Unit – 1 Introduction

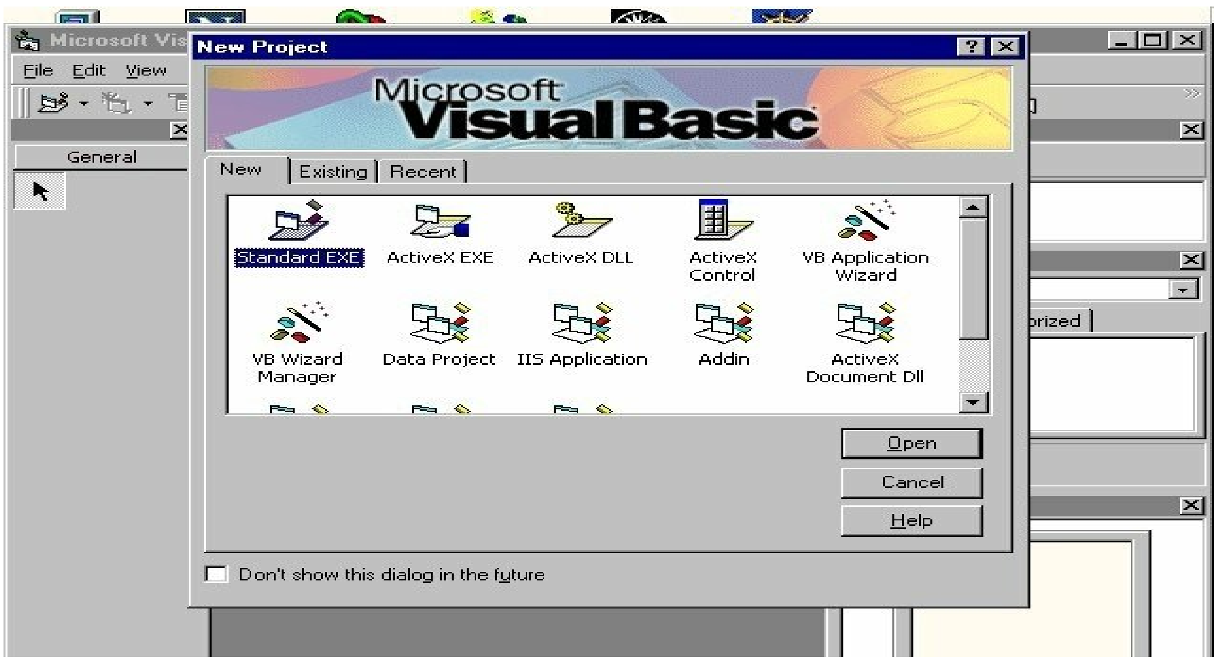
A brief description of Visual Basic

VISUAL BASIC is a high level programming language evolved from the earlier DOS version called BASIC. BASIC means **B**eginners' **A**llpurpose **S**ymbolic **I**nstruction **C**ode. It is a fairly easy programming language to learn. The codes look a bit like English Language. Different software companies produced different version of BASIC, such as Microsoft QBASIC, QUICKBASIC, GWBASIC ,IBM BASICA and so on.
VISUAL BASIC is a VISUAL and events driven Programming Language.These are the main divergence from the old BASIC. In BASIC, programming is done in a text-only environment and the prgram is executed sequentially. In VISUAL BASIC, programming is done in a graphical environment. Because users may click on a certain object randomly, so each object has to be programmed indepently to be able to response to those actions(events).Therefore, a VISUAL BASIC Program is made up of many subprograms, each has its own program codes, and each can be excecuted indepently and at the same time each can be linked together in one way or another.

The Visual Basic Environment

On start up, Visual Basic 6.0 will display the following 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 we could create, however, we shall concentrate on creating Standard EXE programs(EXE means executable program). Now, click on the Standard EXE icon to go into the actual VB programming environment.

Figure 1.1 The Visual Basic Start-up Dialog Box

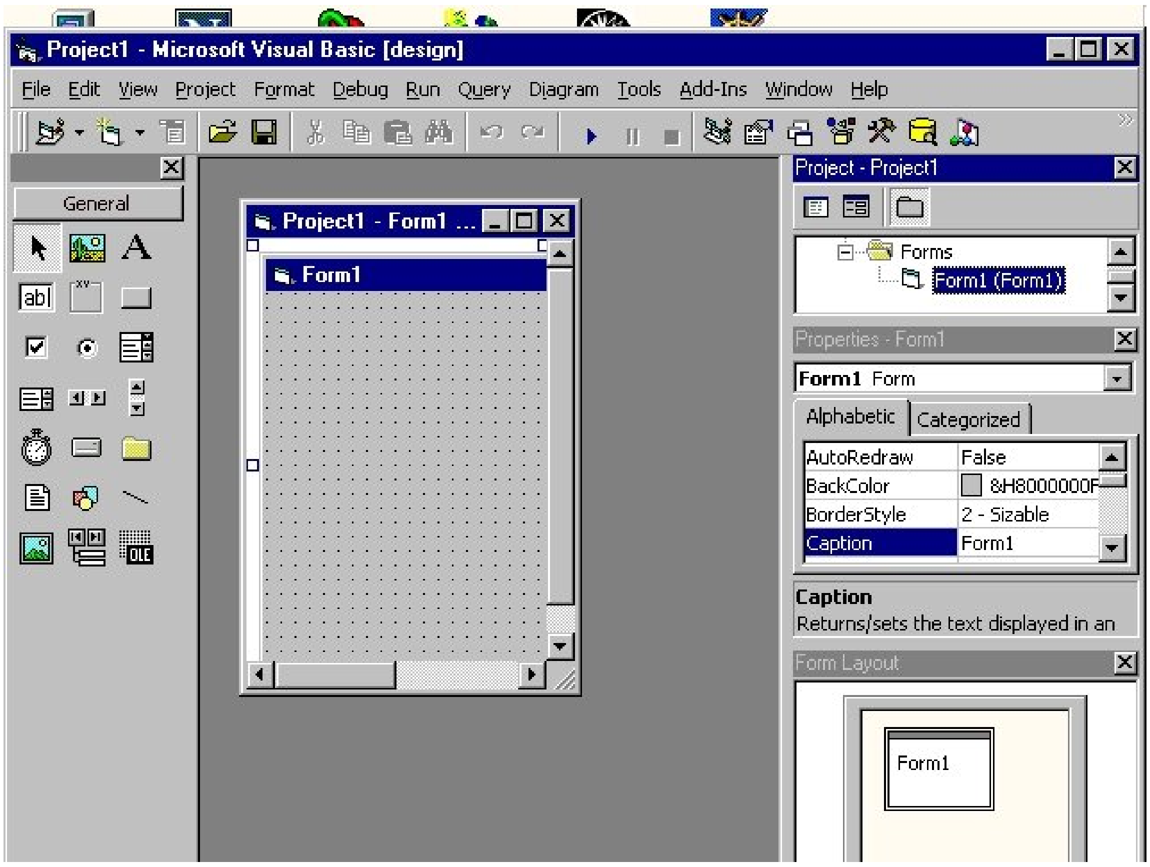


In figure 1.2, the Visual Basic Environment consists of the

- The **Blank Form** window which you can design your application's interface.
- The **Project** window 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 applications.

It also includes a **Toolbox** that consists of all the controls essential for developing a VB Application. Controls are tools such as boxes, buttons, labels and other objects draw on a form to get **input** or **display output**. They also add visual appeal.

Figure 1.2: The Visual Basic Enviroment



AutoRedraw	False
BackColor	&H8000000F
BorderStyle	2 - Sizable
Caption	Form1

Caption
Returns/sets the text displayed in an

Lesson 2: Building a Visual Basic Application
2.1 Creating Your First Application

In this section, we are not going into the technical aspect of VB programming, just have a feel of it. Now, you can try out the examples below:

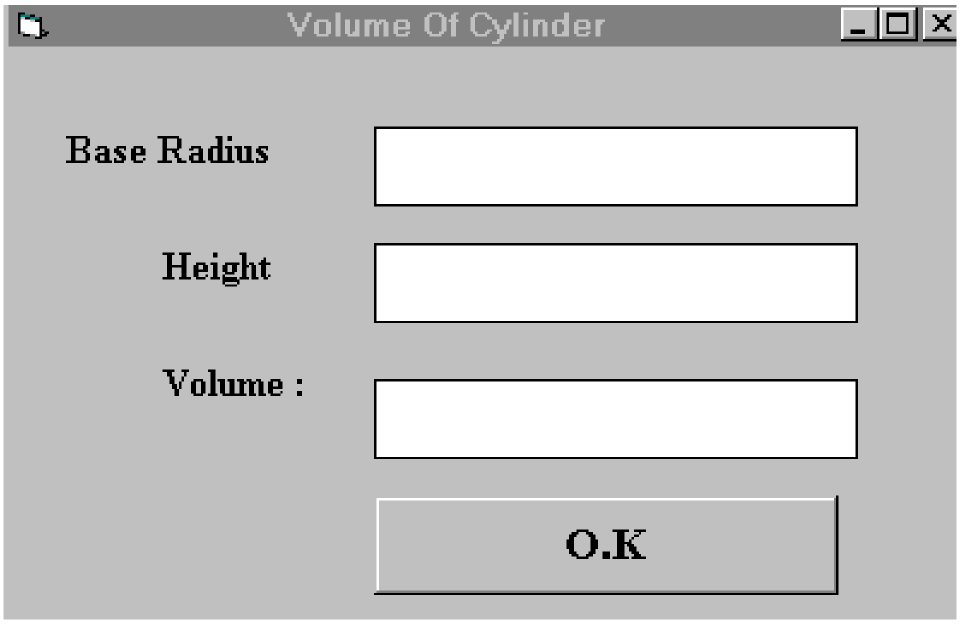
Example 2.1.1 is a simple program . First of all, you have to launch Microsoft Visual Basic. Normally, a default form **Form1** will be available for you to start your new project. Now, double click on form1, the source code window for form1 will appear. Don't worry about the begining and the end statements(i.e **Private Sub Form_Load................End Sub**.); Just key in the lines in between the above two statements exactly as are shown here.When you run the program, you will be surprise that nothing shown up.In order to display the output of the program, you have to add the **Form1.show** statement like in Example 21.2 and Example 2.1.3. Try them out.

Example	Example	Example
Private Sub	Private Sub	Private Sub
For i=1 to 5 print "Hello"	Form1.show For i=1 to 5 print "Hello"	Form1.show For i=1 to10 print i next i
End	End	End

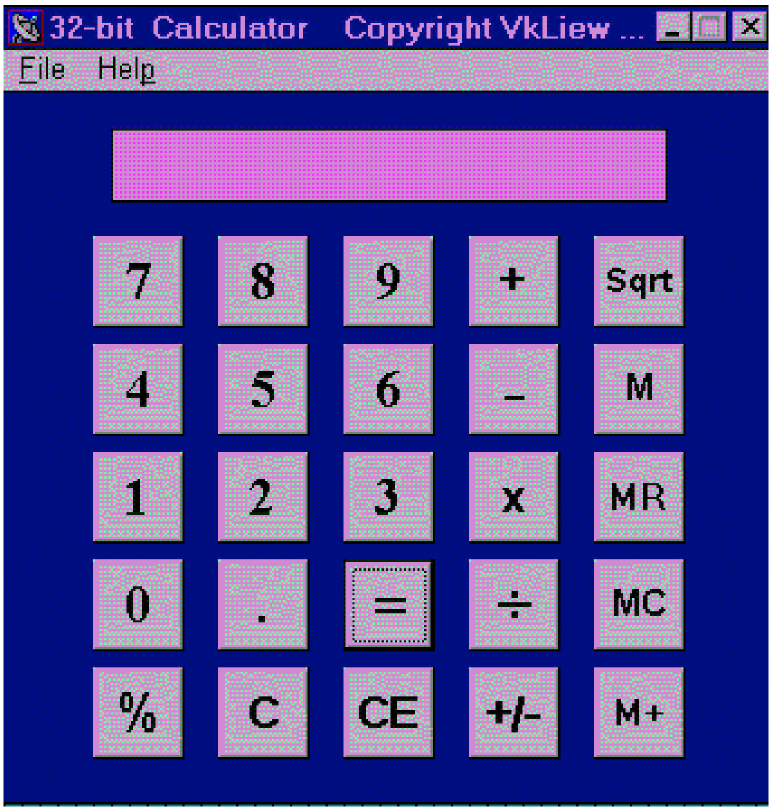
Steps in Building a Visual Basic Application

Step 1 ***Draw the interface***
Step 2 ***Set Properties***
Step 3 ***Write the events code***

Example 2.1
This program is a simple program that calculate the volume of a cylinder. Let design the interface:



First of all, go to the properties window and change the form caption to Volume Of Cylinder. Then draw three label boxes and change their captions to **Base Radius, height** and**volume** respectively. After that, draw three Text Boxes and clear its text contents so that you get three empty boxes. Named the text boxes as**radius ,hght(**we cannot use **height** as it is the built-in control name of VB)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.vbp as well. We shall leave out the codes at the moment which you shall learn it in lesson3.
Example 2.2
Designing an attractive and user friendly interface should be the first step in constructing a VB program. To illustrate, let's look at the calculator program.



Now, please follow the following steps to design the calculator interface.

- Resize the form until you get the size you are satisfed with.
- Go to the properties window and change the default caption to the caption you want , such as 32 Calculator ------------------- Designed by Vkliew.
- Change other properties of the form, such as background color, foreground color , border style.I recommend you set the following properties for Form1 for this calculator program:

BorderStyle	Fixed Single
MaxButton	False
minButton	True

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

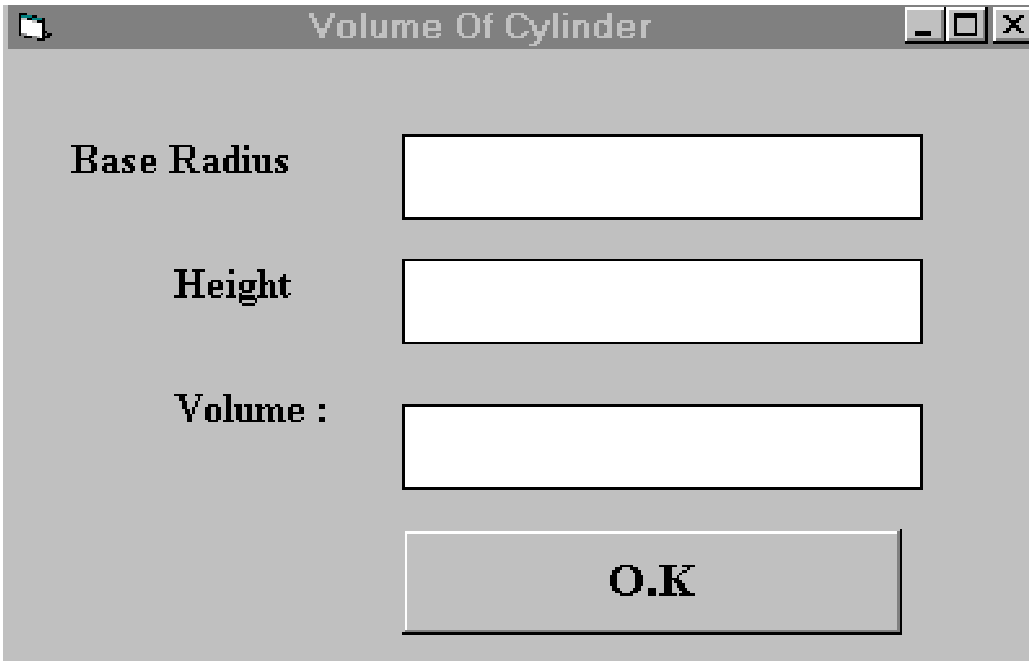
- Draw the Display Panel by clicking on the Label button and and place your mouse on the form. Start drawing by pressing down your mouse button and drag it along.
- Click on the panel and the corresponding properties window will appear. Clear the default label so that the caption is blank(because the display panel is supposed to show the number as we click on the number button). It is good to set the background color to a bright color while the foreground color should be something 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.

- Test run the project by pressing F5. If you are satisfied with the appearance, go ahead to save the project. At the same time, you should also save the file that contain your form.

 Now, I know you are very keen to know how to write the code so that the calculator is working. Please refer to my sample VB programs for the source codes.

Writing the Codes

Now we shall attempt to write the codes for the cylinder program.



Now, doubleclick on the O.K button and enter the codes between Private Sub OK_Click( ) and End Sub

```
Private Sub OK_Click( ) r =
Val(radius.Text)
h = Val(hght.Text) pi =
22 / 7
v = pi * (r ^ 2) * h
volume.Text= Str$(v)
End Sub
```

when you run the program , you should be able to see the interface as shown above. if you enter a value each in the radius box and the height box, then click OK, the value of 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) . Let me describe the steps using pseudocodes 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

The syntax ***radius.Text*** consists of two parts, radius is the **name** of text box while **Text** is the textual contents of the text box. Generally, the syntax is: **Object.Property**
In our example, the objects are ***radius, hght*** and ***volume***, each having ***text*** as their ***property***.Object and property is separated by a period(or dot).The contents of a text box can only be displayed in textual form, or in programming term,as string. To convert the contents of a text box to a numeric value so that mathematical operations can be performed , you have to use the function ***Val.*** Finally, In order to display the results in a text box, we have to perform the reverse procedure, that is, to convert the numeric value back to the textual form, using the function **Str$.**

I shall also explain the syntax that defines the sub procedure **Private Sub OK_click.** *Private Sub* here means that the parameters , values and formulas that are used here belong only to the OK subprocedure(an object by itself).They cannot be used by other sub procedures or modules. OK_Click defines what kind of action the subprocedure OK will response .Here, the action is mouse click. There are other kind of actions like keypress, keyup, keydown and etc that I am going to due with in other lessons.

Working With Controls
Before writing an event procedure for a control to response to a user's action, 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 windows. 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 a user know what to do with that command. For example, in the calculator program, all the captions of the command buttons such as +, - , MC ,MR are commonly found in an ordinary calculator, a user should have no problem in manipulating the buttons.
- You should set a meaningful name for the **Name Property** because it is easier for you to write and read the event procedure and easier to debug your program later.
- Another property that is important is whether you want your control to be **visible** or not at start up.This property can only set to be true or false.
- One more important property is whether the control is **enabled** or not.

Unit – II

There are many types of data we come across in our daily life. For example, we need to handle data such as names, adresses, money, date, stock quotes, statistics and etc everyday. Similarly In Visual Basic, we are also going to deal with these kinds of data. However, to be more systematic, VB divides data into different types.

Types of Visual Basic Data
Numeric Data

Numeric data are data that consists of numbers, which can be computed mathematically with various standard operators such as add, minus, multiply, divide and so on. In Visual Basic, the numeric data are divided into 7 types, they are summarised in Table 6.1

Table 5.1: Numeric Data Types

Type	Storage	Range of Values
Byte	1 byte	0 to 255
Integer	2 bytes	-32,768 to 32,767
Long	4 bytes	-2,147,483,648 to 2,147,483,648
	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E+38 for positive values.
	8 bytes	-1.79769313486232e+308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232e+308 for positive values.
Currency	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	12 bytes	+/- 79,228,162,514,264,337,593,543,950,335 if no decimal is use +/- 7.9228162514264337593543950335 (28 decimal places).

Non-numeric Data Types

Table 5.2: Nonnumeric Data Types

Data Type	Storage	Range
String(fixed length)	Length of string	1 to 65,400 characters
String(variable length)	Length + 10 bytes	0 to 2 billion characters
Date	8 bytes	January 1, 100 to December 31, 9999
Boolean	2 bytes	True or False
Object	4 bytes	Any embedded object
Variant(numeric)	16 bytes	Any value as large as Double
Variant(text)	Length+22 bytes	Same as variable-length string

Suffixes for Literals

Literals are values that you assign to a data. In some cases, we need to add a suffix behind a literal so that VB can handle the calculation more accurately. For example, we can use num=1.3089# for a Double type data. Some of the suffixes are displayed in Table 5.3.

Table 5.3

Suffix	Data Type
&	Long
!	Single
#	Double
@	Currency

In additon, we need to enclose string literals within two quotations and date and time literals within two # sign. Strings can contain any characters, including numbers. The following are few examples:

memberName="Turban,
John."
TelNumber="1800-900-
888-777"
LastDay=#3
1-Dec-00#
ExpTime=#
12:00 am#

Managing Variables

Variables are like mail boxes in the post office. The contents of the variables changes every now and then, just like the mail boxes. In term of VB, variables are areas allocated by the computer memory to hold data. Like the mail boxes, each variable must be given a name. To name a variable in Visual Basic,

you have to follow a set of rules.

Variable Names

The following are the rules when naming the variables in Visual Basic

- It must be less than 255 characters
- No spacing is allowed
- It must not begin with a number
- Period is not permitted

Examples of valid and invalid variable names are displayed in Table 5.4

Table 5.4

Valid Name	Invalid Name	
My_Car	My.Car 1NewBoy	
ThisYear	He&HisFather	*& is not acceptable
Long_Name_Can_beUSE		

Declaring Variables

In Visual Basic, one needs to declare the variables before using them by assigning names and data types. They are normally declared in the genaral section of the codes' windows using the **Dim** statement. The format is as follows:

Dim variableNmae as

DataType Example 5.1

Dim password As
String Dim
yourName As
String Dim
firstnum As
Integer Dim
secondnum As
Integer Dim total
As Integer
Dim doDate As Date

You may also combine them in one line , separating each variable with a comma, as follows:

Dim password As String, yourName As String, firstnum As Integer,............

If data type is not specified, VB will automatically declares the variable as a Variant.
For string declaration, there are two possible format, one for the variable-length string and another for the fixed-length string. For the variable-length string, just use the same format as example 5.1 above. However, for the fixed-length string, you have to use the format as shown below:

*Dim VariableName as String * n*, where n definex the number of characters the string can hold.

Example 5.2:

Dim yourName as String * 10

yourName can holds no more than 10 Characters.

Working with Variables

Assigning Values to Variables

After declaring various variables using the Dim statements, we can assign values to those variables. The general format of an assignment is

Variable=Expression

The variable can be a declared variable or a control property value. The expression could be a mathematical expression, a number, a string, a boolean value(true or false) and etc. The following are some examples:

```
firstNumber=100
secondNumber=firstNumber
-99 userName="John Lyan"
userpass.Text = password
Label1.Visible = True
Command1.Visible = false
Label4.Caption =
textbox1.Text ThirdNumber
= Val(usernum1.Text)
total = firstNumber + secondNumber+ThirdNumber
```

Operators in Visual Basic

In order to compute inputs from users and to generate results, we need to use various mathematical operators. In Visual Basic, except for + and -, the symbols for the operators are different from normal mathematical operators,as shown in Table 6.1.

Table 6.1

Operator	Mathematical function	Example
^	Exponential	2^4=16
*	Multiplication	4*3=12
/	Division	12/4=3
Mod	Modulus(return the remainder from an integer division)	15 Mod 4=3
\	Integer Division(discards the decimal places)	19\4=4
+ or &	String concatenation	"Visual"&"Basic"="Visual Basic"

Example 6.1:

```
firstName=Text1.Text
secondName=Text2.Text
yourName=firstName+seco
ndName
```

```
number1=val(Text3.Text)
number2=val(Text4.Text)
number3=num1*(num2^3)
number4=number3 Mod 2
number5=number4\number
1
Total=number1+number2+number3+number4+num
ber5 Average=Total/5
```

In lesson, we will see how do we use operators in writing the VB programs codes.

Controlling Program Flow

Conditional Operators

To control the VB program flow, we can use various conditional operators. Basically, they resemble mathematical operators. Conditional operators are very powerful tools, they let the VB program compare data values and then decide what action to take, whether to execute a program or terminate the program and etc. These operators are shown in Table 7.1.

Table 7.1: Conditional Operators

Operator	Meaning
=	Equal to
>	More than
<	Less Than
>=	More than and equal
<=	Less than and equal
<>	Not Equal to

* You can also compare strings with the above operators. However, there are certain rules to follows: Upper case letters are less than lowercase letters, "A"<"B"<"C"<"D". ....<"Z" and number are less than
letters.

Logical Operators

In addition to conditional operators, there are a few logical operators which offer added power to the VB programs. There are shown in Table 7.2.

Table 7.2

Operator	Meaning
And	Both sides must be true
or	One side or other must be true
Xor	One side or other must be true but not both
Not	Negates truth

Using If.....Then.....Else Statements with Opreators

To effectively control the VB program flow, we shall use If...Then...Else statement together with the conditonal operators and logical operators.
The general format for the if...then...else statement is

If conditions

Then VB

expressions **Else**

VB expressions

End If

* any If..Then..Else statement must end with End If. Sometime it is not necessary to use Else.

Example:

```
Private Sub OK_Click()

firstnum = Val(usernum1.Text)
secondnum = Val(usernum2.Text)
total = Val(sum.Text)
If total = firstnum + secondnum And Val(sum.Text) <> 0 Then correct.Visible =
True
wrong.Visible =
False Else
correct.Visible =
False wrong.Visible
= True End If

End Sub
```

For more example on If...Then...Else, Click on the [Sample1] and[sample2] program here.

More On Program Control
Select Case
If you have a lot of conditional statements, using If..Then..Else could be very messy. For multiple conditional statements, it is better to use Select Case
The format is :

Select Case

expression

Case value1

```
      Block of one or more VB
statements Case value2
      Block of one or more VB
Statements Case value3
      Block of one or more VB
statements Case value4
.
.
.
Case Else
          Block of one or more VB

Statements End Select
```

* The data type specified in expression must match that of Case values.

Examples

Example 8.1
Examinati

on Grades

```
Dim grade

As String

Private Sub

Compute_Click( )

grade=txtgrade.Text

Select Case grade

Case  "A"
      result.Caption="High
Distinction" Case "A-"
     result.Caption="Dist
inction" Case "B"
       result.Caption
="Credit" Case "C"
result.Caption="Pass"
Case Else
       result.Capti
on="Fail" End
Select
```

*Please note that grade is a string, so all the case values such as "A" are of String data type.

Example 8.2

```
Dim mark As Single
Private Sub Compute_Click()
'Examination Marks

 mark

=

mrk.T

ext

Select

Case

mark

Case

Is   >=

85

comment.Caption = "Excellence"

Case Is >= 70

comment.Caption =

"Good"

Case Is >= 60

   comment.Caption = "Above

Average" Case Is >= 50

comment.Caption = "Average"

Case Else

comment.Caption = "Need to work

harder" End Select
```

End Sub

* Note we use the keyword Is here to impose the conditions. This is generally used for numeric data.

Example 8.3

Example 8.2 could be rewritten as

follows: Dim mark As Single
Private Sub Compute_Click()
'Examination Marks

 mark

=

mrk.T

ext

Select

Case

mark

Case

0 to

49

comment.Caption = "Need to work harder"

Case 50 to 59

comment.Caption = "Average"

Case 60 to 69

   comment.Caption = "Above

Average" Case 70 to 84

comment.Caption = "Good"

Case Else

comment.Caption =

"Excellence" End Select

End Sub

Looping

Visual Basic allows a procedure to be repeated as many times as long as the processor could support. This is generally called looping .

Do Loop

The format are

a) Do While condition

Block of one or more VB

statements Loop

b) Do
Block of one or more VB statements
Loop While condition

c) Do Until condition
Block of one or more VB
statements Loop

d) Do
Block of one or more VB statements

Loop

Until condition

Example 9.1

Do while counter

<=1000

num.Text=counter

counter =counter+1

Loop

* The above example will keep on adding until counter >1000.

The above example can be

rewritten as Do

num.Text=counter
counter=counter+1

Loop until counter>1000

For....Next Loop

The format is:

For counter=startNumber to endNumber (Step increment) One

or more VB statements

Next

Example:

(a) For counter=1 to 10

display.Text=count

er Next

(b) For counter=1 to 1000 step

10 counter=counter+1

Next

(c) For counter=1000

to 5 step -5 counter=counter-

10

Next

Unit – III

Functions are similar to normal procedures but the main purpose of the functions is to accept certain inputs and pass them on to the main program to finish the execution. They are two types of function, the built-in functions(or internal functions) and the functions created by the programmers.

The general format of a function is

functionName(arguments)

where arguments are values that are passed on to the functions.

In this lesson, we are going to learn two very basic but useful internal functions, i.e. the MsgBox( ) and InputBox ( ) functions.

MsgBox ( ) Function

The objective of MsgBox is to produce a pop-up message box and prompt the user to click on a command button before he /she can continues. This message box format is as follows:

yourMsg=MsgBox(Prompt, Style Value, Title)

The first argument, Prompt, will display the message in the message box. The Style Value will determine what type of command buttons appear on the message box, please refer Table 10.1 for types of command button displayed. The Title argument will display the title of the message board.

Table 10.1: Style Values

Style Value	Named Constant	Buttons Displayed
0	vbOkOnly	Ok button
1	vbOkCancel	Ok and Cancel buttons
2	vbAbortRetryIgnore	Abort, Retry and Ignore buttons.
3	vbYesNoCancel	Yes, No and Cancel buttons
4	vbYesNo	Yes and No buttons
5	vbRetryCancel	Retry and Cancel buttons

We can use named constant in place of integers for the second argument to make the programs more readable. Infact, VB6 will automatically shows up a list of names constant where you can select one of them.

example: yourMsg=MsgBox("Click OK to Proceed", 1, "Startup Menu")

and yourMsg=Msg("Click OK to Proceed". vbOkCancel,"Startup

Menu") are the same.

yourMsg is a variable that holds values that are returned by the MsgBox ( ) function. The values are determined by the type of buttons being clicked by the users. It has to be declared as Integer data type in the procedure or in the general declaration section. Table 10.2 shows the values, the corresponding named constant and buttons.

Table 10.2 : Return Values and Command Buttons

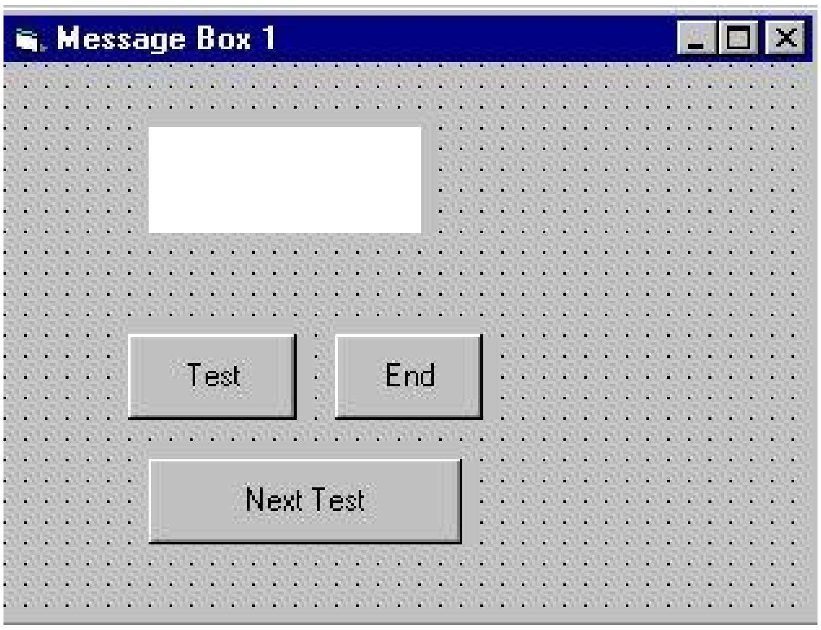
Value	Named Constant	Button Clicked
1	vbOk	Ok button
2	vbCancel	Cancel button
3	vbAbort	Abort button
4	vbRetry	Retry button
5	vbIgnore	Ignore button
6	vbYes	Yes button
7	vbNo	No button

Example 10.1

i. The Interface:

You draw three command buttons and a label as shown in Figure 10.1

Figure 10.1

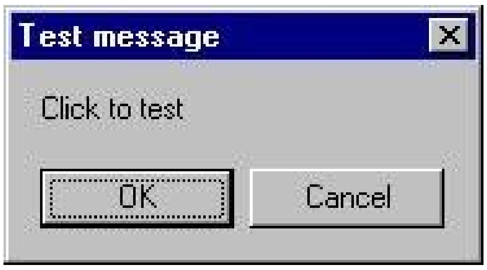


ii. The procedure for the test button:

```
Private Sub Test_Click()
Dim testmsg As Integer
testmsg = MsgBox("Click to test", 1, "Test message") If
testmsg = 1 Then
Display.Caption = "Testing Successful"
Else
Display.Caption = "Testing
fail" End If

End Sub
```

When a user click on the test button, the image like the one shown in Figure 10.2 will appear. As the user click on the OK button, the message "Testing sucessful" will be diplayed and when he/she clicks on the Cancel button, the message "Testing fail" will be displayed.

Figure 10.2


Test message

Click to test

To make the message box looks more sophisticated, you can add an icon besides the message. The are four types of icons available in VB as shown in Table 10.3

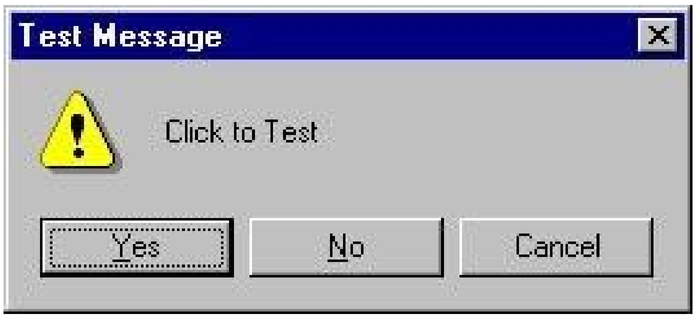
Table 10.3

Value	Named Constant	Icon
16	vbCritical	❌
32	vbQuestion	❓
48	vbExclamation	⚠
64	vbInformation	ℹ

Example 10.2

In this example, the following message box will be displayed:

Figure 10.3



You could draw the same Interface as in example 10.1 but modify the codes as follows:

```
Private Sub test2_Click()
Dim testMsg2 As Integer
testMsg2 = MsgBox("Click to Test", vbYesNoCancel + vbExclamation, "Test Message")
If testMsg2 = 6 Then
display2.Caption = "Testing successful"
ElseIf testMsg2 = 7 Then
display2.Caption = "Are you sure?"
Else
display2.Caption = "Testing
fail" End If

End Sub
```

The InputBox( ) Function

An InputBox( ) function will display a message box where the user can enter a value or a message in the form of text. The format is

myMessage=InputBox(Prompt, Title, default_text, x-position, y-position)

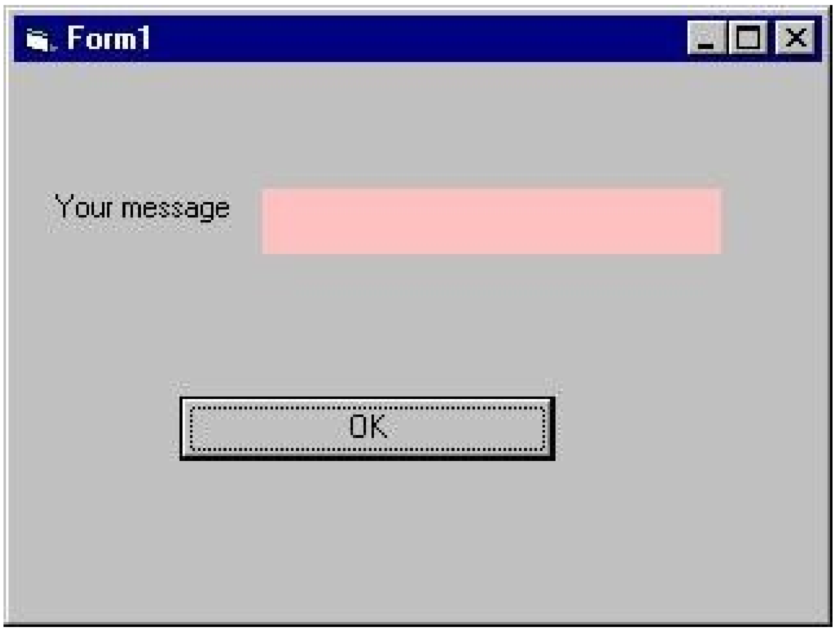
myMessage is a variant data type but typically it is declared as string, which accept the message input bu the users.The arguments are explained as follows:

- Prompt - The message displayed normally as a question asked.
- Title - The title of the Input Box.
- default-text - The default text that appears in the input field where users can use it as his intended input or he may change to the message he wish to key in.
- x-position and y-position - the position or the coordinate of the input box.

Example 10.3

i. The Interface

Figure 10.4

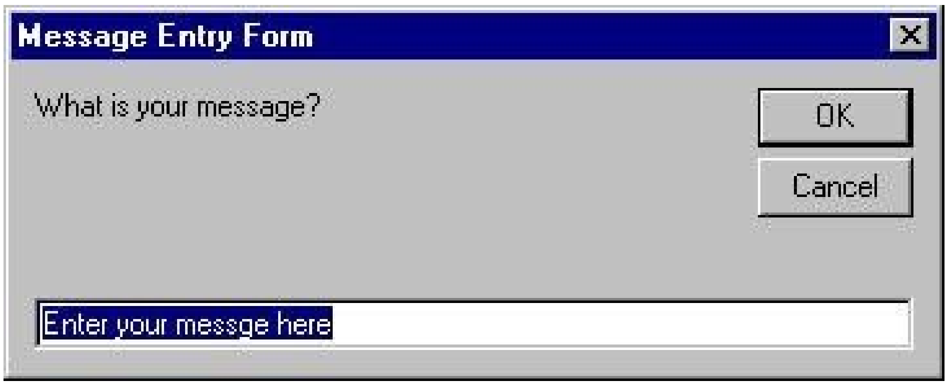


ii. The procedure for the OK button

```
Private Sub OK_Click()
Dim userMsg As String
userMsg = InputBox("What is your message?", "Message Entry Form", "Enter your messge here",
500, 700)
If userMsg <> "" Then
message.Caption =
userMsg Else
message.Caption = "No Message"
End If

End Sub
```

When a user click the OK button, the input box as shown in Figure 10.5 will appear. After user entering the message and click OK, the message will be displayed on the caption, if he click Cancel, "No

message" will be displayed.



Introduction to VB Functions- Part II
11.1 Creating Your Own Functions

The general format of a function is as follows:

Public Function functionName (Arg As dataType,. ....) As dataType

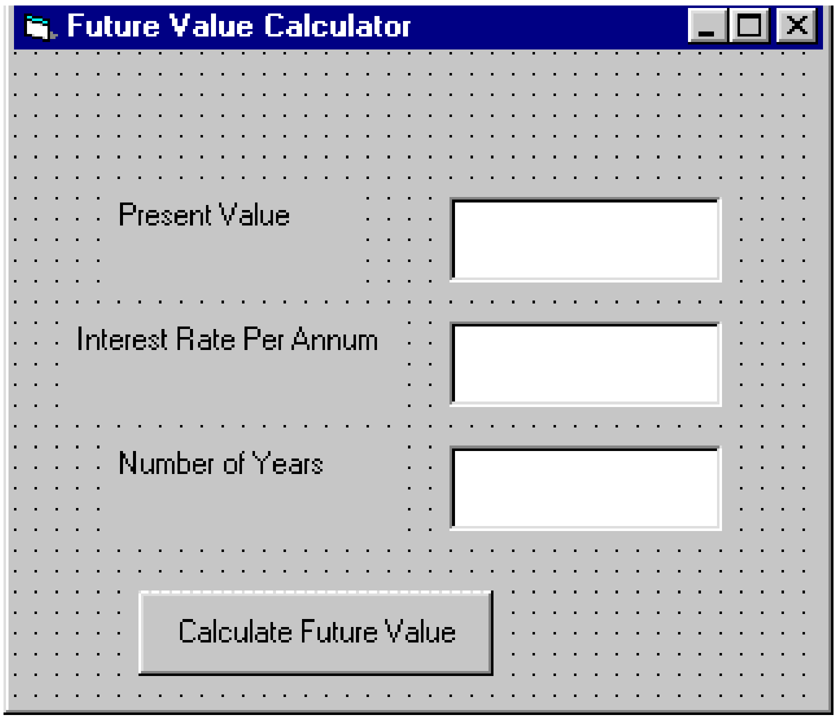
or

Private Function functionName (Arg As dataType,. ....) As dataType

* Public indicates that the function is applicable to the whole program and
Private indicates that the function is only applicable to a certain module or procedure.

Example 11.1

In this example, a user can calculate future value of a certain amount of money he has today based on the interest rate and the number of years from now(supposing he will invest this amount of money somewhere). The calculation is based on the compound interest rate.

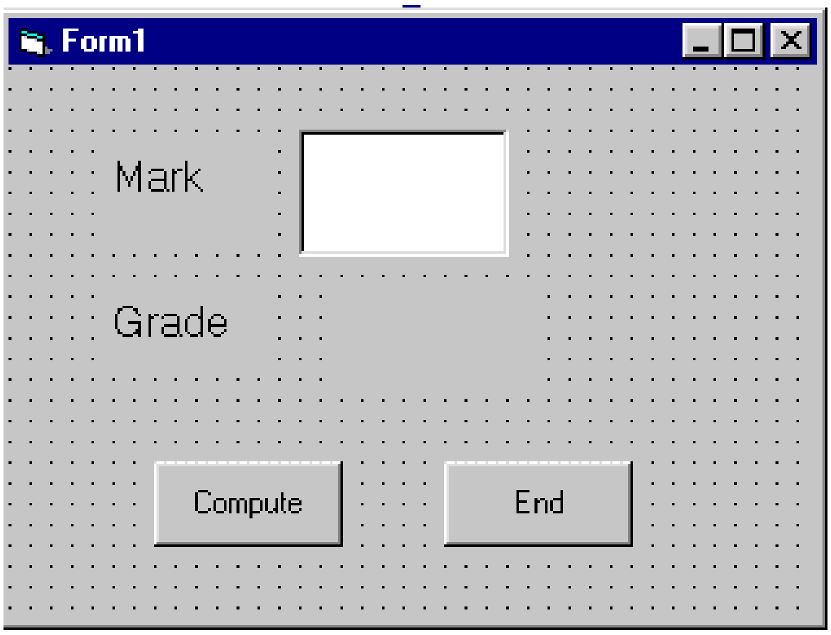


Public Function FV(PV As Variant, i As Variant, n As Variant) As Variant

```
'Formula to calculate Future
Value(FV) 'PV denotes Present
Value
FV = PV * (1 + i / 100) ^ n

End Function

Private Sub compute_Click()
'This procedure will calculate
Future Value Dim FutureVal As
Variant
Dim PresentVal
As Variant Dim
interest As
Variant Dim
period As Variant
PresentVal =
PV.Text interest
= rate.Text
period = years.Text

FutureVal = FV(PresentVal, interest,
period) MsgBox ("The Future Value is "
& FutureVal) End Sub
```

Example 11.2

The following program will automatically compute examination grades based on the marks that a student obtained.



```
Public Function grade(mark As Variant) As
```

```
String Select Case mark

Private Sub

compute_Click()

grading.Caption =

grade(mark)


End Sub

Private Sub
End_Click()
End

End Sub
```

Creating VB Functions For MS Excel
12.2 The Needs to Create User-Defined Functions in MS-Excel

You can create your own functions to supplement the built-in functions in Microsoft Excel spreadsheet which are quite limited. These functions could be very useful and powerful if you know how to program them properly. One main reason we need to create user defined functions is to enable us to customize our spreadsheet environment for individual needs. For example, we might need a function that could calculate commissions payment based on the sales volume, which is quite difficult if not impossible by using the built-in function alone. Lets look at the table below:

Table 12.1: Commissions Payment Table

Sales Volume($)	Commissons
<500	3%
<1000	6%
<2000	9%
<5000	12%
>5000	15%

In the above table, if a saleman attain a sale volume of $6000, he will be paid $6000x12%=$720.00. A visual basic function to calculate the commissions could be written as
follows:

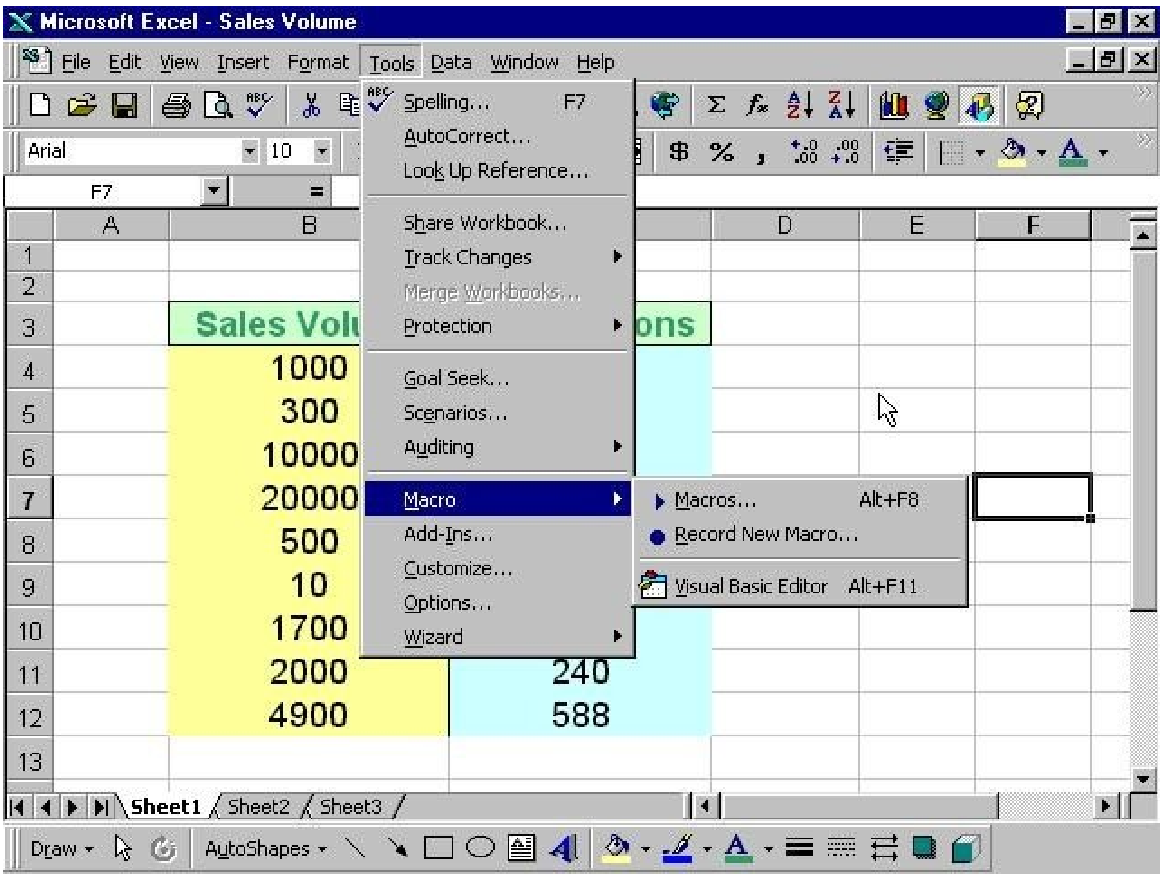
```
Function Comm(Sales_V As Variant)
as Variant If Sales_V <500 Then
Comm=Sales_V*0.03
Elseif Sales_V>=500 and
Sales_V<1000 Then
Comm=Sales_V*0.06
```

```
Elseif Sales_V>=1000 and
Sales_V<2000 Then
Comm=Sales_V*0.09
Elseif Sales_V>=200 and
Sales_V<5000 Then
Comm=Sales_V*0.12
Elseif Sales_V>=5000 Then
Comm=Sales_V*0.15
End If
End Function
```

12.2 Using Microsoft Excel Visual Basic Editor

To create User Defined functions in MS Excel, you can click on tools, select macro and then click on Visual Basic Editor as shown in Figure 12.1

Figure 12.1: Inserting Ms_Excel Visual Basic Editor

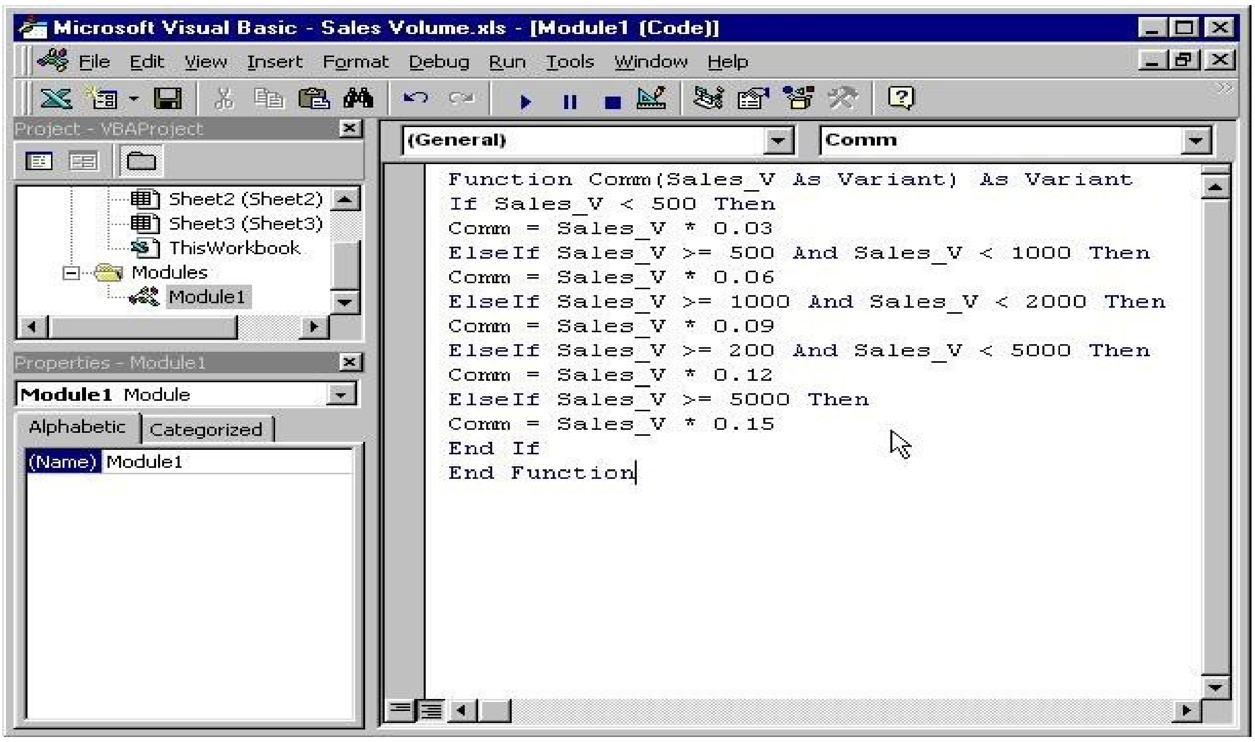


	A	B	C	D	E	F
1						
2						
3		Sales Vol	ons			
4		1000				
5		300				
6		10000				
7		20000				
8		500				
9		10				
10		1700				
11		2000	240			
12		4900	588			
13						

Upon clicking the Visual Basic Editor, the VB Editor windows will appear as shown in figure 12.2. To create a function, type in the function as illustrated in section 12.1 above After typing, save the

file and then return to the Excel windows.

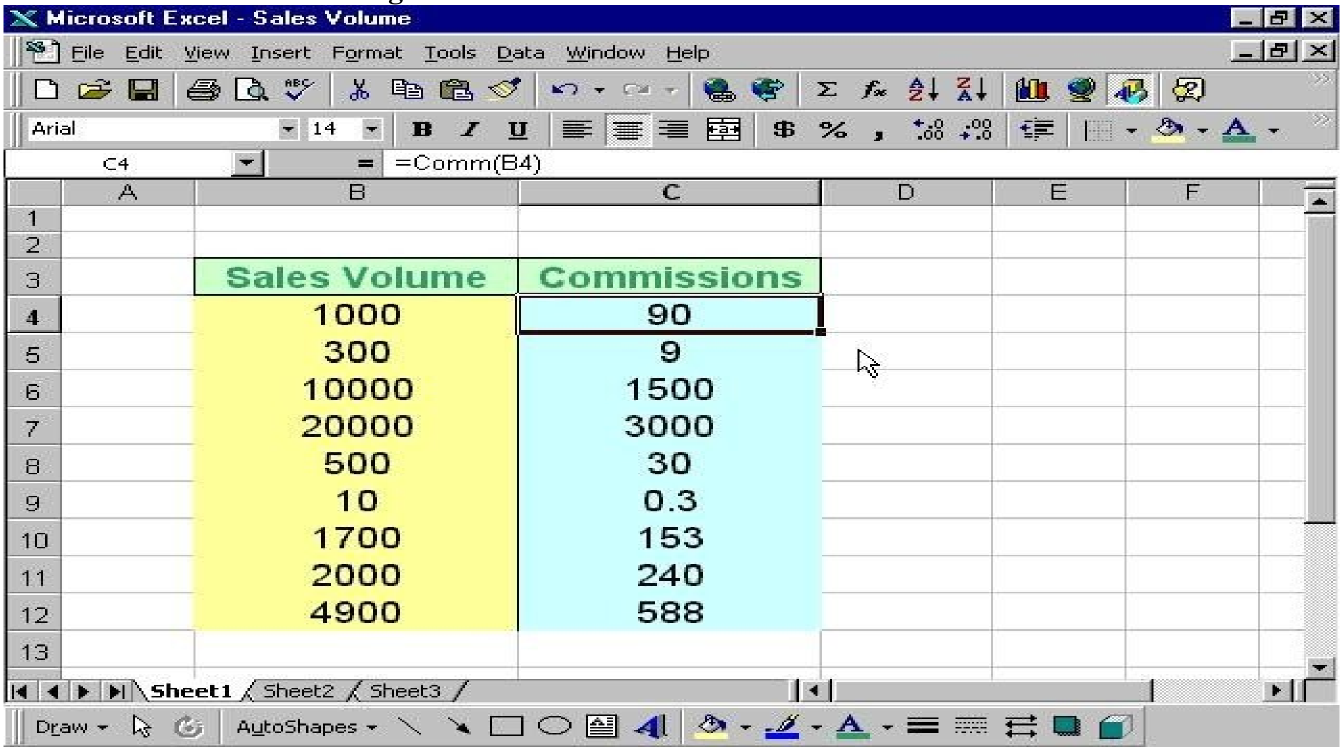
Figure 12.2 : The VB Editor



```
Function Comm(Sales_V As Variant) As Variant
If Sales_V < 500 Then
Comm = Sales_V * 0.03
ElseIf Sales_V >= 500 And Sales_V < 1000 Then
Comm = Sales_V * 0.06
ElseIf Sales_V >= 1000 And Sales_V < 2000 Then
Comm = Sales_V * 0.09
ElseIf Sales_V >= 200 And Sales_V < 5000 Then
Comm = Sales_V * 0.12
ElseIf Sales_V >= 5000 Then
Comm = Sales_V * 0.15
End If
End Function
```

In the Excel windows, type in the titles Sales Volume and Commissions in any two cells. By refering to figure 12.3, key-in the Comm function at cell C4 and by referencing the value in cell B4, using the format Comm(B4). Any value appear in cell B4 will pass the value to the Comm function in cell C4. For the rest of the rows, just copy the formula by draging the bottom right corner of cell C4 to the required cells, and a nice and neat table that show the commisions will automatically appear. It can also be updated anytime

Figure 12.3: MS Excel Windows- Sales Volume


Microsoft Excel - Sales Volume
File Edit View Insert Format Tools Data Window Help
Arial
14
C4
=Comm(B4)
Sheet1
Sheet2
Sheet3
Draw
AutoShapes

Sales Volume	Commissions
1000	90
300	9
10000	1500
20000	3000
500	30
10	0.3
1700	153
2000	240
4900	588

Arrays

Introduction to Arrays

By definition, an array is a list of variables, all with the same data type and name. When we work with a single item, we only need to use one variable. However, if we have a list of items which are of similar type to deal with, we need to declare an array of variables instead of using a variable for each item. For example, if we need to enter one hundred names, instead of declaring one hundred different variables, we need to declare only one array. We differentiate each item in the array by using subscript, the index value of each item, for example name(1), name(2),name(3) ....etc.

Declaring Arrays

We could use Public or Dim statement to declare an array just as the way we declare a single variable. The Public statement declares an array that can be used throughout an application while the Dim statement declare an array that could be used only in a local procedure.

The general format to declare an array is as follow:

Dim arrayName(subs) as dataType

where subs indicates the last subscript in the array.

Example 13.1

Dim CusName(10) as String

will declare an array that consists of 10 elements if the statement Option Base 1 appear in the declaration area, starting from CusName(1) to CusName(10). Otherwise, there will be 11 elements in the array starting from CusName(0) through to CusName(10)

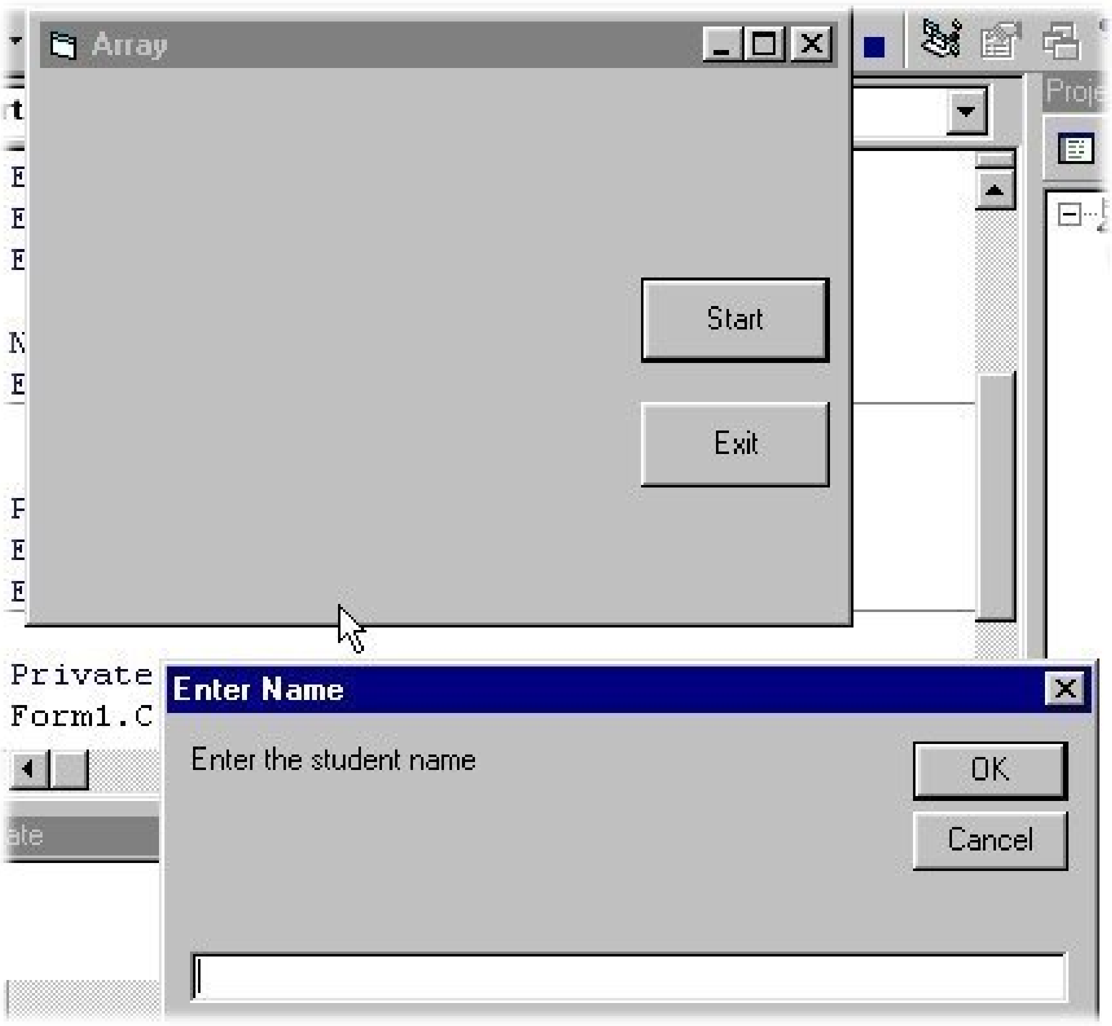
Example 13.2

Dim Count(100 to 500) as Integer

declares an array that consists of the first element starting from Count(100) and ends at Count(500)

Sample Programs

The Interface



The codes

Dim studentName(10)
As String Dim num As
Integer

Private Sub addName()
For num = 1 To 10
studentName(num) = InputBox("Enter the student name", "Enter Name", "", 1500, 4500)
If studentName(num) <> "" Then
Form1.Print
studentName(num)
Else
E
E
n
d

I

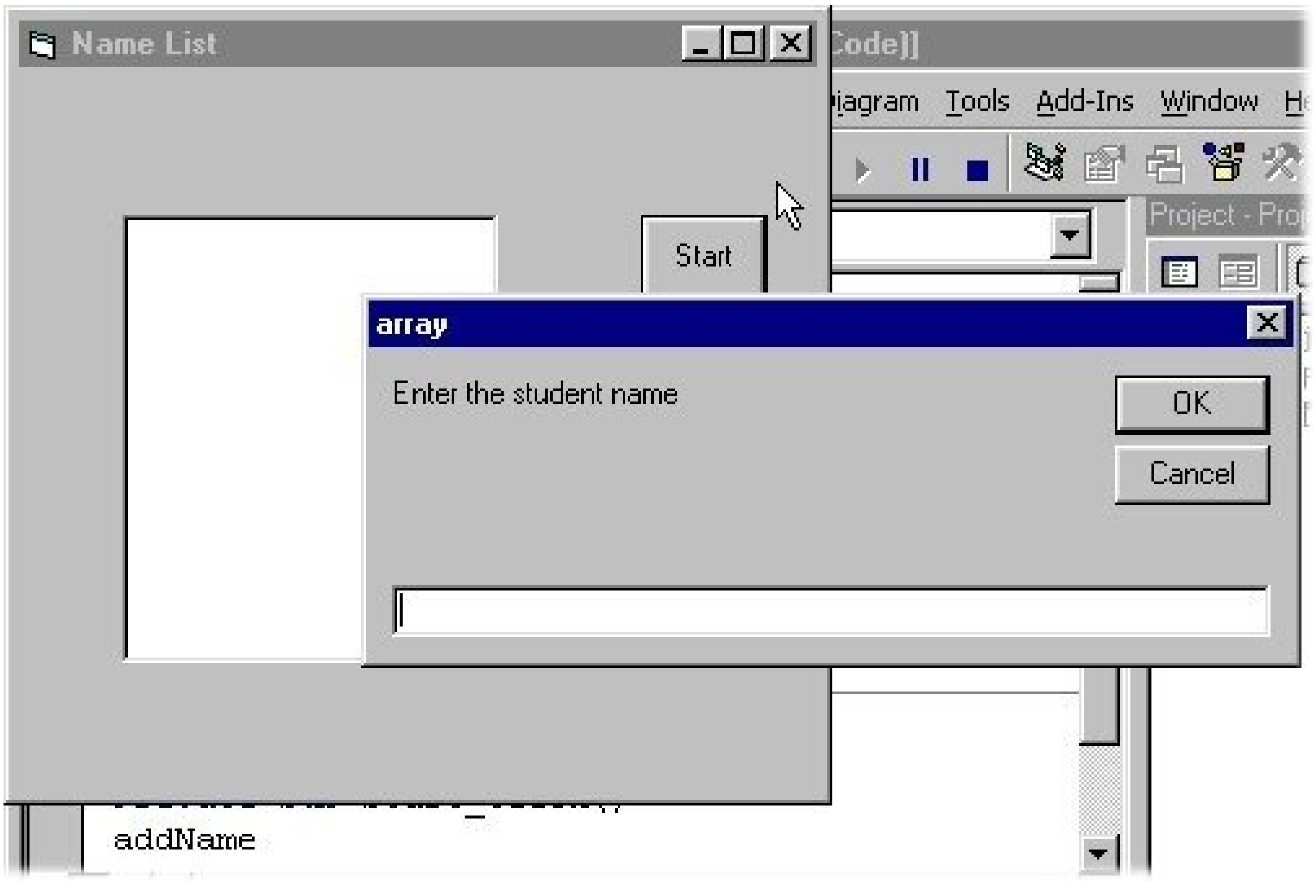
F

```
Next
End Sub

Private Sub
Exit_Click()
End
End Sub

Private Sub Start_Click()
Form1.Cls
addName
End Sub
```

The above program accepts data entry through an input box and displays the entries in the form itself. As you can see, this program will only allows a user to enter 10 names each time he click on the start button.

(ii)

The Interface


Name List
Code)]
iagram Tools Add-Ins Window He
Project - Pro
Start
array
Enter the student name
OK
Cancel

```
addName
```

The Codes

```
Dim studentName(10)
As String Dim num As
Integer

Private Sub addName( )
For num = 1 To 10
studentName(num) = InputBox("Enter the student
name") List1.AddItem studentName(num)
```

```
N

e
x
t

E
n
d

S
u
b
```

```
Private  Sub  Start_Click()
addName

End Sub
```

The above program accepts data entries through an InputBox and displays the items in a list box.

[Back to contents page]

Lesson 14: Working with Files

Introduction

Up until lesson 13 we are only creating programs that could accept data at runtime, when a program is terminated, the data also disappear. Is it possible to save data accepted by a VB program into a storage device, such as a hardisk or diskette, or even CDRW? The answer is possible. Is this chapter, we will learn how to create files by writing them into a storage device and then retrieve the data by reading the contents of the files using customized VB programs.

Creating files

To create a file , use the following command

```
Open "fileName" For Output As #fileNumber
```

Each file created must have a file name and a file number for identification. As for file name, you must also specify the path where the file will reside.

For example

```
Open "c:\My Documents\sample.txt" For Output As #1
```

will create a text file by the name of sample.txt in the My Document folder. The accompany file number is 1. If you wish to create and save the file in A drive, simply change the path, as follows"

```
Open "A:\sample.txt" For Output As #1
```

If you wish to create a HTML file , simple change the extension to .html

```
Open "c:\My Documents\sample.html" For Output As # 2
```

Sample Program : Creating a text file

```
Private Sub
create_Click()
Dim intMsg As
String
Dim StudentName As String

Open "c:\My Documents\sample.txt" For Output
As #1 intMsg = MsgBox("File sample.txt
opened")
```

```
StudentName = InputBox("Enter the student Name")
Print #1, StudentName
intMsg = MsgBox("Writing a" & StudentName & " to sample.txt ")

Close #1

intMsg = MsgBox("File sample.txt
closed") End Sub
```

* The above program will create a file sample.txt in the My Documents' folder and ready to receive input from users. Any data input by users will be saved in this text file.

Reading files

To read a file created in section 14.2, you can use the input # statemment. However, we can only read the file according to the format when it was written. You have to open the file according to its file number and the variable that hold the data. We also need to declare the variable using the DIM command.

Sample Program: Reading file

```
Private Sub
Reading_Click()
Dim variable1 As
String
Open "c:\My Documents\sample.txt" For Input
As #1 Input #1, variable1
Text1.Text
= variable1
Close #1

End Sub
```

* This program will open the sample.txt file and display its contents in the Text1 textbox.

Back to contents page]

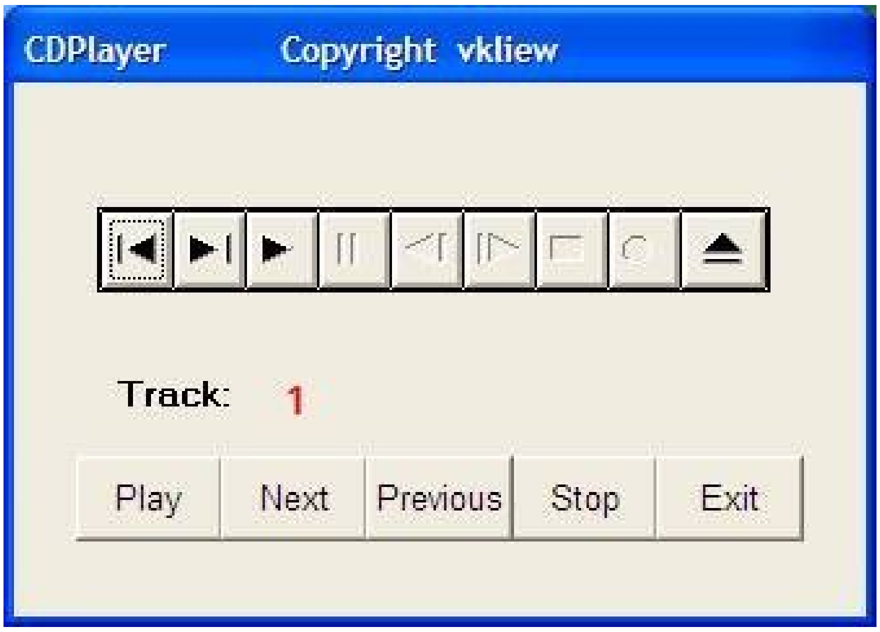
Lesson 15: Creating Multimedia Applications

You can create various multimedia applications in VB that could play audio CD, audiofiles, VCD , video files and etc.
To be able to play multimedia files or multimedia devices, you have to insert Microsoft Multimedia Control into your VB applications
that you are going to create. However, Microsoft Multimedia Control is not normally included in the startup toolbox, therefore you need
to add the MM control by pressing Ctrl+T and select it from the components dialog box that is displayed.

15.1 Creating a CD player

(a) The Interface.

¡¡



¡¡

First of all, you place a Multimedia control into your form and rename it as any name of your choice. Here I use myCD to replace the default name MMControl1. Next, you can put two labels on your form, change caption of the left label to **Track** and rename the one on the right to trackNum and make its caption invisible(this lable is to display CD track numbers at runtime.). Finally, put five command buttons in your form and name them as Play, Next, Previous, Stop and Exit. You can also choose to make the MM Control visible or invisible at runtime. If you choose to make it visible,you could play the CD using the buttons available on the control itself or you can click on the buttons at the bottom that are created by you.

(b) The Code

```
Private Sub Form_Load()
¡®To position the page at the center
Left = (Screen.Width ¨C Width) \ 2 Top
= (Screen.Height ¨C Height) \ 2
¡®Open the CD
myCD.Command = ¡°Open¡±

End Sub
Private Sub myCD_StatusUpdate()
¡®Update the track number
trackNum.Caption = myCD.Track
End Sub

Private Sub Next_Click()
myCD.Command = ¡°Next¡±
End Sub
```

```
Private    Sub    Play_Click()
myCD.Command = ¡°Play¡±

End Sub

Private Sub Previous_Click()
myCD.Command  =  ¡°Prev¡±
End Sub

Private Sub Stop_Click()
myCD.Command = ¡°Stop¡± End
Sub
Private Sub
Exit_Click() End
End Sub
```

[Back to Content Page]

¡¡

Lesson 16: Creating Multimedia Applications-Part 2
In previous lesson, we have programmed a CD player. Now, with some modifications, we will transform the CD player into an audio file player. This player will be created in such a way that it could search for wave and midi files in your drives and play them.

In this project, you need to insert a ComboBox, a DriveListBox, a DirListBox, a TextBox and a FileListBox into your form.I Shall briefly discuss the function of each of the above controls.
Besides, you must also insert Microsoft Multimedia Control(MMControl) in your form , you may make it visible or invisible. In my program, I choose to make it invisible so that I could use the command buttons created to control the player.

- ComboBox- to display and enable selection of different type of files.
- DriveListBox- to allow selection selection of different drives available on your PC.
- DirListBox - To display directories
- TextBox - To display selected files
- FileListBox- To display files that are available

Relevant codes must be written to coordinate all the above controls so that the application can work properly. The program should flow in the following logical way:

Step 1: User choose the type of files he wants to play.

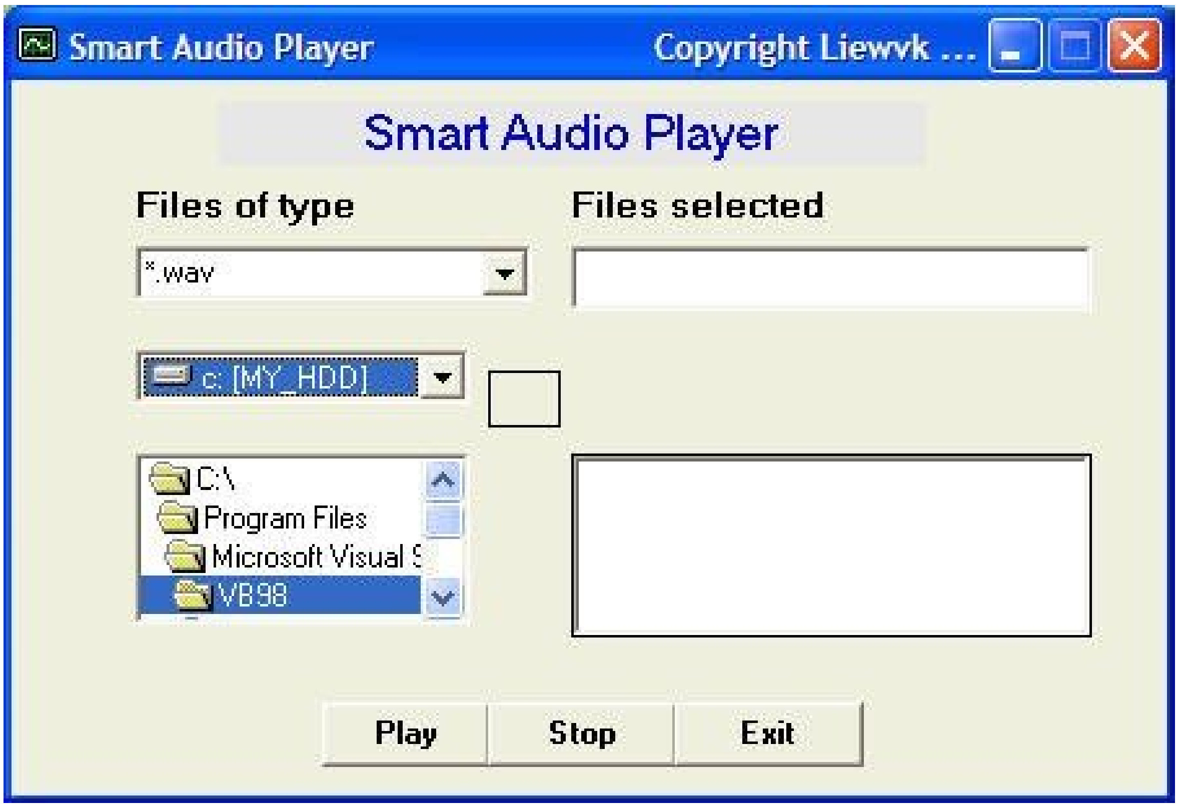
Step2:User selects the drive that might contains the relevant audio files.

Step 3:User looks into directories and subdirectories for the files specified in step1. The files should be displayed in the FileListBox.

Step 4: User selects the files from the FileListBox and click the Play button.

Step 5: User click on the Stop to stop playing and Exit button to end the application.

The Interface



¡¡

The Code

Private Sub
Combo1_Change() ' to
determine file type

If ListIndex = 0
Then
File1.Pattern =
("*.wav") ElseIf
ListIndex = 1
Then
File1.Pattern =
("*.mid") Else
Fiel1.Pattern
= ("*.*")
End If

End Sub

Private Sub Dir1_Change()

'To change directories and subdirectories(or folders and subfolders)

```
File1.Path = Dir1.Path
```

```
If Combo1.ListIndex =
0 Then File1.Pattern =
("*.wav")
ElseIf Combo1.ListIndex = 1
Then File1.Pattern =
("*.mid")
Else
File1.Pattern
= ("*.*")
End If
End Sub

Private Sub

Drive1_Change() 'To

change drives

Dir1.Path = Drive1.Drive

End Sub

Private Sub File1_Click()
If Combo1.ListIndex =
0 Then File1.Pattern =
("*.wav")
ElseIf Combo1.ListIndex = 1
Then File1.Pattern =
("*.mid")
Else
File1.Pattern
= ("*.*")
End If

If Right(File1.Path, 1) <> "\" Then
filenam = File1.Path + "\" +
File1.FileName Else
filenam = File1.Path +
File1.FileName End If
Text1.Text

= filenam

End Sub


Private Sub Form_Load()

'To center the Audioplayer startup page
```

```
Left = (Screen.Width -
Width) \ 2 Top =
(Screen.Height - Height) \ 2
Combo1.Text = "*.wav"
```

```
Combo1.AddItem
"*.wav"
Combo1.AddItem
"*.mid"
Combo1.AddItem
"All files"

End Sub

Private Sub

AudioPlayer_Click() End

Sub


Private Sub play_Click()

'To play WaveAudio file or Midi File


Command2_Click
If Combo1.ListIndex = 0 Then
AudioPlayer.DeviceType =
"WaveAudio" ElseIf
Combo1.ListIndex = 1 Then
AudioPlayer.DeviceType =
"Sequencer" End If
AudioPlayer.FileName =
Text1.Text
AudioPlayer.Command =
"Open" AudioPlayer.Command
= "Play"

End Sub

Private Sub stop_Click()
If AudioPlayer.Mode = 524 Then Exit
Sub If AudioPlayer.Mode <> 525 Then
AudioPlayer.Wait = True
AudioPlayer.Command = "Stop"
End If
AudioPlayer.Wait = True
AudioPlayer.Command =
"Close" End Sub

Private Sub
Exit_Click()
End
End Sub
¡¡
```

[Back to Content Page]

Lesson 17: Creating Multimedia Applications-Part 3

In lesson 16, we have created an audio player. Now, with some modifications, we will transform the audio player into a picture viewer. This player will be created in such a way that it could search for all types of graphics your drives and play them.

Similar to the previous project, in this project, you need to insert a ComboBox, a DriveListBox, a DirListBox, a TextBox and a FileListBox into your form. I Shall brieflyexplain again the function of each of the above controls.

- ComboBox- to display and enable selection of different type of files.
- DriveListBox- to allow selection selection of different drives available on your PC.
- DirListBox - To display directories
- TextBox - To display selected files
- FileListBox- To display files that are available

Relevant codes must be written to coordinate all the above controls so that the application can work properly. The program should flow in the following logical way:

Step 1: User choose the type of files he wants to play.

Step2:User selects the drive that might contains the relevant graphic files.

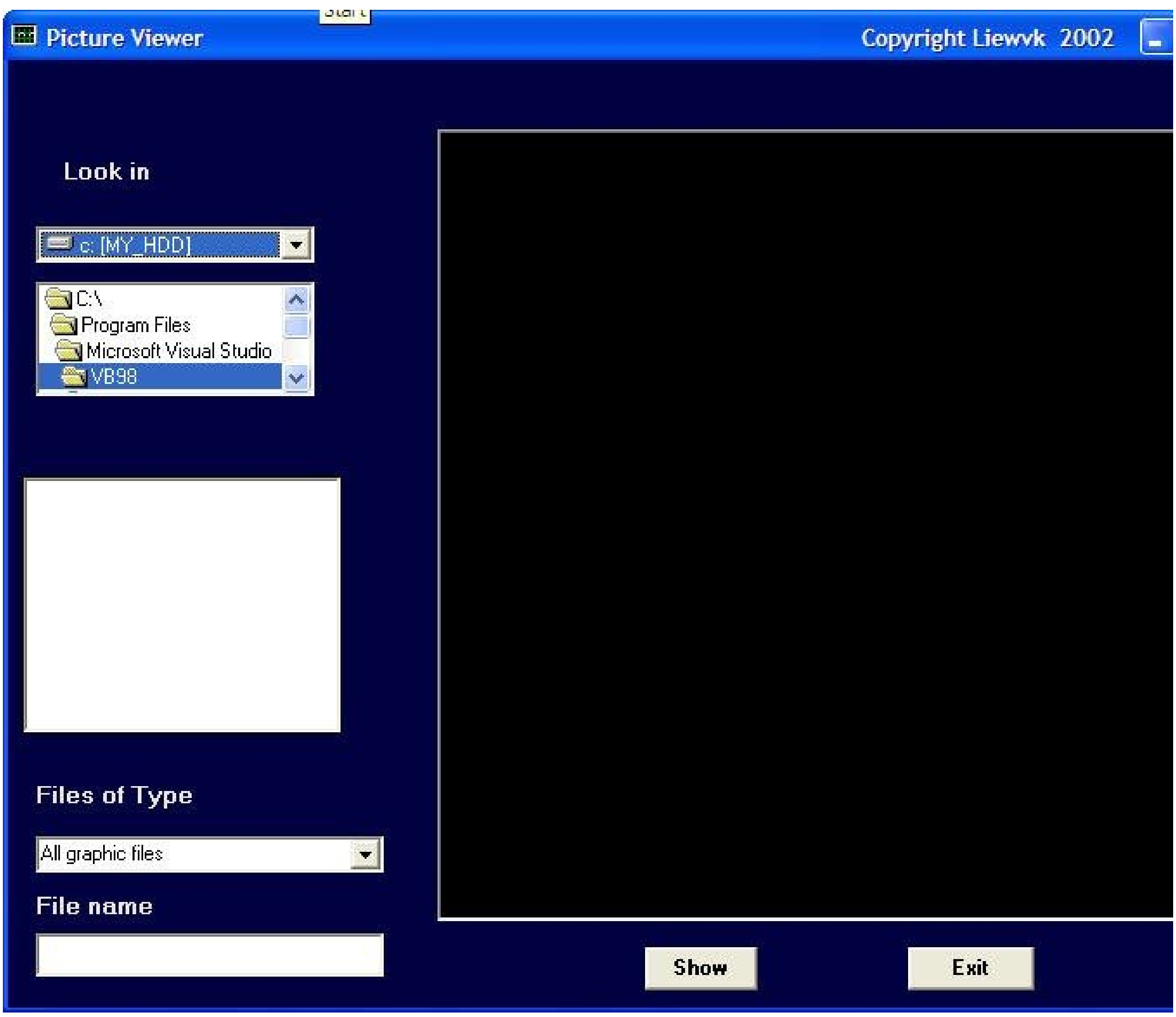
Step 3:User looks into directories and subdirectories for the files specified in step1. The files should be displayed in the FileListBox.

Step 4: User selects the files from the FileListBox and click the Show

button. Step 5: User click on Exit button to end the application.

¡¡

The Interface



[Test run the program]

The Code

Private Sub Form_Load()
Left = (Screen.Width
- Width) \ 2 Top =
(Screen.Height -
Height) \ 2
Combo1.Text = "All graphic
files" Combo1.AddItem "All
graphic files" Combo1.AddItem
"All files"
End Sub
¡¡

```
Private Sub
Combo1_Change()
If ListIndex = 0
Then
```

```
File1.Pattern =
("*.bmp;*.wmf;*.jpg;*.gif")
Else
Fiel1.Patte
rn =
("*.*")
End If
End Sub

Private Sub

Dir1_Change()

File1.Path = Dir1.Path
File1.Pattern = ("*.bmp;*.wmf;*.jpg;*.gif")

End Sub

Private Sub
Drive1_Change()
Dir1.Path = Drive1.Drive
End Sub

Private Sub
Exit_Click()
End
End Sub

Private Sub File1_Click()
If Combo1.ListIndex = 0
Then File1.Pattern =
("*.bmp;*.wmf;*.jpg;*.gif")
Else
File1.Patte
rn =
("*.*")
End If
If Right(File1.Path, 1) <> "\" Then
filenam = File1.Path + "\" +
File1.FileName Else
filenam = File1.Path +
File1.FileName End If
Text1.Text

= filenam

End Sub

Private Sub play_Click()
MMPlayer.FileName =
Text1.Text
End Sub
```

```
Private Sub show_Click()
If Right(File1.Path, 1) <> "\" Then

filenam = File1.Path + "\" +
File1.FileName Else
filenam = File1.Path +
File1.FileName End If
```

```
picture1.Picture =
LoadPicture(filenam) End
Sub
```

[Back to Content Page]

¡¡

Lesson 18: Creating Multimedia Applications-Part 4: A Multimedia Player
In lesson 16, we have created an audio player. Now, with some modifications, we will transform the audio player into a multimedia player that could play all kinds of movie files besides audio files. This player will be created in such a way that it could search for all types of graphics your drives and play them.

In this project, you need to insert a ComboBox, a DriveListBox, a DirListBox, a TextBox ,a FileListBox and a picture box(for playing movie) into your form. I Shall briefly discuss the function of each of the above controls. Besides, you must also insert Microsoft Multimedia Control(MMControl) in your form , you may make it visible or invisible. In my program, I choose to make it invisible so that I could use the command buttons created to control the player.

- ComboBox- to display and enable selection of different type of files.
- DriveListBox- to allow selection selection of different drives available on your PC.
- DirListBox - To display directories
- TextBox - To display selected files
- FileListBox- To display files that are available

Relevant codes must be written to coordinate all the above controls so that the application can work properly. The program should flow in the following logical way:

Step 1: User choose the type of files he wants to play.

Step2:User selects the drive that might contains the relevant audio files.

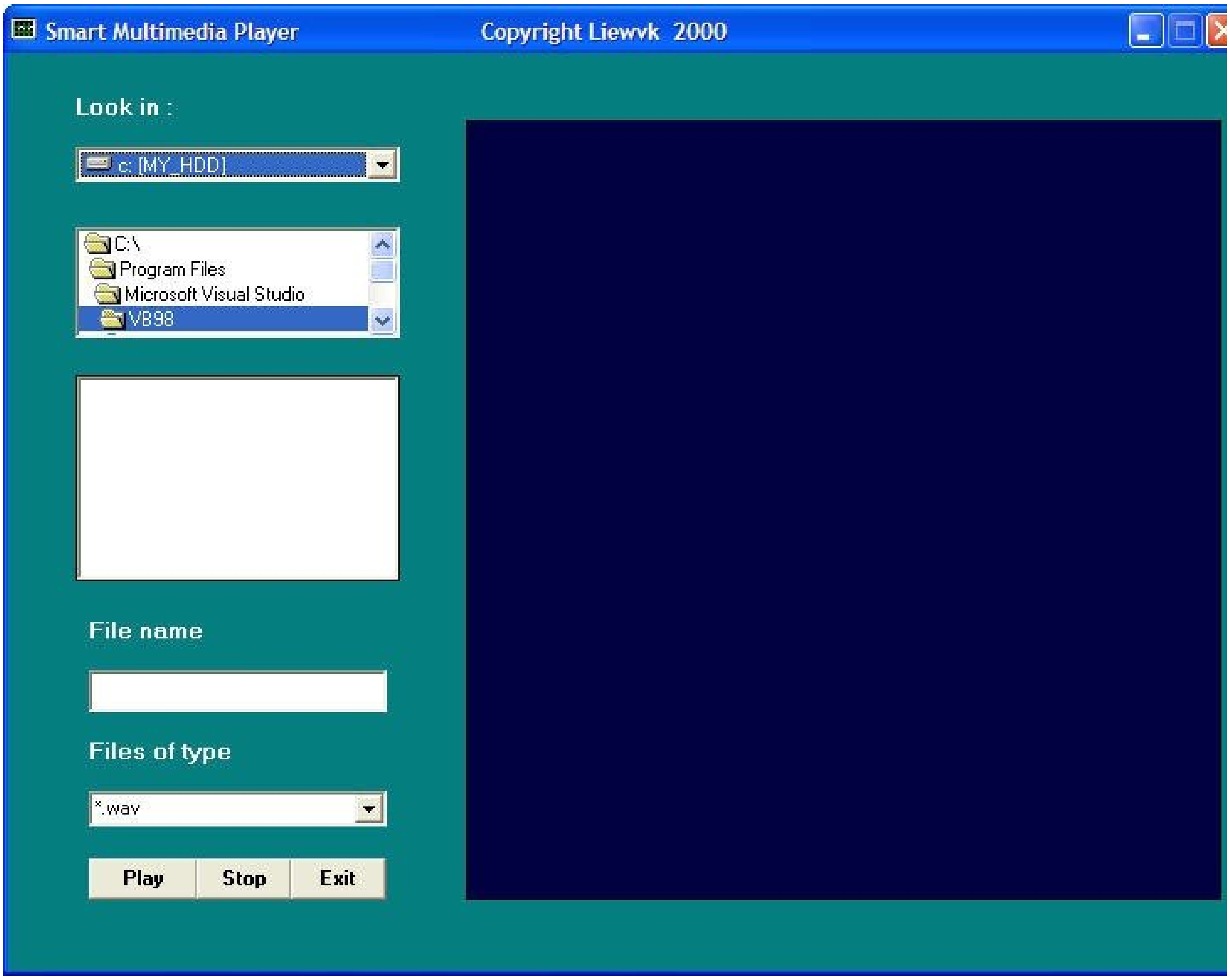
Step 3:User looks into directories and subdirectories for the files specified in step1. The files should be displayed in the FileListBox.

Step 4: User selects the files from the FileListBox and click the Play button.

Step 5: User click on the Stop to stop playing and Exit button to end the application.

¡¡

The Interface



¡¡

[Test run the program]

¡¡

The Code
Private Sub Form_Load()
Left = (Screen.Width -
Width) \ 2 Top =
(Screen.Height - Height) \ 2
Combo1.Text = "*.wav"
Combo1.AddItem "*.wav"
Combo1.AddItem "*.mid"
Combo1.AddItem
"*.avi;*.mpg"
Combo1.AddItem "All files"

```
End Sub

Private Sub
Combo1_Change() If
ListIndex = 0 Then
File1.Pattern = ("*.wav")
ElseIf ListIndex = 1
Then File1.Pattern =
("*.mid") ElseIf
ListIndex = 2 Then
File1.Pattern =
("*.avi;*.mpg") Else
Fiel1.Pattern
= ("*.*")
End If

End Sub

Private Sub
Dir1_Change()
File1.Path =
Dir1.Path
If Combo1.ListIndex =
0 Then File1.Pattern =
("*.wav")
ElseIf Combo1.ListIndex = 1
Then File1.Pattern =
("*.mid")
ElseIf Combo1.ListIndex = 2
Then File1.Pattern =
("*.avi;*.mpg") Else
File1.Pattern
= ("*.*")
End If
End Sub

Private Sub
Drive1_Change()
Dir1.Path =
Drive1.Drive
End Sub

Private Sub
Exit_Click()
End
End Sub

Private Sub File1_Click()
If Combo1.ListIndex =
0 Then File1.Pattern =
("*.wav")
ElseIf Combo1.ListIndex = 1
Then File1.Pattern =
("*.mid")
ElseIf Combo1.ListIndex = 2 Then
```

```
File1.Pattern =
("*.avi;*.mpg") Else
File1.Pattern
= ("*.*")
End If

If Right(File1.Path, 1) <> "\" Then
filenam = File1.Path + "\" +
File1.FileName Else
filenam = File1.Path +
File1.FileName End If
Text1.Text

= filenam

End Sub


Private Sub

MMPlayer_Click() End

Sub

Private Sub

Picture1_Click() End

Sub

Private Sub play_Click()
MMPlayer.FileName =
Text1.Text
MMPlayer.Command =
"Open"
MMPlayer.Command =
"Play"
MMPlayer.hWndDisplay =
videoscreen.hWnd End Sub

Private Sub stop_Click()
If MMPlayer.Mode = 524 Then Exit
Sub If MMPlayer.Mode <> 525
Then MMPlayer.Wait = True
MMPlayer.Command = "Stop"
End If
MMPlayer.Wait = True
MMPlayer.Command =
"Close"
End Sub
```

[Back to Content Page]

¡¡

¡¡

Your Go

Search: Software Go Browse: All Categories

You clicked on this item...

Icon **Microsoft Visual Basic .NET Standard**
by Microsoft
Average Customer Review: 2.5 out of
Usually ships in 24 hours
This item ships for **FREE** with **Super Saver Shipping**. See details.

List Price: ~~$109.00~~
Price: $99.99
You Save: $9.01 (8%)
Special Offers: $20.00
Price After Special Offers: $79.99

Amazon.com.uk
More than just a programming language, Visual Basic .NET is a visual development tool for Windows. Building an application is a matter of first creating a visual interface using the drag-and-drop form designer, and then writing code to bring the interface to life. Visual Basic is popular with... Read more

Add to Shopping Car
Add to my Wish List
Or buy used: $85.00

You may also be interested in these items...

Show items that are:

● Closely Related
○ Moderately Related
○ Loosely Related

it