible, the author shall not be liable for any errors as well as harm or damage arising from using the instructions given in this book by any person.
Acknowledgement

I would like to express my sincere gratitude to many people who have made their contributions in one way or another to the successful publication of this book. My special thanks go to my children Xiang, Yi and Xun. My daughter Xiang edited this book while my sons Yi and Xun contributed their ideas and even wrote some of the sample programs for this book. I would also like to appreciate the support provided by my beloved wife Kim Huang and my youngest daughter Yuan. I would also like to thank the millions of visitors to my Visual Basic Tutorial Website at www.vbtutor.net; especially those who contributed their comments, for their support and encouragement.

About the Author

Dr. Liew Voon Kiong holds a bachelor's degree in mathematics, a master's degree in management and a doctoral degree in business administration. He has been involved in programming for more than 30 years. He created the popular online Visual Basic Tutorial at www.vbtutor.net in 1996 and since then the web site has attracted millions of visitors and it is one of the top
searched Visual Basic Tutorial website in many search engines including Google. To provide more support for the Visual Basic hobbyists, he has written this book based on the Visual Basic tutorial.

Contents

CHAPTER 1 INTRODUCTION TO VISUAL BASIC 14

1.1 A brief description of Visual Basic 14
1.2 The Visual Basic Environment 15

CHAPTER 2 BUILDING VISUAL BASIC APPLICATIONS 19

2.1 Creating Simple Visual Basic Applications 19
   Example 2.1 Print a Message 22
   Example 2.2 Arithmetic Calculation 23
   Example 2.3 Arithmetic Calculation 24
   Example 2.4(a) Using + 25
   Example 2.4(b) Using & 25
2.2 Steps in Building a Visual Basic Application 26
   Example 2.5 Calculate the Volume of a Cylinder 26
   Example 2.6 Designing a Calculator UI 27

CHAPTER 3 WORKING WITH CONTROLS 31

3.1 The Properties of the Controls 31
3.2 Handling some of the common controls 34
   3.2.1 Text Box 34
Example 3.1 Adding Numbers in Two Text Boxes 34
3.2.2 Label 35
3.2.3 Command Button 36
3.2.4 The Picture Box 36
3.2.5 The Image Box 36
3.2.6 The List Box 37
Example 3.2 Adding Items to a List Box 37
3.2.7 The Combo Box 37
Example 3.3 Adding Items to a Combo Box 38
3.2.8 The Check Box 38
Example 3.4 Using Check Boxes 38
3.2.9 The Option Button 39
Example 3.5 Controlling Shapes with Option Buttons 39
3.2.10 The Drive List Box 40
3.2.11 The Directory List Box 41
3.2.12 The File List Box 41

CHAPTER 4 WRITING THE CODES 44

Example 4.1 Using Label and Text Box 45
Example 4.2 Using Image Control 45
Example 4.3 Using Picture Box and Timer 46
Example 4.4 Text Concatenation 47
Example 4.5 Addition using Val 47

CHAPTER 5 VISUAL BASIC DATA 50
CHAPTER 8 CONTROLLING PROGRAM FLOW – PART II

70

8.1 The Select Case … End Select Statement 70

Example 8.1 To compute Examination Grades 70
Example 8.2 Examination Grades 72
Example 8.3 Examination Results 73
Example 8.4 Calculator 74

CHAPTER 9 LOOPING PART I 77

9.1 Do… Loop 77

Example 9.1 A Counter 77
Example 9.2 Loop Until 78
Example 9.3 Do Until 79
Example 9.4 Loop While 79
Example 9.5 Summation 79

9.2 Exiting the Do Loop 81

Example 9.6 Exit Do 82

9.3 The While…Wend Loop 83

Example 9.7 Populating a List Box with While Wend Loop 83

CHAPTER 10 LOOPING PART II 86

10.1 The For…Next Loop 86

Example 10.1 A Counter 86
Example 10.2 Step Increment 87
Example 10.3 Decreasing Step 87
Example 10.4 Printing a Message Repetitively 88

10.2 Exiting the For…Next Loop 88

Example 10.5 Exit For 89

10.3 The Nested Loop 90

Example 10.6 Nested Loop 91

CHAPTER 11 BUILT-IN FUNCTIONS 94

11.1 Introduction to Visual Basic Functions 94

11.2 The MsgBox ( ) Function 94

Example 11.1 Message Box 96

Example 11.2 Message Box with an Alert Icon 98

11.3 The InputBox ( ) Function 99

Example 11.3 Using an Input Box 100

CHAPTER 12 MATH FUNCTIONS 102

12.1 The Rnd Function 102

Example 12.1 Generating Random Numbers 102

Example 12.2 Generating Random Integers 103

Example 12.3: A password generator 105

12.2 The Numeric Functions 106

Example 12.4 Rounding Numbers 107

12.3 Trigonometric Functions 108

Example 12.5 Computing Trigonometric Values 109

CHAPTER 13 FORMATTING FUNCTIONS 112

13.1 The Tab Function 112
Example 13.1 Using Tab to Generate Multiplication Table 112

Example 13.2 Multiplication Table 113

13.2 The Space function 115

Example 13.3 Generating Multiplication Table Using Space Function 115

Example 13.3 Generating Multiplication Table Using Space Function 115

Example 13.4 Formatting Numbers using Format Function 118

Example 13.5 121

Example 13.5 121

13.4 Formatting Date and Time 122

Example 13.6 Displaying Date and Time in Different Formats 122

Example 13.7 Displaying Time and Date in Various Formats 125

CHAPTER 14 STRINGS FUNCTIONS 128

14.1 The Len Function 128

14.2 The Right Function 128

14.3 The Left Function 129

14.4 The LTrim Function 129

14.5 The RTrim Function 129

14.6 The Trim function 130

14.7 The Mid Function 130

14.8 The InStr function 130

14.9 The Ucase and the Lcase functions 131
14.10 **The Str and Val Functions** 131
14.11 **The Chr and the Asc Functions** 131

*Example 14.1 Using Various String Functions* 132

**CHAPTER 15 CREATING YOUR OWN FUNCTIONS** 134

*Example 15.1 A Function that Calculate Area of Triangle* 134
*Example 15.2 A Function that Calculate Profit Margin* 135
*Example 15.3 Future Value Calculator* 137
*Example 15.4 Examination Grades Generator* 138
*Example 15.5 BMI Calculator* 140

**CHAPTER 16 CREATING FUNCTIONS FOR MS EXCEL** 143

16.1 **Creating User-Defined Functions in MS-Excel** 143
16.2 **Using Microsoft Excel Visual Basic Editor** 144

**CHAPTER 17 ARRAYS** 147

17.1 **Introduction to Array** 147
17.2 **Declaring Arrays** 147

*Example 17.1 Creating an Array of Names using Input Box* 149

*Example 17.2 Displaying an Array of Names using a List Box* 150

17.3 **The Control Array** 151

**CHAPTER 18 TWO DIMENSIONAL ARRAYS** 153

*Example 18.1 Two-Dimensional Array* 154
CHAPTER 19 GRAPHICS

19.1 The line and Shape controls

Example 19.1 Selecting Shapes

19.2 The Image Box and the Picture Box

Example 19.2 Loading Images Randomly

19.3 PSet, Line and Circle Drawing Methods

Example 19.3 Graphics Drawing Tool

Example 19.4 The Picture Viewer

Example 19.5 An Advanced Picture Viewer

CHAPTER 20 CREATING MULTIMEDIA APPLICATIONS

178

20.1 Creating a CD Player

20.2 Creating an Audio Player

20.3 Creating a Multimedia Player

CHAPTER 21 ANIMATION-PART I

196

Example 21.1 Moving Jet

Example 21.2 Moving Jet Using Commands

Example 21.3 Moving Objects

Example 21.4 Zoom in and Out

CHAPTER 22 ANIMATION - PART II

205

22.1 Animation using a drag and drop procedure.

22.2 Animation with complete motion
Example 27.1 Creating a Text File 312

27.3 Reading a File 312

Example 27.2 Reading a text file 313
Example 27.3 A simple database management system 313
Example 27.4 Creating and Reading Text File with Common Dialog 317

CHAPTER 28 CREATING DATABASE APPLICATIONS 320

Example 28.1 Simple Database Application 320
Example 28.2 Manipulating Database 322

CHAPTER 29 CREATING DATABASE APPLICATIONS 325

WITH ADO CONTROL 325

Example 29.1 An ADO Application 325

CHAPTER 30 MANAGING DATABASE WITH DATAGRID CONTROL 334

CHAPTER 31 SQL QUERIES 341

31.1 The SELECT Keyword 341
31.2 The WHERE Keyword 346

Example 31.1 Query based on Author 347
Example 31.2: Query based on year 347
Example 31.3 350
Example 31.4 350
CHAPTER 37 CREATING REPORTS

37.1 A BRIEF INTRODUCTION TO REPORTING TOOL IN VB6 395
37.2 STEPS IN BUILDING YOUR REPORT IN VISUAL BASIC 6 395

   Step 1: Adding Data Report 395
   Step 2: Connecting the report to database using Data Environment Designer 396
   Step 3: Retrieving Information from the Database 399

CHAPTER 38 INTERNET AND WEB APPLICATIONS 404

38.1 THE WEB BROWSER 404
38.2 THE FTP PROGRAM 409

INDEX 420
Chapter 1 Introduction to Visual Basic

❖ A brief description of Visual Basic
❖ Getting to know the Visual Basic environment

1.1 A brief description of Visual Basic

VISUAL BASIC is a high-level programming language evolved from the earlier DOS version called BASIC. BASIC stands for Beginners' All-purpose Symbolic Instruction Code. The program codes in Visual Basic resembles the English language. Different software companies produce different versions of BASIC, such as Microsoft QBASIC, QUICKBASIC, GWBASIC, and IBM BASICA and so on.

Visual Basic is an easy programming language to learn and it is for anybody who is interested in programming but lack professional training in software engineering. Learning VB will help young children to improve their logical thinking skills and develop their minds. You can choose to program in VB purely for fun and enjoyment or you can create more advanced applications such as educational courseware and commercial software.
VISUAL BASIC is a VISUAL and events driven Programming Language. These are the main divergences from the old BASIC. In BASIC, programming is done in a text-based environment and the program is executed sequentially. In VISUAL BASIC, programming is done in a graphical environment. In old BASIC, you have to write a text-based procedure to design the interface, but Visual Basic enables you to design the interface by dragging and resizing the objects as well as changing their colors, just like any windows-based programs.

Visual Basic is event-driven because users may click on a certain object randomly, so each object has to be programmed independently to be able to response to those actions (events). Examples of events are clicking a command button, entering text into a text box, selecting an item in a list box etc. Therefore, a VISUAL BASIC Program is made up of many subprograms; each with its own program code which can be executed independently and at the same time can be linked together in one way or another.

1.2 The Visual Basic Environment

Upon start up, Visual Basic 6.0 will display the dialog box as shown in Figure 1.1. You can choose to start a new project, open
an existing project, or select a list of recently opened programs. A project is a collection of files that make up your application. There are various types of applications that can be created; however, we shall concentrate on creating Standard EXE programs (EXE means executable program). Now, click on the Standard EXE icon to go into the VB programming environment, as shown in Figure 1.1.

Figure 1.1 Visual Basic 6 Start-up Dialog Box
In Figure 1.2, the Visual Basic Environment consists of

- A blank form for you to design your application's interface.
- The project window which displays the files that are created in your application.
The properties window which displays the properties of various controls and objects that are created in your application.

It also has a Toolbox that consists of all the controls essential for developing a VB Application. The controls include text boxes, command buttons, labels, combo boxes, picture boxes, image boxes, timers and other objects that can be dragged to and drawn on the form to perform certain tasks according to the events assigned to them. You may also add additional objects. First, click on the project item on the menu, then on the components item on the drop-down list, and lastly select the controls you want to use in your program. The controls for Standard.EXE window is shown in Figure 1.3.
Exercise 1

1. Write down the meaning of BASIC.
2. Briefly explain the difference between VISUAL BASIC and the old BASIC.
3. List out all the controls in a Standard.exe Visual Basic project.
Chapter 2 Building Visual Basic Applications

❖ Creating simple Visual Basic Applications.
❖ Getting to know the steps in building a Visual Basic Application.

2.1 Creating Simple Visual Basic Applications

In this section, we are not going into the technical aspects of VB programming; the purpose of this section is to let you get a feel for it. Now, you can try out a few examples in this chapter. Example 2.1 is a simple program. First, you have to launch Microsoft Visual Basic. Normally, a default form Form1 will be available for you to start your new project. Double click on Form1, and the source code window as shown in Figure 2.1 will appear. The top of the source code window consists of a list of objects (on the left) and their associated events or procedures (on the right). In Figure 2.1, the object displayed is Form and the associated procedure is Load.
Figure 2.1 The Source Code Window

When you click on the object box, the drop-down list will display a list of objects you have inserted into your form as shown in Figure 2.2. Here, you can see the form, the command button with the name Command1, the Label with the name Label1 and the PictureBox with the name Picture1. Similarly, when you click on the procedure box, a list of procedures associated with the object will be displayed as shown in Figure 2.3. Some of the procedures associated with the object Form are Activate, Click, DblClick (Double-Click), DragDrop, keyPress
etc. Each object has its own set of procedures. You can select an object and write codes for any of its procedures to perform specific tasks.

Figure 2.2 List of Objects
Figure 2.3 List of Procedures

You do not have to worry about the beginning and the end statements (i.e. Private Sub Form_Load......End Sub.); Just key in the codes between the above two statements exactly as shown here. When you run the program, don’t be surprised that nothing shows up. In order to display the output of the program, you have to add the Form1.show statement like in Example 2.1 or you can just use the Form_Activate ( ) event procedure as shown in example 2.2. The command Print does not mean printing using a printer. Instead, it means displaying the output on the computer screen. Now, press F5 or click on the run button to run the program and you will get the output as shown in Figure 2.4.
Example 2.1 Print a Message

Private Sub Form_Load()
    Form1.Show
    Print "Welcome to Visual Basic tutorial"
End Sub

Figure 2.4 Out of Example 2.1

You can also perform simple arithmetic calculations as shown in example 2.2. VB uses * to denote the multiplication operator and / to denote the division operator. The output is shown in Figure 2.5, where the results are arranged vertically.
Example 2.2 Arithmetic Calculation

Private Sub Form_Activate ( )
    Print 20 + 10
    Print 20 - 10
    Print 20 * 10
    Print 20 / 10
End Sub

Figure 2.5 Output of Example 2.2

Example 2.2 can also be written as

Private Sub Form_Activate ( )
    Print 20 + 10, 20 - 10, 20 * 10, 20 / 10
End Sub

The numbers will be arranged in a horizontal line separated by spaces as shown in Figure 2.6

![Figure 2.6 Output in a Horizontal Line](image)

Example 2.3 is an improved version of example 2.2 as it employs two variables x and y and assigns initial values of 20 and 10 to them, respectively. When you need to change the values of x and y, just change the initial values rather than changing every individual value which is more time consuming.

**Example 2.3 Arithmetic Calculation**

Private Sub Form_Activate ( )
x = 20
y = 10
Print x + y
Print x - y
Print x * y
Print x / y

End Sub

Besides, you can also use the + or the & operator to join two or more texts (string) together like in example 2.4 (a) and (b)

**Example 2.4(a) Using +**

Private Sub Form_Activate ()

A = "Tom 
B = "likes 
C = "to 
D = "eat 
E = "burgers."
Print A + B + C + D + E

End Sub

**Example 2.4(b) Using &**

Private Sub Form_Activate ()

A = "Tom 


B = "likes "
C = "to "
D = "eat "
E = "burgers."

Print A & B & C & D & E
End Sub

The output is as shown in Figure 2.7

Figure 2.7

2.2 Steps in Building a Visual Basic Application
There are three basic steps in building a VB application, as follows:

Step 1: Design the interface

Step 2: Set the properties of the controls (Objects)

Step 3: Write the events' procedures

**Example 2.5 Calculate the Volume of a Cylinder**

This program is a simple program that calculates the volume of a cylinder.

![Figure 2.8 A Program to Calculate Volume of a Cylinder](image)

First of all, go to the properties window and change the form caption to Volume of Cylinder, then drag and insert three labels
into the form and change their captions to Base Radius, Height and Volume respectively. After that, insert three text boxes and clear its text contents so that you get three empty boxes. Name the text boxes as radius, hght and volume, respectively. Lastly, insert a command button and change its caption to O.K and its name to OK. Now save the project as cylinder.vbp and the form as cylinder.frm. We shall leave out the codes now which you shall learn in the following chapters.

**Example 2.6 Designing a Calculator UI**

Designing an attractive and user-friendly interface is the first step in constructing a VB program. To illustrate this point, let us look at the calculator program.
Now, follow the steps below to design the calculator interface.

- Resize the form until you get the size you are satisfied with.
- Go to the properties window and change the default caption of the form to the caption you like, such as Calculator.
• Change the other properties of the form, such as background color, foreground color, and border style. For this program, I recommend you set the following properties for Form1:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BorderStyle</td>
<td>Fixed Single</td>
</tr>
<tr>
<td>MaxButton</td>
<td>False</td>
</tr>
<tr>
<td>minButton</td>
<td>True</td>
</tr>
</tbody>
</table>

These properties will ensure that the users cannot resize or maximize your calculator window, but able to minimize the window.

• Draw the Display Panel by clicking on the Label button and place your mouse on the form. Start drawing by pressing down your mouse button and dragging it.

• Click on the panel and the corresponding properties window will appear. Clear the default label so that the caption is blank. It is better to set the background color to a brighter color while the foreground color should be of a darker color like black (for easy viewing). Change
the name to display as I am going to use it later to write codes for the calculator.

- Now draw the command buttons that are necessary to operate a calculator. I suggest you follow exactly what is shown in the image above.

Now run the project by pressing F5. If you are satisfied with the appearance, go ahead, and save the project.

Exercise 2

1. Write down the list of procedures which are associated with the Form object.

2. Write a program to display the sentence "I like Visual Basic".

3. Write a program to compute the value of
   \[1500+1000-450*10+300\div5\]

4. Using a value of \(x=9\) and a value of \(y=13\), write a program to compute the values of \(2x+y\), \((x+y)\div2\) and \(x^2y\)
Chapter 3 Working with Controls

❖ Setting the properties of the controls
❖ Learning how to work with the controls

3.1 The Properties of the Controls

Before writing an event procedure for the control to response to a user's input, you have to set certain properties for the control to determine its appearance and how it will work with the event procedure. You can set the properties of the controls in the properties window or at runtime.

Figure 3.1 is a typical property window for a form. You can rename the form caption to any name that you like best. In the properties window, the item which appears at the top part is the object currently selected (in Figure 3.1, the object selected is Form1). At the bottom part, the items listed in the left column represent the names of various properties associated with the selected object while the items listed in the right column represent the states of the properties. Properties can be set by highlighting the items in the right column and then changing them by typing or selecting the options available.
For example, to change the caption just highlight Form1 under the name Caption and change it to other names. You may also alter the appearance of the form by setting it to 3D or flat. You can do other things like changing the foreground and background color, changing the font type and font size, enabling or disabling the minimize and maximize buttons and etc.
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Name)</td>
<td>Form1</td>
</tr>
<tr>
<td>Appearance</td>
<td>1 - 3D</td>
</tr>
<tr>
<td>AutoRedraw</td>
<td>False</td>
</tr>
<tr>
<td>BackColor</td>
<td>#H8000000</td>
</tr>
<tr>
<td>BorderStyle</td>
<td>2 - Sizable</td>
</tr>
<tr>
<td><strong>Caption</strong></td>
<td><strong>Form1</strong></td>
</tr>
<tr>
<td>ClipControls</td>
<td>True</td>
</tr>
<tr>
<td>ControlBox</td>
<td>True</td>
</tr>
<tr>
<td>DrawMode</td>
<td>13 - Copy Per</td>
</tr>
<tr>
<td>DrawStyle</td>
<td>0 - Solid</td>
</tr>
<tr>
<td>DrawWidth</td>
<td>1</td>
</tr>
<tr>
<td>Enabled</td>
<td>True</td>
</tr>
<tr>
<td>FillColor</td>
<td>#H8000000</td>
</tr>
<tr>
<td>FillStyle</td>
<td>1 - Transpare</td>
</tr>
<tr>
<td>Font</td>
<td>MS Sans Serif</td>
</tr>
<tr>
<td>FontTranspare</td>
<td>True</td>
</tr>
<tr>
<td>ForeColor</td>
<td>#H8000000</td>
</tr>
<tr>
<td>HasDC</td>
<td>True</td>
</tr>
<tr>
<td>Height</td>
<td>3600</td>
</tr>
<tr>
<td>HelpContextID</td>
<td>0</td>
</tr>
<tr>
<td>Icon</td>
<td>(Icon)</td>
</tr>
<tr>
<td>KeyPreview</td>
<td>False</td>
</tr>
<tr>
<td>Left</td>
<td>0</td>
</tr>
<tr>
<td>LinkMode</td>
<td>0 - None</td>
</tr>
<tr>
<td>LinkTopic</td>
<td>Form1</td>
</tr>
<tr>
<td>MaxButton</td>
<td>True</td>
</tr>
<tr>
<td>MDIChild</td>
<td>False</td>
</tr>
<tr>
<td>MinButton</td>
<td>True</td>
</tr>
<tr>
<td>MouseIcon</td>
<td>(None)</td>
</tr>
<tr>
<td>MousePointer</td>
<td>0 - Default</td>
</tr>
<tr>
<td>Moveable</td>
<td>True</td>
</tr>
<tr>
<td>NegotiateMenu</td>
<td>True</td>
</tr>
<tr>
<td>OLEDropMode</td>
<td>0 - None</td>
</tr>
<tr>
<td>Palette</td>
<td>(None)</td>
</tr>
<tr>
<td>PaletteMode</td>
<td>0 - Halftone</td>
</tr>
</tbody>
</table>

**Caption**
Returns/sets the text displayed in an object's title bar or below an
Figure 3.1

You can also change the properties at runtime to create special effects such as changing of colors, shape, animation effects and so on. For example, the following code will change the form color to red every time the form is loaded. VB uses hexadecimal system to represent colors. You can check the color codes in the properties windows which are shown under ForeColor and BackColor.

Private Sub Form_Load()
    Form1.Show
    Form1.BackColor = &H000000FF&
End Sub

Another example is to change the control Shape to a shape at runtime by writing the following code. This code will change the shape to a circle at runtime. Later you will learn how to change the shapes randomly by using the RND function.

Private Sub Form_Load()
    Shape1.Shape = 3
End Sub
I would like to stress that learning how and when to set the objects' properties is very important as it can help you to write a good program. Therefore, I advise you to spend a lot of time playing with the objects' properties. I am not going into the details on how to set the properties. However, I would like to stress a few important points about setting up the properties.

- You should set the Caption Property of a control clearly so that the user knows what to do with that command. For example, in the calculator program, as all the captions of the command buttons such as +, -, MC, MR are commonly found in an ordinary calculator, the user should have no problem in manipulating the buttons.

- A lot of programmers like to use a meaningful name for the Name Property because it is easier for them to write and read the event procedure and easier to debug or modify the programs later. However, it is not a must to do that as long as you label your objects clearly and use comments in the program whenever you feel necessary.

- One more important property is whether the control is being enabled or not.

- Finally, you must also consider making the control visible or invisible at runtime, or when should it become visible or invisible.
3.2 Handling some of the common controls

3.2.1 Text Box

The text box is the standard control that is used to receive input from the user as well as to display the output. It can handle string (text) and numeric data but not images or pictures. Strings in a text box can be converted to numeric data by using the function `Val(text)`. The following example illustrates a simple program that processes the inputs from the user.

**Example 3.1 Adding Numbers in Two Text Boxes**

In this program, two text boxes are inserted into the form together with a few labels. The two text boxes are used to accept inputs from the user and one of the labels will be used to display the sum of two numbers that are entered into the two text boxes. A command button is also programmed to calculate the sum of the two numbers using the plus operator. The program creates a variable, ‘sum’, to accept the summation of values from Textbox 1 and text box 2. The procedure to calculate and display the output on the label is shown below. The output is shown in Figure 3.1.

```vbnet
Private Sub Command1_Click()
```
'To add the values in Textbox 1 and text box 2

Sum = Val(Text1.Text) + Val(Text2.Text)

'To display the answer on label 1

Label1.Caption = Sum

End Sub

Figure 3.1

3.2.2 Label

The label is used to provide instructions and guides to the users, it can also be used to display outputs. One of its most important properties is **Caption**. Using the syntax **Label.Caption**, it can display text and numeric data. You can change its caption in the
properties window and also at runtime. Please refer to Example 3.1 and Figure 3.1 for the usage of Label.

### 3.2.3 Command Button

The command button is used to execute commands. It displays an illusion that the button is pressed when the user clicks on it. The most common event associated with the command button is the Click event, and the syntax for the procedure is:

```vba
Private Sub Command1_Click()
    Statements
End Sub
```

### 3.2.4 The Picture Box

The Picture Box is the control that is used to handle graphics. You can load a picture during the designing phase by clicking on the picture item in the properties window and select the picture from the selected folder. You can also load the picture at runtime using the `LoadPicture` method. For example, this statement will load the picture grape.gif into the picture box.

```vba
Picture1.Picture = LoadPicture("C:\VB program\Images\grape.gif")
```

You will learn more about the picture box in future chapters. The image in the picture box is not resizable.
3.2.5 The Image Box

The Image Box is another control that handles images and pictures. It functions almost identically to the picture box. However, there is one major difference, the image in an Image Box is stretchable, which means it can be resized. This feature is not available in the Picture Box. Similar to the Picture Box, the LoadPicture method can also be used. For example, this statement loads the picture grape.gif into the image box.

```vbnet
Image1.Picture = LoadPicture("C:\VB program\Images\grape.gif")
```

3.2.6 The List Box

The function of the List Box is to present a list of items where the user can click and select items from the list. To add items to the list, we can use the AddItem method. For example, if you wish to add several items to List box 1, you can key in the following statements

**Example 3.2 Adding Items to a List Box**

```vbnet
Private Sub Form_Load()
  List1.AddItem "Chapter1"
  List1.AddItem "Chapter2"
  List1.AddItem "Chapter3"
End Sub
```
List1.AddItem "Chapter4"
End Sub

The items in the list box can be identified by the **ListItem** property, the value of the ListIndex for the first item is 0, the second item has a ListIndex 1, and the second item has a ListIndex 2 and so on

### 3.2.7 The Combo Box

The function of the Combo Box is also to present a list of items where the user can click and select the items from the list. However, the user needs to click on the small arrowhead on the right of the combo box to see the items which are presented in a drop-down list. To add items to the list, you can also use the **AddItem** method. For example, if you wish to add several items to Combo box 1, you can key in the following statements

#### Example 3.3 Adding Items to a Combo Box

Private Sub Form_Load()
    Combo1.AddItem "Item1"
    Combo1.AddItem "Item2"
    Combo1.AddItem "Item3"
    Combo1.AddItem "Item4"
End Sub
3.2.8 The Check Box

The Check Box control lets the user select or unselect an option. When the Check Box is checked, its value is set to 1 and when it is unchecked, the value is set to 0. You can include the statements Check1.Value = 1 to mark the Check Box and Check1.Value = 0 to unmark the Check Box, and use them to initiate certain actions. For example, the program will change the background color of the form to red when the check box is unchecked and it will change to blue when the check box is checked. You will learn about the conditional statement If...Then...Elseif in later chapters. VbRed and vbBlue are color constants and BackColor is the background color property of the form.

Example 3.4 Using Check Boxes

Private Sub Check1_Click ()

If Check1.Value = 0 Then
    Form1.BackColor = vbRed
ElseIf Check1.Value = 1 Then
    Form1.BackColor = vbBlue
End If

End Sub
3.2.9 The Option Button

The Option Button also lets the user selects one of the choices. However, two or more Option Buttons must work together because as one of the Option Buttons is selected, the other Option Buttons will be unselected. In fact, only one Option Button can be selected at one time. When an option button is selected, its value is set to "True" and when it is unselected; its value is set to "False". In the following example, the shape control is placed in the form together with six Option Buttons. When the user clicks on different option buttons, different shapes will appear. The values of the shape control are 0, 1, and 2,3,4,5 which will make it appear as a rectangle, a square, an oval shape, a rounded rectangle and a rounded square respectively.

**Example 3.5 Controlling Shapes with Option Buttons**

Private Sub Option1_Click ( )
    Shape1.Shape = 0
End Sub

Private Sub Option2_Click()
    Shape1.Shape = 1
End Sub

Private Sub Option3_Click()
Shape1.Shape = 2
End Sub

Private Sub Option4_Click()
Shape1.Shape = 3
End Sub

Private Sub Option5_Click()
Shape1.Shape = 4
End Sub

Private Sub Option6_Click()
Shape1.Shape = 5
End Sub

3.2.10 The Drive List Box

The Drive List Box is used to display a list of drives available in your computer. When you place this control into the form and run the program, you will be able to select different drives from your computer as shown in Figure 3.2
3.2.11 The Directory List Box

The Directory List Box is used to display the list of directories or folders in a selected drive. When you place this control into the form and run the program, you will be able to select different directories from a selected drive in your computer as shown in Figure 3.3.
3.2.12 The File List Box

The File List Box is used to display the list of files in a selected directory or folder. When you place this control into the form and run the program, you will be able to a list of files in a selected directory as shown in Figure 3.4.
*You can coordinate the Drive List Box, the Directory List Box and the File List Box to search for the files you want. The procedure will be discussed in later chapters.

Exercise 3

1. Start a Visual Basic project and set the following properties of the form
   a. Appearance=Flat
   b. Caption=My First Program
   c. Font=Times New Roman, Font Size=10
   d. Forecolor=White
   e. Backcolor=Blue
2. Insert six shape controls into the form and set their shape properties to rectangle, square, oval, circle, rounded rectangle, rounded square.

3. Insert two text boxes, one label and one command button, then write the program to compute the product of two numbers that are entered into the text boxes and display the result in the label.

4. Insert a picture control in the form and load a picture from your computer using the LoadPicture Method.

5. Insert a List Box control into the form and add the 5 items into the List Box using the AddItem Method.
Chapter 4 Writing the Codes

❖ Learn about the Visual basic program structure.
❖ Setting the properties of the controls at runtime.
❖ Getting to know some basic syntax in Visual Basic.

In chapter 2 and chapter 3, you have learned how to enter program and run the sample VB programs but without much understanding about the logics of VB programming. Therefore, in this chapter we will tackle a few basic rules about writing VB program codes.

Each control or object in VB can usually run many kinds of events or procedures; these events are listed in the dropdown list in the code window which is displayed when you click on the procedures’ box after double-clicking on an object (refer to Figure 2.3). Among the events are loading a form, clicking of a command button, pressing a key on the keyboard or dragging an object and etc. For each event, you need to write an event procedure so that an action or a series of actions can be performed.

To start writing an event procedure, you need to double-click an object. For example, when you double-click on the command
button, the code window will appear showing an event procedure:

Private Sub Command1_Click

(Key in your program code here)

End Sub

You then need to key in the procedure in the space between Private Sub Command1_Click............. End Sub. The program code is made up of a number of statements that set certain properties or trigger some actions. The syntax of Visual Basic’s program code is almost like English though not the same, so it is very easy to learn.

The syntax to set the property of an object or to assign certain values to it where **Object** and Property are separated by a period is:

Object.Property

For example, the statement **Form1.Show** means to show the form with the name Form1, **label1.Visible=true** means label1 is set to be visible, **Text1.text= "VB"** is to assign the text VB to the text box with the name Text1, Text2.text=100 is to assign a value of 100 to the text box with the name text2,
Timer1.Enabled=False is to disable the timer with the name Timer1 and so on. Let us examine a few examples below:

Example 4.1 Using Label and Text Box

Private Sub Command1_click

Label1.Visible=False
Label2.Visible=True
Text1.Text="You are correct!"
End sub

Example 4.2 Using Image Control

Private Sub Command1_click

Label1.Caption=" Welcome"
Image1.visible=True
End sub

Example 4.3 Using Picture Box and Timer

Private Sub Command1_click

Picture1.Show=true
Timer1.Enabled=True

Lable1.Caption="Start Counting"

End sub

In example 4.1, clicking on the command button will make label1 become invisible and label2 become visible; and the text "You are correct" will appear in Text1 textbox. In example 4.2, clicking on the command button will make the caption label1 change to "Welcome" and Image1 will become visible. Example, clicking on the command button will make Picture1 show up, timer starts running and the caption of label1 change to "Start Counting".

Syntaxes that do not involve setting of properties are also English-like, some of the commands are Print, If...Then...Else...End If, For...Next, Select Case...End Select, End and Exit Sub. For example, Print "Visual Basic" is to display the text Visual Basic on screen and End is to end the program. Other commands will be explained in detail in the coming chapters.

Program codes which involve calculations are very easy to write, as they are similar to mathematics. However, in order to write an
event procedure that involves calculations, you need to know the basic arithmetic operators in VB as they are not exactly the same as the normal operators that we use, except for + and -. For multiplication, we use *, for division we use /, for raising a number x to the power of n, we use \( x^n \) and for square root, we use \( \text{Sqr}(x) \). There are also more advanced mathematical functions such as \( \text{Sin}, \text{Cos}, \text{Tan}, \text{Log} \) and etc. Besides, there are also two important functions that are related to arithmetic operations, i.e. the functions \( \text{Val} \) and \( \text{Str} \) where \( \text{Val} \) is to convert text into a numeric and \( \text{Str} \) is to convert a numeric into a string. While the function \( \text{Str} \) is not as vital because VB can display a numeric values as a string implicitly, failure to use \( \text{Val} \) will result in wrong calculation. Let us examine example 4.4 and example 4.5.

**Example 4.4 Text Concatenation**

Private Sub Form_Activate()
    Text3.text=text1.text+text2.text
End Sub

**Example 4.5 Addition using Val**

Private Sub Form_Activate()
    Text3.text=val(text1.text)+val(text2.text)
End Sub
When you run the program in Example 4.4 and enter 12 in Text1 textbox and 3 in Textbox2, it will give you a result of 123, which is wrong. It is because VB treats the numbers as strings and so it just joins up the two strings. On the other hand, running Example 4.5 will give you the correct result, i.e., 15.

Now we shall write the codes for the cylinder program (the interface is shown in Figure 2.8). First, name the text box as radius, hght, and volume. To get the values of the various text boxes, use Val(radius.text), Val(hght.Text) and assign them to the variables r and h. In addition, assign the value 22/7 to the variable pi. After that, write the equation \( v = \pi \times (r^2) \times h \) to compute the value of volume of cylinder and then assign it to the variable. Finally, display the value in the volume textbox using the function Str.

```vbc
Private Sub OK_Click( )
    r = Val(radius.Text)
    h = Val(hght.Text)
    pi = 22 / 7
    v = pi * (r ^ 2) * h
    volume.Text = Str(v)
```
When you run the program, you should be able to see the interface as shown in Figure 2.8. Enter values in the radius box and the height box, and then click OK. The value of the Volume will be displayed in the volume box.

I shall attempt to explain the above source program to newcomers in Visual Basic (If you are a veteran, you can skip this part). The program could be explained using pseudo codes as follows:

- procedure for clicking the OK button to calculate the volume of cylinder
- get the value of r from the radius text box
- get the value of h from the height text box
- assign a constant value 22/7 to pi
- calculate the volume using formula
- output the results to the Volume text box
- End of Procedure

Exercise 4

1. Write a program to compute the area of a triangle.
2. Write a program to calculate the circumference and area of a circle.