



Southern Technical University

Amara Technical Institute

Computer Systems Dept

Class: Second Year

VISUAL BASIC

Lecturer

Abbas J. Kadhim

2019-2020

CHAPTER Two

Visual Basic's Visual Nature

Introduction

Visual Basic is more than just a programming language. The secret to Visual Basic is in its name: visual. Older programming languages, such as BASIC, worked well in a text-only computing environment, but such languages do not support the graphical interface needed for today's computers. Visual Basic programming process requires interacting with the Visual Basic visual environment as we will see in this chapter.

2-1: Starting Visual Basic.

After installing Microsoft Visual Studio 6.0, choose Visual Basic 6.0 from the start menu. As soon as we start Visual Basic, the Application wizard is there to help. The New Project dialog box, shown in Figure (2-1) will appear. Create a standalone program by selecting the Standard EXE icon. The icon is named to represent the resulting application's filename extension (.exe for executable) if you compile the application you create. Even if you will not be compiling your application right away, the Standard EXE icon is the one you'll

choose most of the time while learning Visual Basic. A standard EXE application is an application that you can compile or run interpretively.

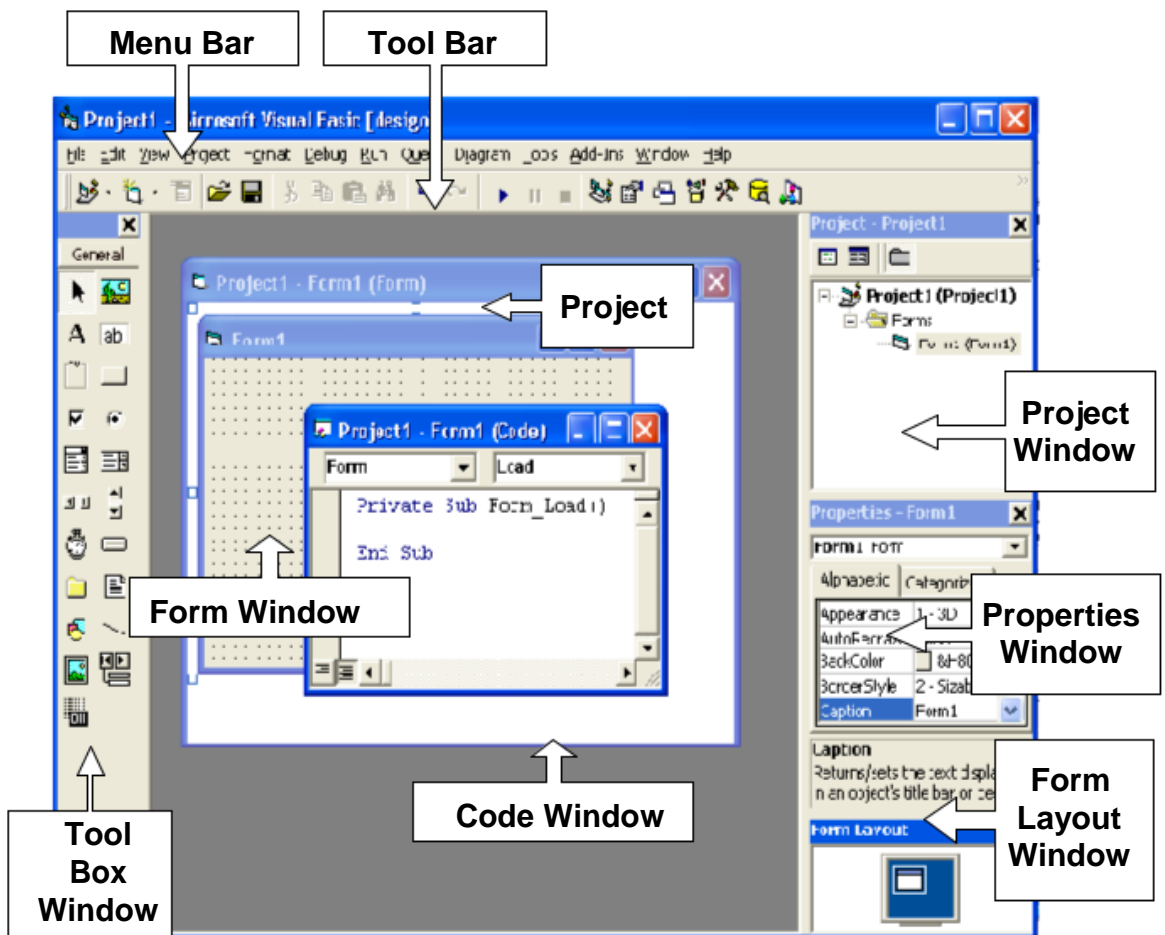


Figure(2-1): The New Project dialog box

2-2: Visual Basic environment

The Visual Basic environment contains several windows with which you will work as you build application. Figure (2-2) illustrates the major parts of the Visual Basic screen which consists of the following windows:

1. Form Window
2. Toolbox Window
3. Properties Window
4. Code Window
5. Form Layout Window
6. Project Explorer Window
7. Menu Bar
8. Tool Bar

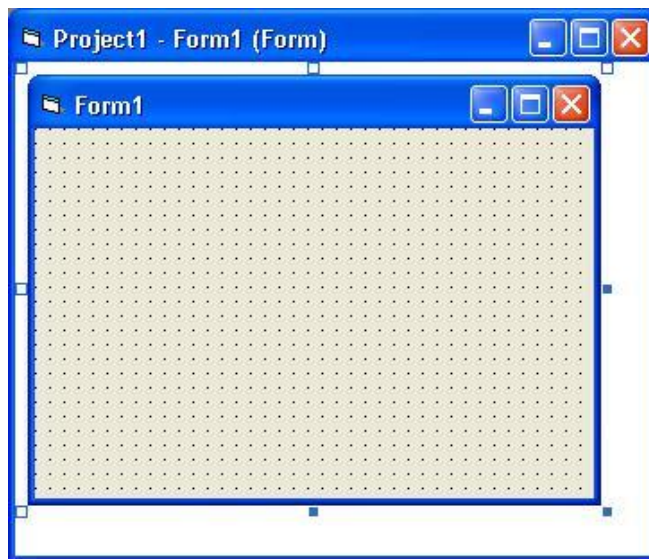


Figure(2-2) Visual Basic environment

2-2-1: The Form window

Most of your work goes on inside the Form window. You'll design all your application's forms, which are the background windows that your users see, in the central editing area where the Form window appears. You can resize the Form window to make the windows you create in your applications as large or small as needed. (Scrollbars appear to let you scroll the Form window if you need to see parts of the forms that run off the screen or underneath other Visual Basic windows).

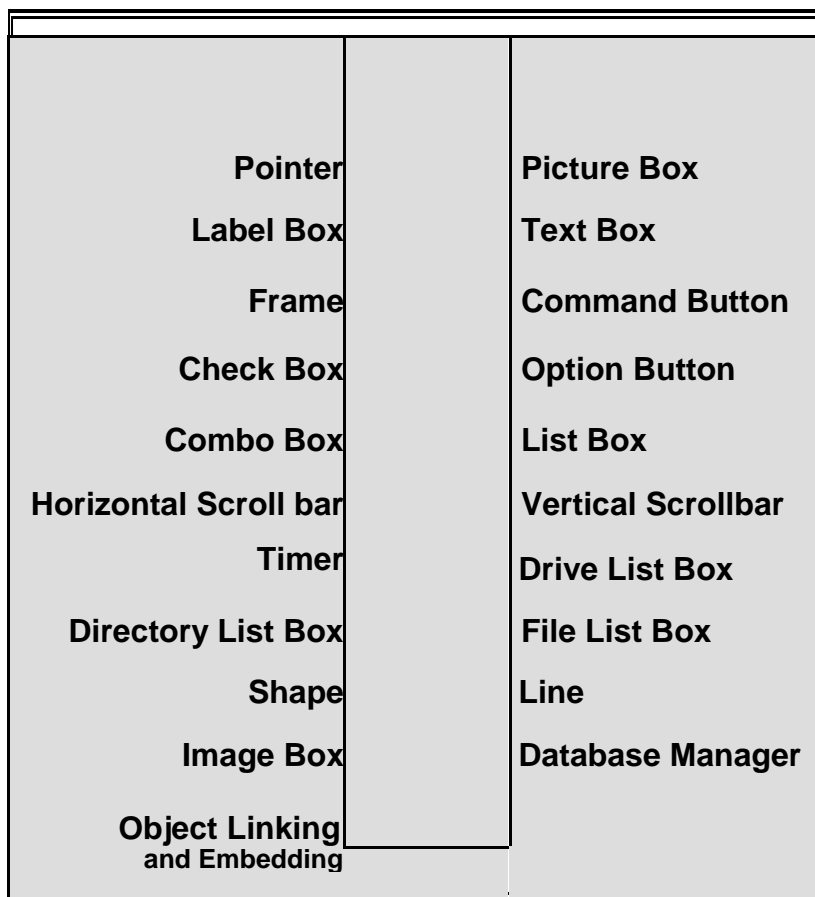
Keep in mind that an application may contain multiple forms; you can display one or more of those forms in their own Form window editing areas, as shown in Figure (2-3). The active form is the form with the highlighted title bar in its window. Activate a form by clicking anywhere within the window or on the title bar.



Figure(2-3): The Form

2-2-2: Toolbox window

The Toolbox window, typically called the toolbox, is a collection of tools that act as a repository of controls you can place on a form. These controls are dragged to the form window to perform a specific job. Figure (2-4) shows the most common collection of toolbox tools that we use. The function of each control tool is listed in the next page.



Figure(2-4): The Toolbox

1. The Pointer Tool

The pointer tool is the only toolbox item that is not a control. About the only use for the pointer tool is to eliminate the crosshair mouse cursor that appears when you select any of the other controls in the Toolbox window.

2. Picture Box

This tool is used to view pictures in addition it can be a container for another tools.

3. Label Box

The Label Control is one of the simplest controls to work with. With the label control, you can add descriptive text to the form in any location by using different styles and sizes of fonts.

4. Text Box

Unlike the Label Box, the user can change values within a text box control. You can get answers from the user by using text box controls.

5. Frame

The Frame Control enables you to group items together on a form. The group works almost like a miniform within the form.

6. Command Button

You have seen command buttons in almost every Windows program, including Visual Basic and the currently-running application. Command buttons give users pushbutton access to events that you place within an application.

7. Check Box

Check box controls offer multiple-choice values from which the user can select. Once the user selects one or more check boxes, your program can analyze the selected check boxes and make decisions based on those responses.

8. Option Buttons

Unlike check boxes, option buttons give your users a list from which to choose, but they can select exactly one option out of the list.

9. Combo List Box

A Dropdown Combo list is one of three kinds of lists that you can provide for your user. The Dropdown list saves room on the screen by consuming only a single line on the form until the user opens the list to display the rest of the items in it.

10. List Box

If you want users to select from a choice of options that you have supplied and you want to prevent them from adding additional items to the list, use list boxes.

11. Horizontal Scroll Bar

Use this tool to choose a value among range of values horizontally. Also you can scroll the hide windows horizontally.

12. Vertical Scroll Bar

Use this tool to choose a value among range of values vertically. Also you can scroll the hide windows vertically.

13. Timer

With this tool we can execute many operations in specific periods.

14. Drive List Box

If we use this tool we can list all the drivers found in the used computer.

15. Directory List Box

This tool lists all the directories (Folders) found in the specific driver.

16. File List Box

This tool lists all the files found in the specific folders.

17. Shapes

By using this tool we can draw many shapes (like circles , squares , rectangles, ... etc) .

18. Line

By using this tool we can draw lines in the form.

19. Image Box

The image control is one of two controls (the picture box is the other) that display graphic images.

20. Database manager

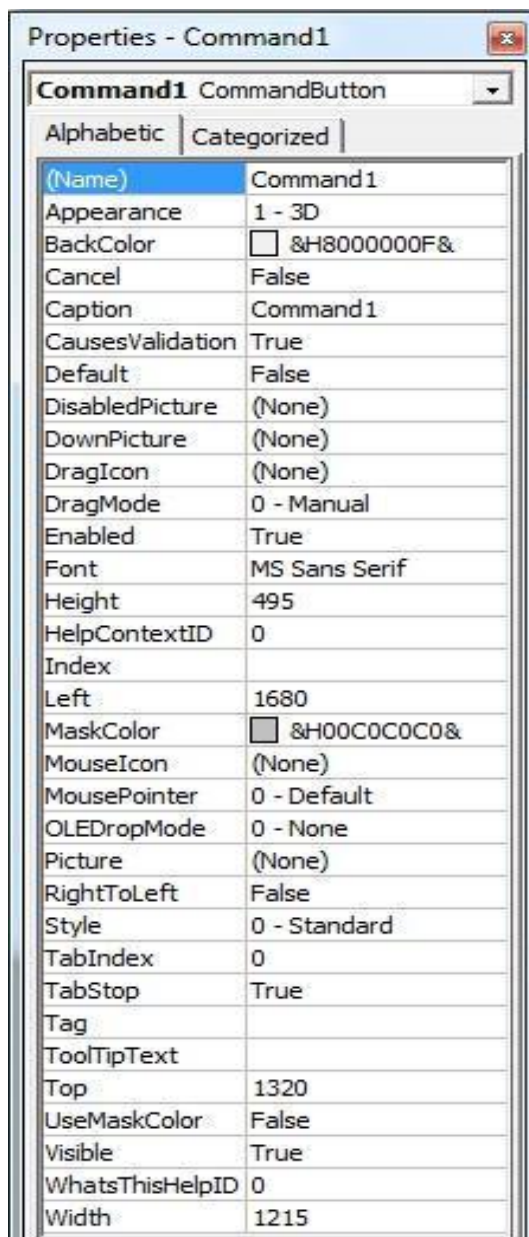
We use this tool when we want to build a database used in many applications.

21. Object Linking and Embedding (OLE)

Allow this tool to receive many applications in the visual basic environment (like MS Excel , MS Word , ... etc).

2-2-3: Properties Window

A form can hold many controls. As you add controls to a form, you can select a control by clicking the control. When you select a control, the Properties window changes to list every property related to that control. So when you add a control to a Visual Basic application, Visual Basic sets the control's initial property values. When you display the Properties window for a control, you can modify its property values. Figure(2-5) beside shows a Properties window listing some of the properties for a Label control. Notice that the name, type, and description in the Property window reflect the selected control. To assign a value to a property, select the property and type a new value. Sometimes a drop-down list box



Figure(2-5): Properties Window

will appear when you can select one of an established set of values for that property.

Now we will list the wide used properties and its function:

- **Alignment:** Determines whether text on the control, such as a label or command button, is left-justified, centered, or right-justified on the control.
- **AutoSize:** If **True** the control will take an automatic size to fit its contents.
- **BackColor:** Specifies the color of the control's background, which you select from a palette of colors when you open the property's drop-down list box of colors.
- **BorderStyle:** Determines whether the control has a border around it.
- **Caption:** Lists the text displayed on the control appears in the form window.
- **Enabled:** Either **True** if you want the control to respond to the user or **False** if you want the control not to respond to the user.
- **Font:** Displays a Font dialog box from which you can set various font properties, such as size and style, for a control's text.
- **ForeColor:** Specifies the color of the control's foreground, which you select from a palette of colors when you open the property's drop-down list box of colors.
- **Height:** Specifies the height of the control.
- **Left:** Indicates the starting from the left edge of the form where the control appears. For a form, the **Left** property specifies the distance from the left edge of the screen.

- **MousePointer:** Determines the shape of the mouse cursor when the user moves the mouse over the control at runtime.
- **Name:** Specifies the name of the control that will appear in the code window.
- **Text:** Lists the text displayed on the control. Used to input and output texts.
- **Top:** Is the starting distance from the top edge of the form where the control appears. For a form, the **Top** property describes distance from the top edge of the screen.
- **Visible:** Set by a drop-down list box, this property is **True** if you want the control to be visible on the form or **False** if you want the control to be hidden from view.
- **Width:** Specifies the width of the control.
- **ScaleMode:** This property can be found on the form properties window. This property gives you a choice of eight choices according to your measuring unit. These units are:
 1. Twip = 0.567 cm
 2. Point = 0.72 inch
 3. Pixel = 1 form point
 4. Character = 120 Twip vertically , 240 Twip Horizontally
 5. Inch = 2.54 cm
 6. Millimeter (mm)
 7. Centimeter (cm)
 8. User (the user can define it)

The default unit is "Twip" but the user can change it according to his requirements. Put in your mind that the distance between any two points on the form = 120 Twip = 6 Point = 8 Pixels = 1 character H, 0.5 Character V = 0.083 inch = 2.117 mm = 0.212 cm.

- **ScaleWidth and ScaleHeight:** Returns the width and the height for the inner area for the form. The other two properties "Width" and "Height" returns the width and the height for the outer area for the form. So the default form dimensions will be:

Width=4800 Twip

ScaleWidth=4680 Twip

Height=3600 Twip

ScaledHeight=3090 Twip

Note

Do preface each object name you assign with a three-letter prefix that describes the object. Then when you later look at the list of objects, you not only know the object's name but also its type (command button, text box, form, or whatever). **Table(2-1)** below lists common prefixes used for Visual Basic object.

Prefix	Object type
cbo	Combo box
chk	Check box
cmd	Command button
dir	Directory list box
drv	Drive list box
fil	File list box
fra	Frame
frm	Form
hsb	Horizontal scrollbar
img	Image
lbl	Label
lin	Line
lst	List box
mnu	Menu
ole	OLE client
opt	Option button
pic	Picture box
shp	Shape
tmr	timer

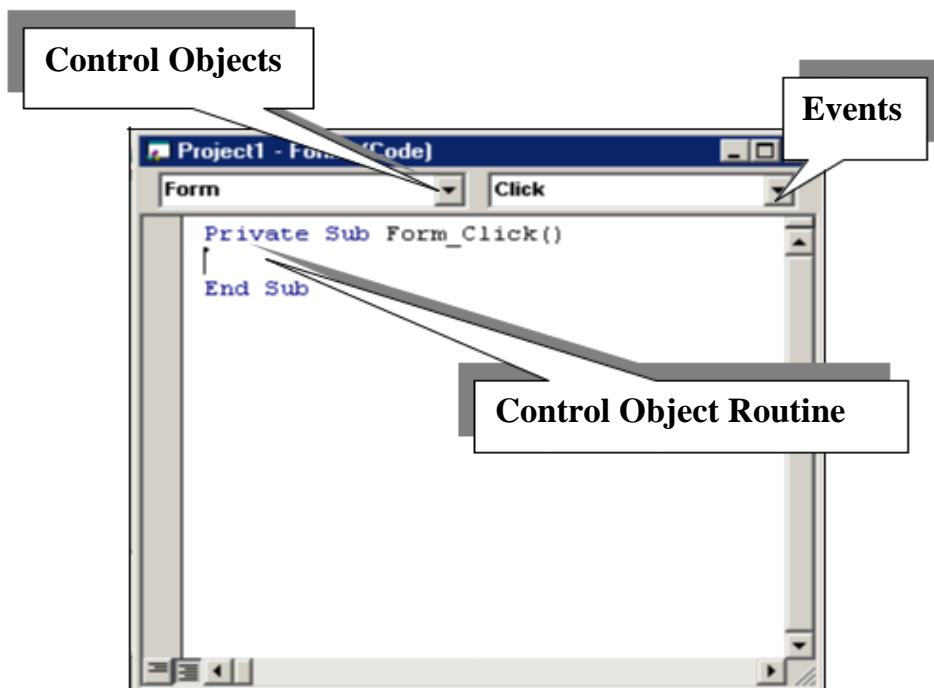
Table(2-1): Common used prefix

2-2-4: Code window

Code Unlike most other programming languages, you do not have to write much code as you develop applications in Visual Basic. The more advanced the application needs to be, the more code you will have to write to tie things together. The visual parts of Visual Basic, however, eliminate much of the code that you would have to write if you were still working in a text-based environment. Although you should not

expect to understand anything just yet, Figure (2-6) below shows a Code window that contains two list box and Control Object Routine.

The right list box contains all control objects used in the project in addition to form object. The left list box lists the Events which is a callback mechanism. With it, objects can notify users that something interesting has happened and table(2-2) shows the most common events used in this book. The Control Object Routine is the program that we will write it to perform a specific job. Usually the Visual Basic program starts with the statement "Private Sub" and ends with the statement "End Sub".



Figure(2-6): The Code Window

Event	Event Time
Click	Clicking the mouse left button once
DbClick	Clicking the mouse left button twice.
MouseMove	Moving the mouse over the control object.
MouseDown	During Clicked the mouse left button.
MouseUp	Releasing the mouse left button.
DragDrop	Moving the object using the mouse.
KeyPress	Pressing any key from the keyboard.
KeyDown	During key pressing and before releasing it.
KeyUp	After pressing the key

Table(2-2): The most common code window events

Note

There are two other ways to display the Code window; you can select View Code to see the window or you can also press F7 to display the code.

2-2-5: Form Layout Window

The Form Layout window shown in Figure(2-7) is an interesting little window connected closely to the Form window, because the Form Layout window shows you a preview of the Form window's location.

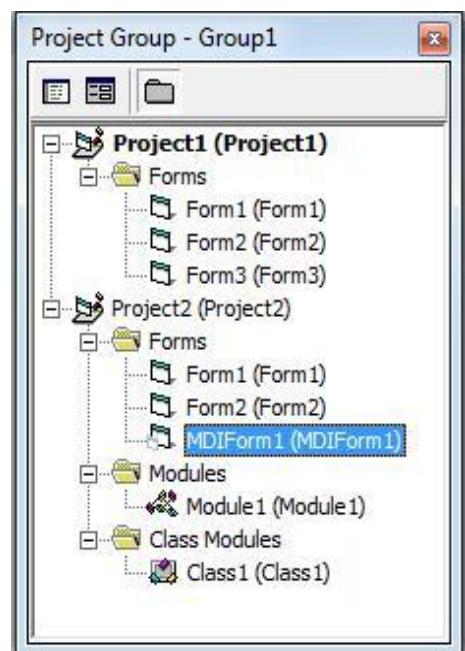


Figure(2-7): Form Layout

If one or more forms appear in your Form window, thumbnail sketches of those forms will also appear in the Form Layout window. The window shows you where each of the forms will appear on the screen when you user runs the application and, through using the program, views the various forms.

2-2-6: Project Explorer Window

Use the Project window to manage your application's components. As Figure(2-8) shows, a windows program, more accurately called an application, can consist of several files. Before you compile a Windows program, the number of Visual Basic-related files can get even more numerous. The Project window enables you to manage all those components and bring the component you want to work with to the editing area



Figure(2-8): Project Explorer

2-2-7: Menu Bar

The top of the screen contains the menu bar and toolbar. The menu bar (Shown in Figure (2-9)) contains lists of pull-down menus with which you can manage your Visual Basic program. If you have

worked much with other Windows programs, you are already familiar with the File, Edit, View, Tools, and Help menu bar commands because they are similar across many Windows applications.

Figure below describes all the Visual Basic menu bar commands with which you will work and their advantages.



Figure(2-9): Menu Bar

- **File:** The File menu contains all file-related commands with which you can load and save Visual Basic applications. It also provides printing access for printed program descriptions as well as the Exit command.
- **Edit:** Programmers often use the commands on the Edit menu for copying, cutting, and pasting text and graphical controls among applications. The Edit commands also help you with the creation of your programs by supplying common search and replace actions.
- **View:** The View menu command enables you to control the viewing of your application's Code, Form, and Project windows, various routines that can appear inside the Code window, as well as the toolbar and toolbox.
- **Insert:** To insert additional objects, such as a second form, in your Visual Basic application.
- **Run:** When you complete an application, you can see the results of your work with the Run menu. The Run menu enables you to

execute programs, halt the execution, and resume the execution after a halt.

- **Tools:** To test your program and work with additional tools that comes with Visual Basic such as the Menu Editor. One of the most powerful features of Visual Basic is its debugging capability. With the Debug menu, you can execute a Visual Basic program one statement at a time.
- **Add-Ins:** To add additional components to Visual Basic. The default toolbox does not contain all the tools that you get with the Visual Basic Working Model. For example, you can add additional tools to your toolbar by selecting the Add-Ins option and locating the extra tools.
- **Help:** When you select from the Help menu with the Working Model edition of Visual Basic, you will get online help.

2-2-8: Tool Bar

The toolbar differs from the Toolbox window. The toolbar supplies quick push-button commands for common tasks. Figure (2-10) below shows the toolbar and labels each button on the toolbar. Many of the toolbar buttons represent menu commands. Instead of issuing a menu command or using a shortcut key, you can click a toolbar button with the mouse to perform the same task. So the toolbar contains quick access to many commands.

Note that not all of the toolbar buttons are dark some are grayed out, just as some of the pull-down menu bar commands are grayed out at times. Visual Basic knows that certain commands have to be activated at specific times within the program.

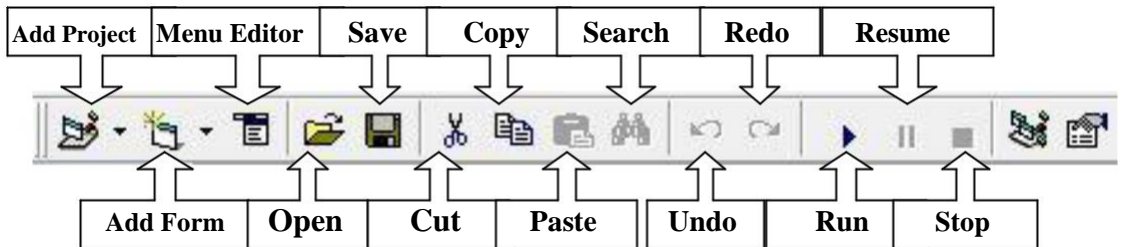


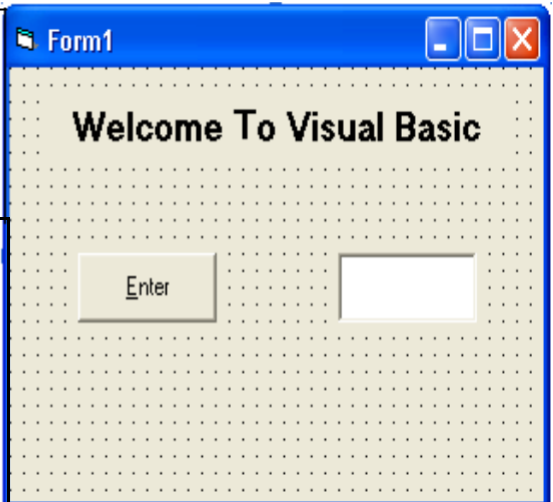
Figure (2-10): Tool bar

2-3: Applied Examples:

In the following few pages we will preview some examples that can help you to understand what visual basic is and how can we deal with this programming language. In these examples we show the solution forms before and after execution in addition to clarify the solution steps obviously.

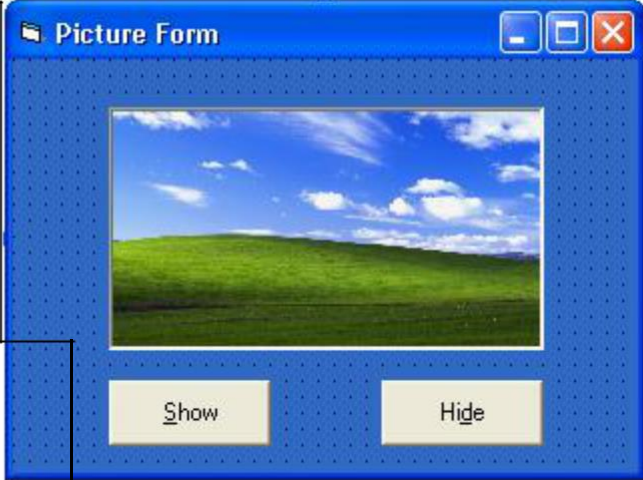
Example-1:

What would appear on the form when we put the following control objects? The units is Twip.

Name = lblWelcome Alignment = 2-Center Caption = Welcome To visual Basic AutoSize = True Font = Bold (14)		
Name = txtHello Text = Left = 2880 Top = 1320 Width = 1215 Height = 495	Name = cmdEnter Caption = &Enter Left = 600 Top = 1320 Width = 1215 Height = 495	

Example-2:

What would appear on the form when we put the following control objects?

Name = frmPicture Caption = Picture Form BackColor = HighLight ScaleMode = Pixel ScaleWidth =312 ScaleHeight = 206 Width = 4800 Height =3600		
Name = imgBliss Boarder Style = 1-Fixed Single Stretch = True Picture = C:\My Documents\... My Pictures\Bliss.Bmp Left = 48 Top = 24 Width = 217 Height = 121	Name =cmdHide Caption = Hi&de Left = 184 Top = 160 Width =81 Height = 33	Name =cmdShow Caption = &Show Left = 48 Top = 160 Width =81 Height = 33

Note: The character (&) "Ampersand" used in the **Caption** property used to put the letter underlined to make a shortcut to it.

Note: When you call a picture in the Image Box you must True the Stretch property before browsing the picture.

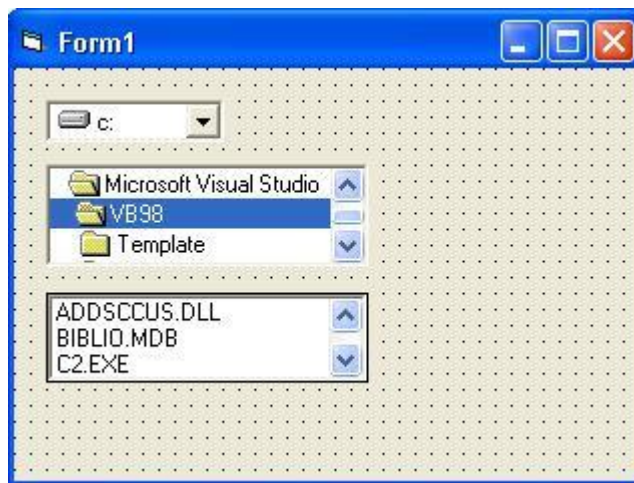
Example-3:

What would appear on the form when we put the following control objects? The unit is Millimeter.

Name = drv1
Left = 4.233
Top = 4.233
Width = 23.45
Height = 5.556

Name = dir1
Left = 4.233
Top = 12.7
Width = 42.59
Height = 13.49

Name = fil1
Left = 4.233
Top = 29.633
Width = 42.59
Height = 11.90



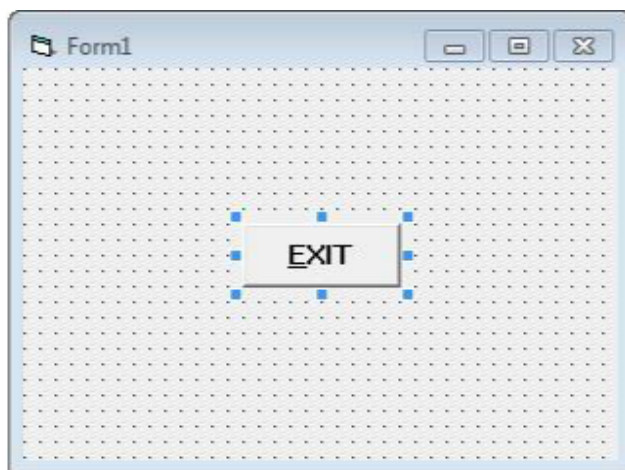
Example-4:

Design a project containing a single form with address "My First Program". A command button with name "Exit" is placed on this form. When we clicked this button the application will be stopped. Use the Twip unit.

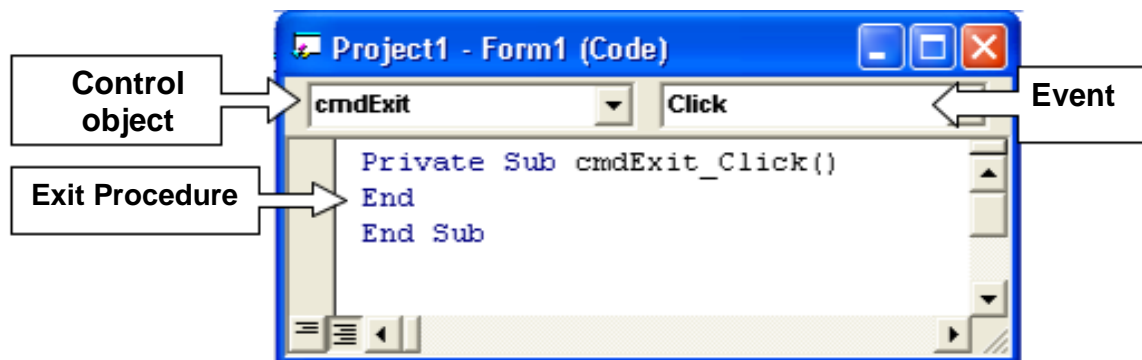
Solution Steps:

1. Change the form property "Caption" to "My First Program".
2. Put a command button on the form and change its properties as shown:

Name =cmdExit
Caption = &Exit
Top = 1200
Width =1215
Left = 1680
Height = 495



3. Open the code window for the "Exit" command and write the quit procedure (End) between the statements "Private Sub" and "End Sub".



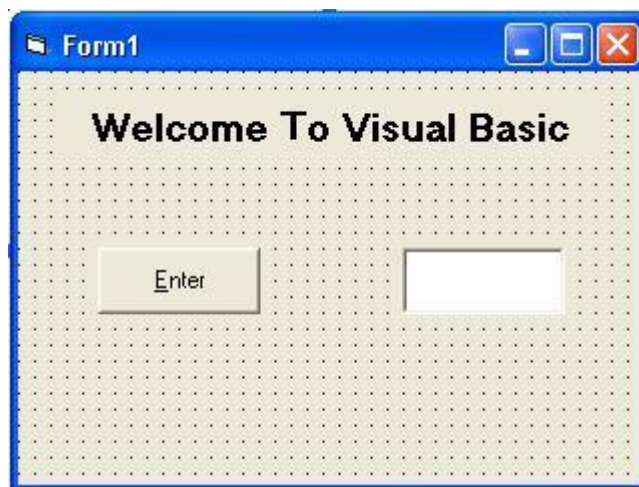
4. Run the project by pressing the blue triangle in the toolbar (or press F5). Now press the "Exit" command to quit from the application.

Example-6:

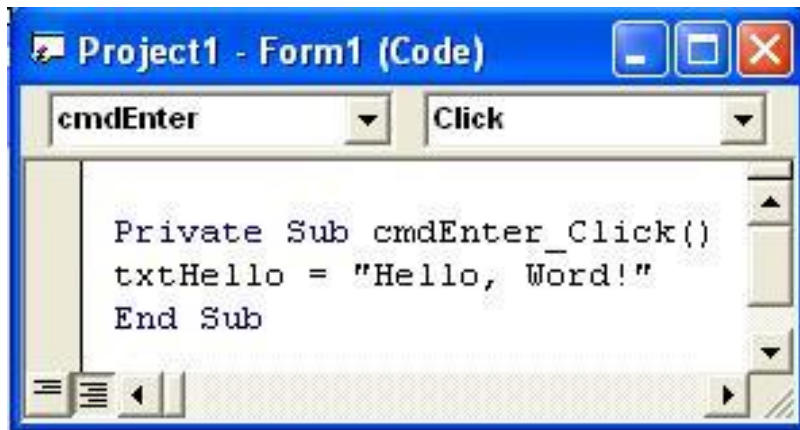
Write a VB program that display the phrase ("Hello, World!") in a Text Box when we click on the command button "Enter". The form also contains the label ("Welcome To Visual Basic").

Solution

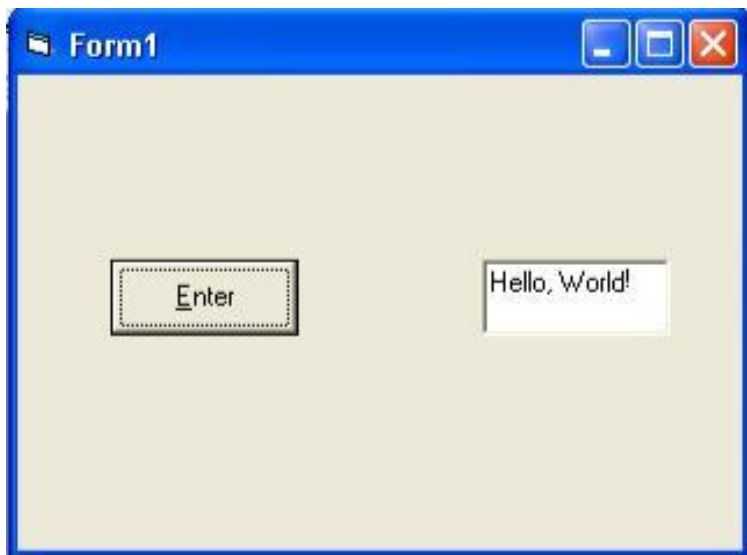
After setting the control objects, the form will look like this:



Double click over the "Enter" command to view the code window which we will write our program. As shown:



Run the project and click the command "Enter" (or press Alt+E) to execute the program.



2-4 Problems

1. Identify the control objects used to design the below forms and list the properties for each object.

Form1

list1

one
Two
Three
Four

list2

A
B
C
D

clear of

☒ non
☐ list1
☐ list2

Translate Remove End

2. Implement the following visual basic form:

Form2

Alphabets: A

Numbers: 1

Symbols: @

Others: NULL

ASCII Value

3. Implement the following two forms in the same visual basic project.

The image shows a Windows form titled "Form1". It contains four input fields arranged in a 2x2 grid. The top-left field is labeled "Amount of payment:", the top-right is "Annual Interest Rate:", the bottom-left is "Number of Payments:", and the bottom-right is "Present Value:". To the right of these fields is a large button labeled "Calculate". Below the input fields is a large text box labeled "Future Value:". The form has a standard Windows window border with minimize, maximize, and close buttons.

The image shows a Windows form titled "DirListBox Demo". It features a search bar at the top. Below the search bar is a directory list box displaying a list of folders. The folders are: E:\, تعليمي, تعلم, فيجوال بيسك, CodeBank, Visual Basic 6 Master Reference, D, and DirListBox. The "DirListBox" folder is currently selected and highlighted in blue. The form has a standard Windows window border with minimize, maximize, and close buttons.

4. What would appear on the form if we add the following control objects if you know that the unit was Twip.

Name = lblFirst Alignment = 2-Center Caption = My First Program AutoSize = True Font = Bold (20) Top = 081 Left = 600	Name = txtName Text = Left = 600 Top = 1500 Width = 1215 Height = 500	Name = cmdEnter Caption = &Enter Left = 2880 Top = 1500 Width = 1215 Height = 500
---	--	--

5. What would appear on the form if we add the following control objects if you know that the unit was Pixel.

Name = txtText Alignment = 2-Center Text = Visual Basic Left = 48 Top = 24 Width = 217 Height = 121	Name = optHide Caption = Hi&de Vlaue = False Left = 184 Top = 160 Width = 81 Height = 33	Name = optShow Caption = &Show Value = True Left = 48 Top = 160 Width = 81 Height = 33
---	--	--

6. What would appear on the form when we put the following control objects? The used unit is Twip.

Property	Control objects				
Name	drv	dir	fil	lbl	txt
Caption	-----	-----	-----	Browser	-----
Left	120	1440	3240	1440	1440
Top	840	840	840	120	2280
Width	1215	1575	1215	1575	1575
Height	315	990	990	222	375

CHAPTER THREE

Visual Basic Programming Language

Introduction

Visual Basic programming is one of the most enjoyable ways to program. Much of creating a Visual Basic program requires placing graphic objects on the screen and setting attributes for those objects that determine how the objects are to look and behave. Visual Basic is truly the only programming language today that beginning programmers can learn easily. In addition, Visual Basic allows advanced programmers to create powerful Windows applications. In this chapter and following chapters, we'll discuss some best statements and functions used in visual basic programs with the use of practices coding. All of these practices come from professional programmers, but of you must implement them to know and learn how to build complementary programs. In general, visual basic program contain of the following steps:

1. Declaring variables and constants
2. Input variables and constants
3. Using the assignment statements
4. Using mathematical and logical operations
5. Printing the results

3-1: Variables Assignment- Statement

Variables hold values that can change. A variable's value can change because a variable is nothing more than a storage area that can hold one value at a time. When you store a different value in a variable, the original value is replaced. So the variable is a temporary named storage area inside your program's memory that holds data. You are responsible for naming all variables in your code. Two different variables cannot have the same name within the same procedure because Visual Basic cannot distinguish between them. Unlike control properties that are already named, variables don't have names until you give them names. The assignment statement is the easiest way to store values in variables. Here is the format of the assignment statement:

Variable Name = Value
or : Control name . Property = Value

Where the terms:

Variable Name: The name of the suggested variable that we want to store the value.

Control name: The name of the control object.

Property: Control objects property that we want to change.

Value: The new value that we want to store it in the variable (or in the control object) and it can be any of the following:

- **A numerical value:** Any number
- **A literal value:** Any string (usually parenthesized by double quit ("--"))
- **A literal constants:** Usually started by the prefix (vb) (to see these literal constants see **Appendix-A**)
- **A mathematical expression:** Any mathematic operations.
- **A variable:** Another variable related with the original variable.
- **A logical expression:** Logical operations that its result True or False.
- **Functions:** These functions are prepared function usually found in Visual Basic library.

3-2: Experimental Projects

In the following few pages we will see some of most popular projects in windows applications.

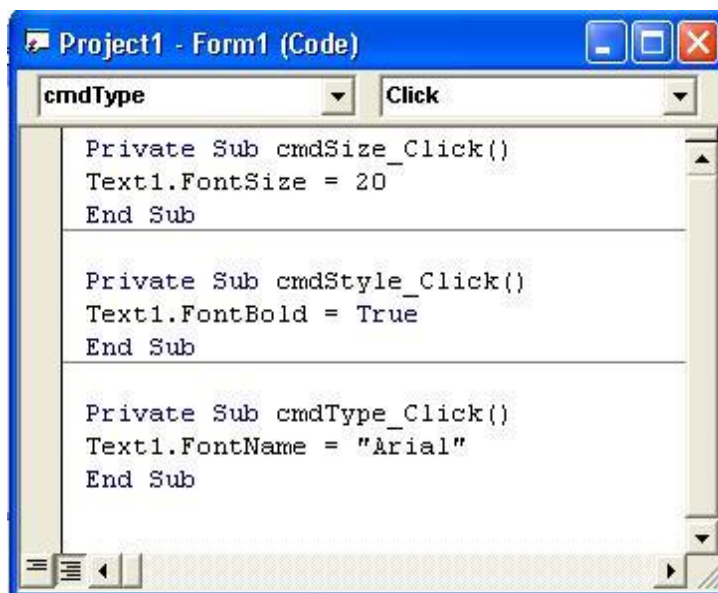
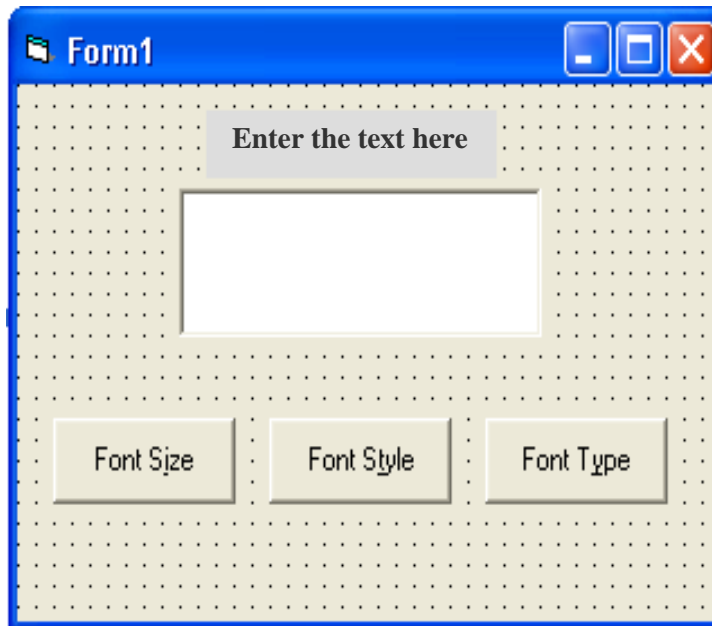
3-2-1: Font Editor

This project contains a text box allow to receive any text then by using three command boxes named "Size", "Style" and "Name" do the following:

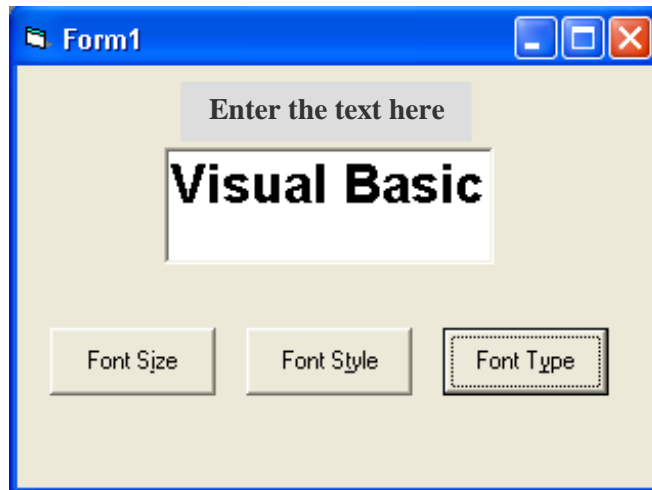
1. Change the font size to 20.
2. Change the font style to **Bold**.
3. Change the font name to "Arial".

Design Steps

- Set the control objects and write the codes for each commands, the project will look like this:



- Run the project and enter any text in the text box then click the three commands and see the result as shown below:

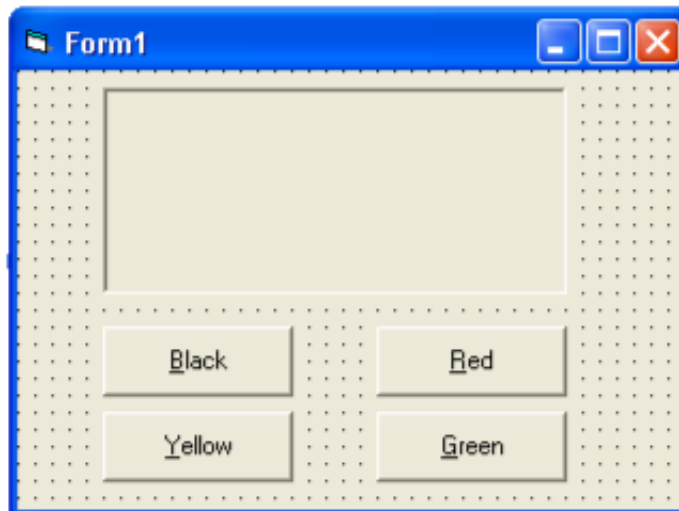


3-2-2: Dealing with literal constants

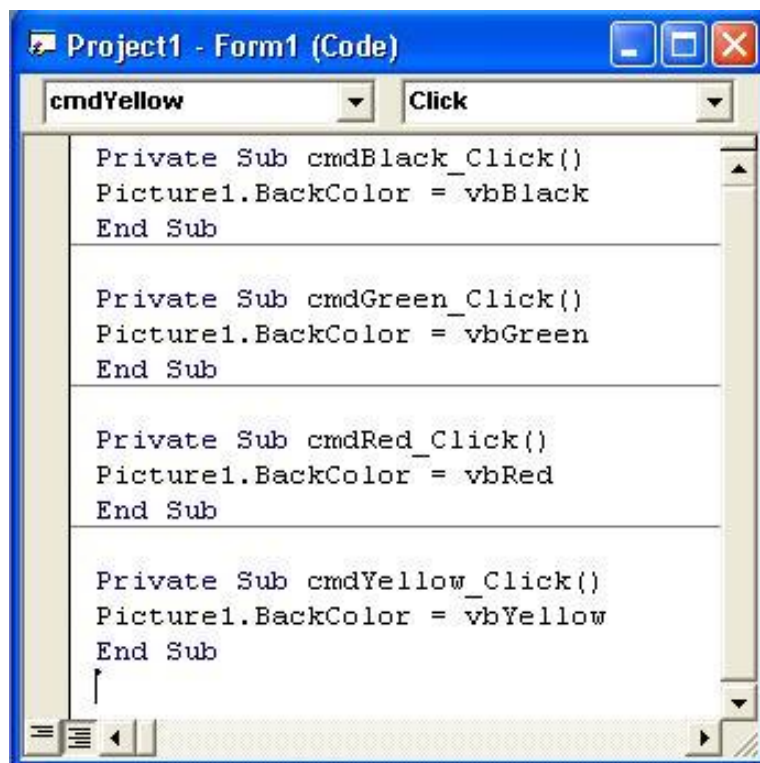
This project contains a picture box and four command buttons ("Black", "Red", "Yellow" and "Green"). The picture background will be changed according to the selected color represented by command buttons.

Design Steps

- Put the control objects in their places shown in figure below then change their properties



- Write the code for each control object after completing the code window will look like this:



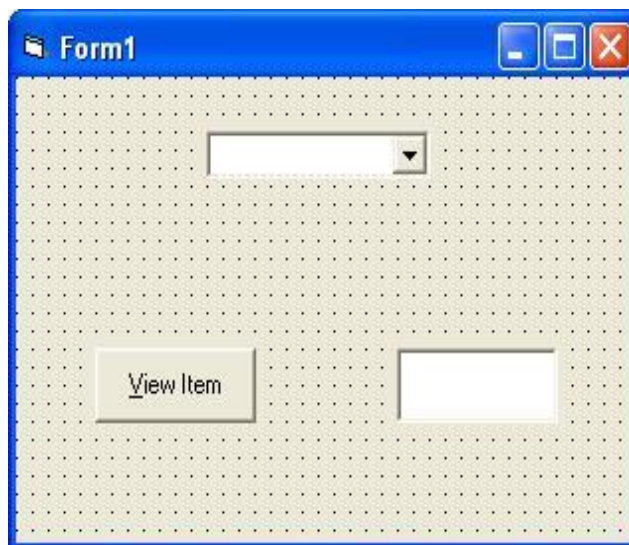
Note that we use the literal constants that represent the different colors. These constants (vblack, vbred, vbyellow, and vbgreen) are used in the Backcolor property to change the picture color.(See Appendix-A for more literal constants).

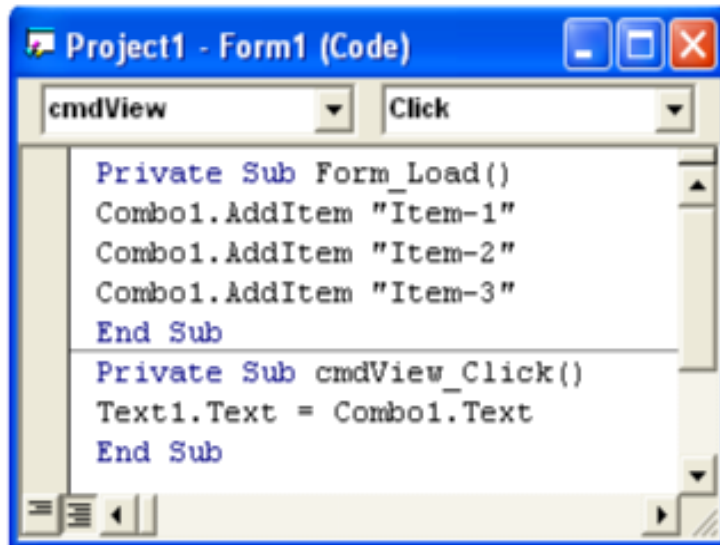
3-2-3: Using Combo Boxes

Design a project contains a combo box contains three items (Item1, Item2, Item3) added from the code. The project also contains a command "View item" and text box when we click this command the selected item will appear on the text box.

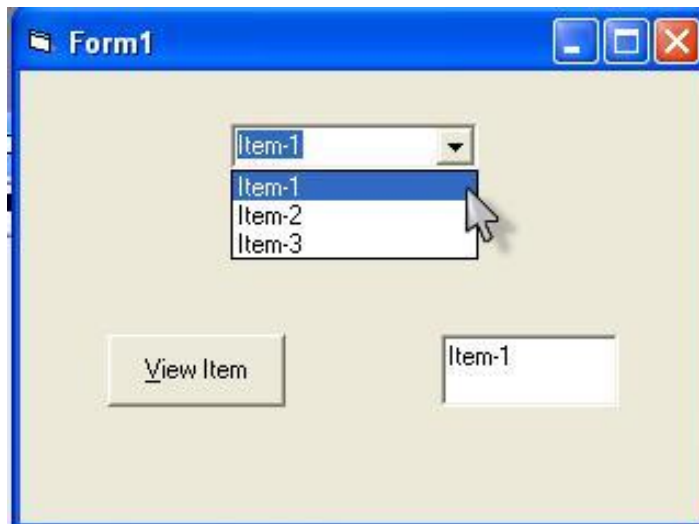
Design Steps

- The form will be as shown after putting the objects that we will use it in our project and writing the code to do our job.





- After running the program select any item from the combo box and show the selected item in the text box as shown in figure below

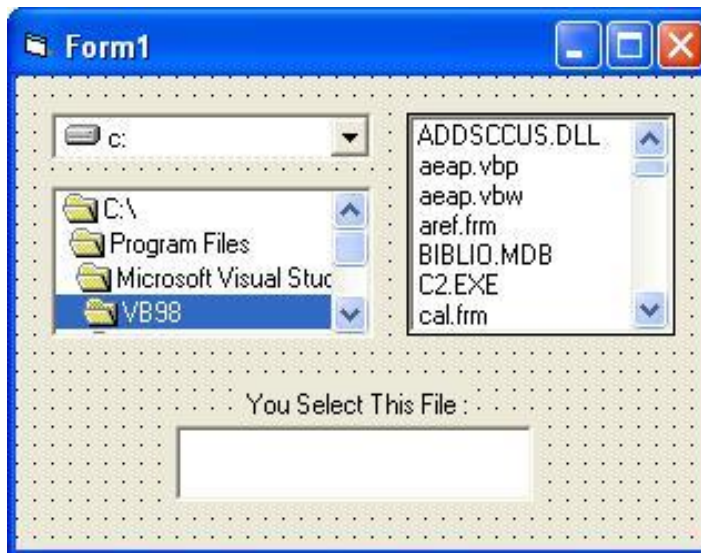


3-2-4: Working with Drive, Directory, and File List Boxes

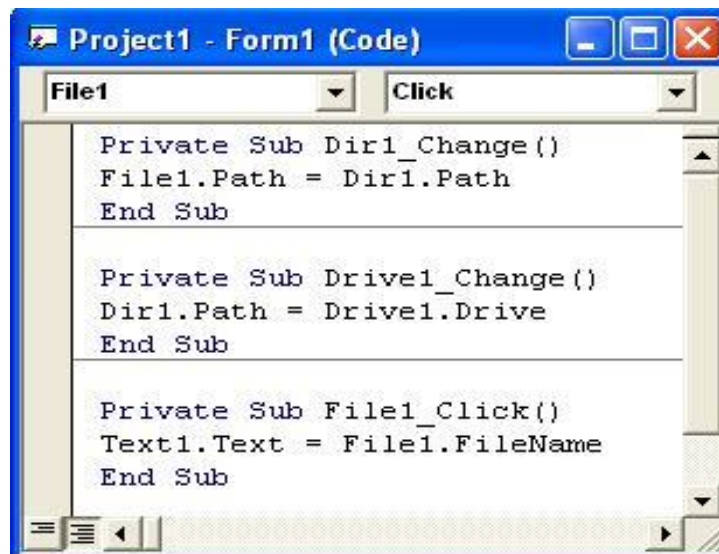
The following project contains Drive List box, Directory List Box, and File List Box. The selected file will appear on a text box when we select the drive, directory then the file from the selected objects.

Design Steps

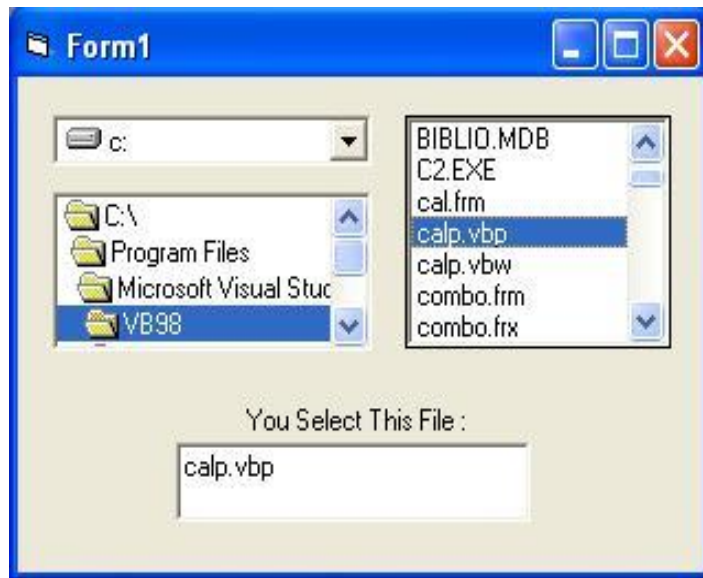
The following three figures shows the designing steps before and after executing the project and also shows the code window.



And the code will be :

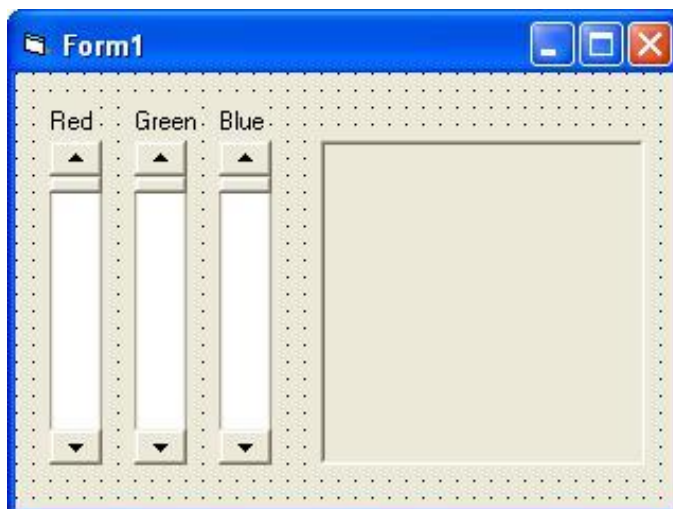


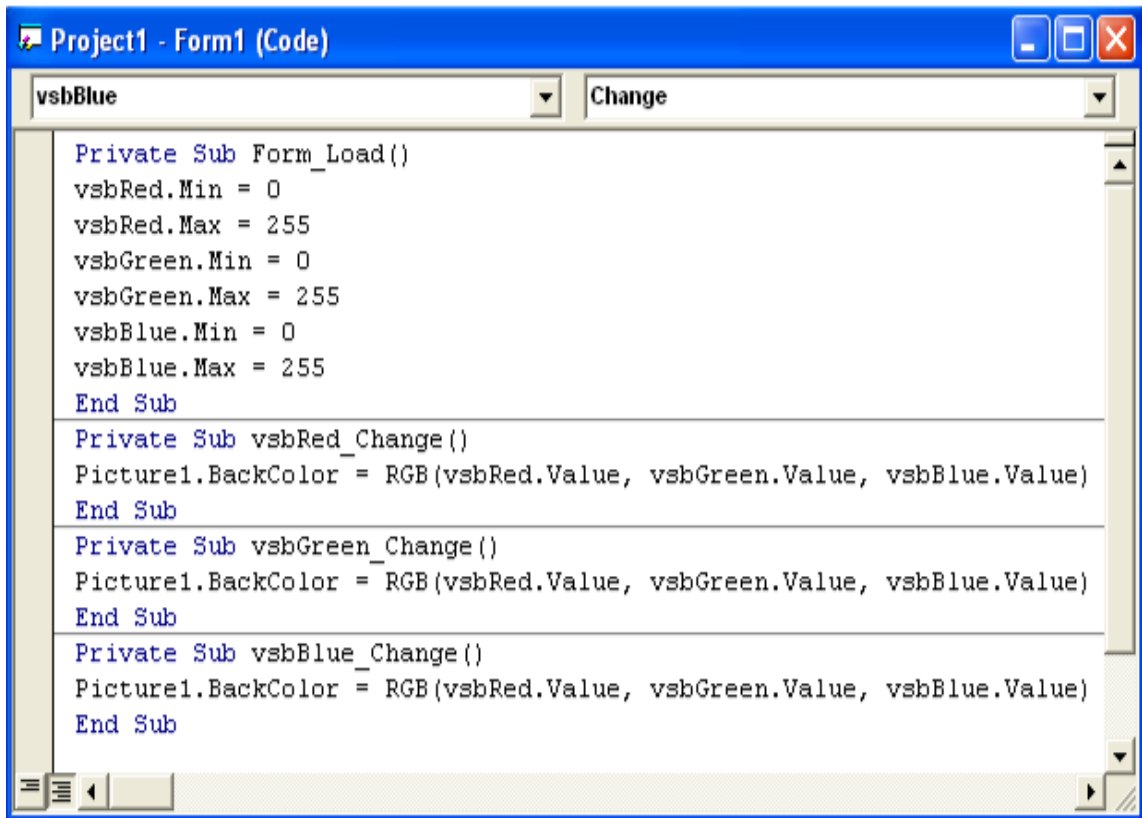
Run the project and show the result



3-2-5: Dealing with Vertical and Horizontal Scroll Bars

This project contains three Vertical Scroll Bars represents three colors (Red, Green, Blue) and a picture box. The three colors will be merged and appeared on the picture box background according to the selected colors. The scroll bars are scaled in the code from 0 to 255.



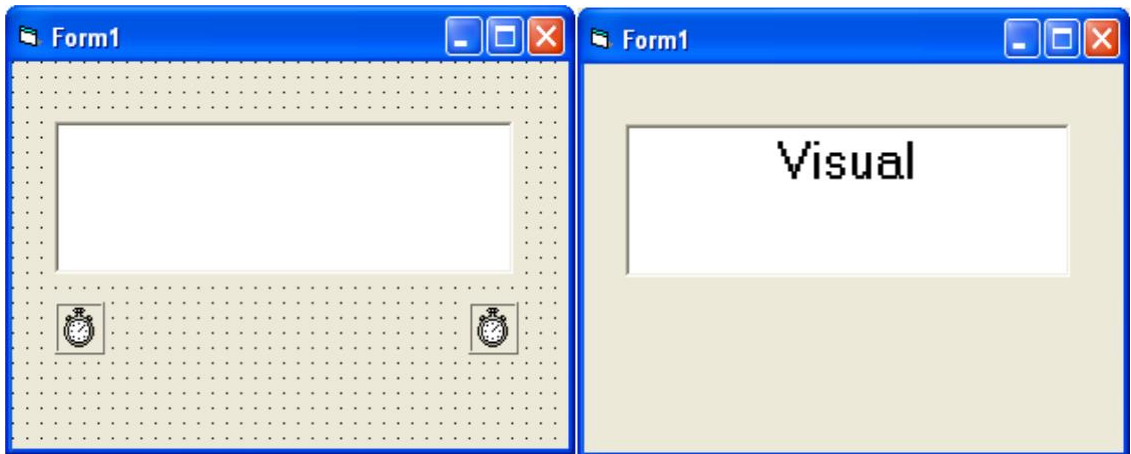


Hint: The function RGB (R, G, B) returns the resulted color from mixing the three standard colors (Red, Green, Blue) respectively. The value for each color alternated between (0 to 255). As an example RGB(255, 0, 0) returns the red color, RGB(0, 255, 0) returns the green color, and so on. In this question the values of R, G, and B are varied using the scroll bars.

3-2-6: Using the Timer object

The following project displays the string "Visual" on a text box after 2000 smilliseconds from the execution then displays the string "Basic" on the same text box after 4000 milliseconds. Change the timers interval and the

text properties (Alignment: center, font size:20, font bold: True) from the code.



```
Project1 - Form1 (Code)
Timer2
Timer

Private Sub Form_Load()
    Timer1.Interval = 2000
    Timer2.Interval = 4000
    Text1.Alignment = 2
    Text1.FontSize = 20
    Text1.FontBold = True
End Sub

Private Sub Timer1_Timer()
    Text1.Text = "Visual"
End Sub

Private Sub Timer2_Timer()
    Text1.Text = "Basic"
End Sub
```

3-3: Const- Statement

Constants are special case of variables. Constants are variables declared with storing a value on it and this value can not be changed during the program execution time. Constants are declared and assigned using the **Const** statement by the use of the following format:

Const Constant Name = Value

Where:

Constant Name: The name of the suggested constant that we want to store the value.

Value: The value that we want to assign as a constant value and it can be any of the following:

- A numerical value
- A literal value
- A literal constants
- A mathematical expression
- A logical value

As an example There are many constant values in many applications like pie (π) in mathematics and (ϵ_0) in physics and so on. So the definition of these constants in visual basic will be:

Const Pie = 3.14

Const Eps = 8.85E-12

If you add the following line to the program:

Pie = Pie + 1

An error message will appear tell you that the "assignment to constant not permitted".

3-4: With- statement

This statement is important when the user want to change several properties for the same control object. The general form for this statement is:

```

With Control object Name
    .property 1=value 1
    .property 2=value 2
    .property 3=value 3
    |
    |
    .property n=value n
End With
  
```

Where:

Control Object Name: The name of the selected control object that we want to change their properties.

Property 1 to Property n: The properties that we want to change for the same object.

Value 1 to Value n: The values that explained earlier that will be given to the properties 1 to n.

As an example if we have a Text Box named "txtName" and we want to change the following properties for the same text box: 1. text=Emad. 2. Font Bold. 3.Red font color 4.Font size = 16.

With txtName

.Text = "Electrical Engineering"

.FontBold = True

.ForeColor = vbRed

.FontSize = 16

End With

3-5: Rem- statement

This is an abbreviation for the remark statement that is used to explain a step (or more) by using a suitable comment without any execution. The remarked step will be colored by a green color to know that this is a comment line. A simplest way to remark a line is to use the single quotation symbol (') before the comment. The Rem statement (or (') symbol) is placed before the comment and the remarked line may be any where of the program. The format that used to express this statement is:

Rem Any comment **or** ' Any comment

As an example we will use the Rem statement in the first line of the program and in the same time we will use the single quotation (') in the end of the second line.

Rem This Is a Visual Basic Program

Text1.Text = 5 **'Enter a value from 1 to 10**

3-6: Dim- statement

Before you can use a variable, you must declare the variable by telling Visual Basic the name and data type that the variable is to hold. The **Dim** statement declares variables by assigning them a name and data type. Before you can use a variable, you must first declare that variable with a Dim statement. Here is the format of the **Dim** statement that you use to declare variables:

Dim VariableName As Data type **or** **Dim** VariableName Symbol

Where :

VariableName is the name you assign to the variable.

DataType and **Symbol** are one of the data types and symbols listed in table(3-1) below:

<i>Data Type</i>	<i>Symbol</i>	<i>Range</i>	<i>Storage</i>
Byte	N/A	0 to 255	1 byte
Integer	%	-32,768 to 32,767	2 bytes
Long	&	-2,147,483,648 to 2,147,483,647	4 bytes
Single	!	-3.402823E+38 to -1.401298E-45 (for negative values) 1.401298E-45 to 3.402823E+38 (for positive values)	4 bytes
Double	#	-1.797693134862E308 to -4.940656458412E-324 for negative values and from 4.94065645841247E-324 to 1.79769313486232E308 for positive values	8 bytes
Currency	@	-922,337,203,685,477.580 to 22,337,203,685,477.580	8 bytes
String	\$	1 to about 65,400 characters	1 byte/chr
Date	N/A	January 1, 100 to December 31, 9999	8 bytes
Boolean	N/A	True or False	2 bytes
Variant	N/A	Any value as large as Double	16 bytes

Table(3-1): Data Types and Symbols for various type of data

Note

If you don't declare any variable, Visual Basic assumes that an undeclared variable is of the Variant data type.

3-7: Public – statement

The statement is similar to Dim statement used to declare variables but the only difference that it is used to declare the general variables. So the Dim statement used to declare the local variables that we use it only in the same form procedure and the defined objects included on it. The Public statement more general from the Dim statement because it is used to define the variables used in the project and all the forms folded in it. The general form for this statement is:

Public VariableName As Data type or Public VariableName Symbol

Where VariableName, Data type, and Symbol are defined earlier in Dim statement

Note

The Dim statement can be written in the general procedure located at the top of the code window procedure without using Private Sub and End Sub. While the Public statement written in the Code Module window which can be viewed from the following path:

Project Menu → Add Module.

3-8: Option Explicit – statement

The statement **Option Explicit** tells Visual Basic that the rest of the code in this module is to declare all variables before they are used.

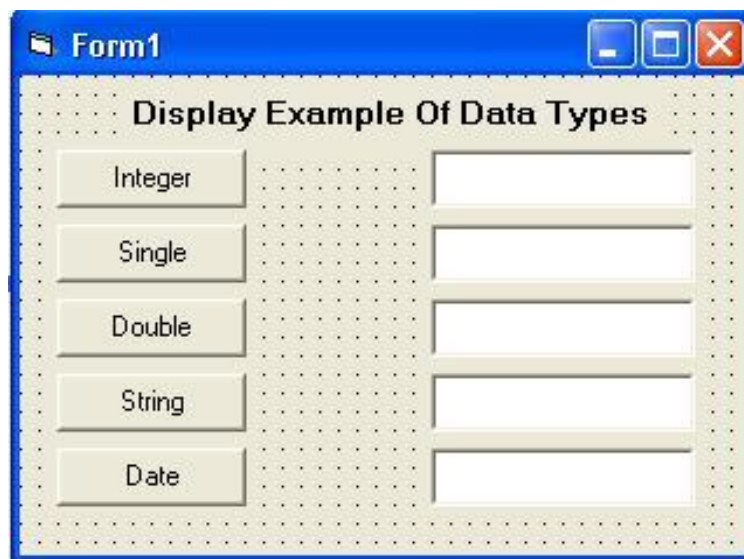
Thereafter, if you misspell a variable name in the middle of your program, Visual Basic will catch the error. The Option Explicit takes its place before the Dim or Public statement

Example

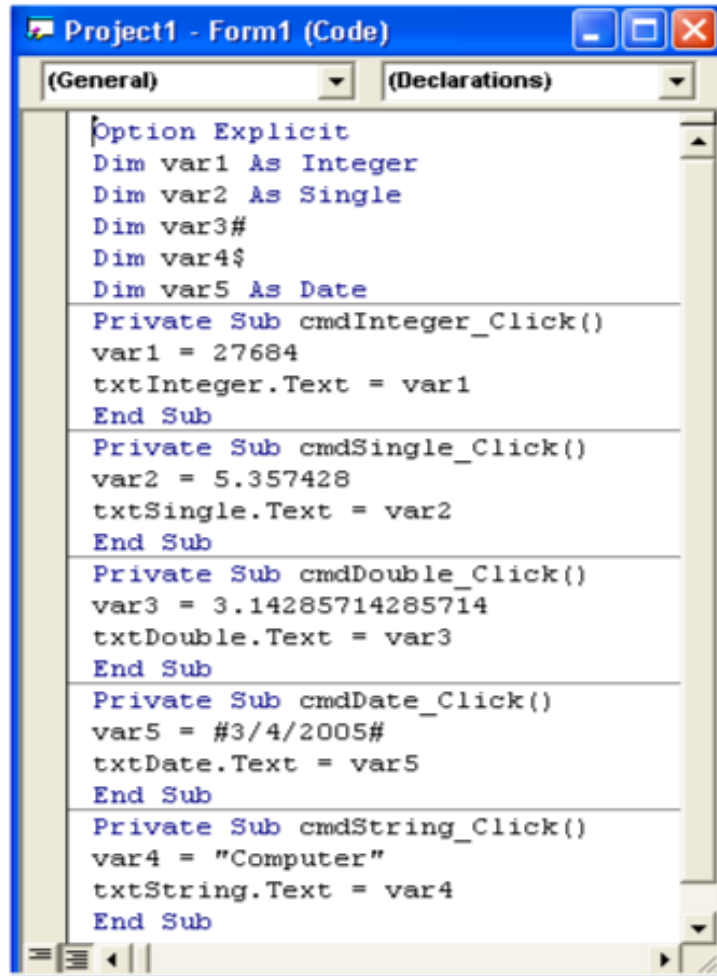
Design a project have a single form labeled "Display Example of Data Types" contains five commands ("Integer", "Single", "Double", "String" and "Date"). When you click any of them a specific data type will appear on a five text boxes.

Solution

After setting the objects and writing the codes for each command, the below figures will appear:



The programming code will be as follow:



```
Option Explicit
Dim var1 As Integer
Dim var2 As Single
Dim var3#
Dim var4$
Dim var5 As Date

Private Sub cmdInteger_Click()
var1 = 27684
txtInteger.Text = var1
End Sub

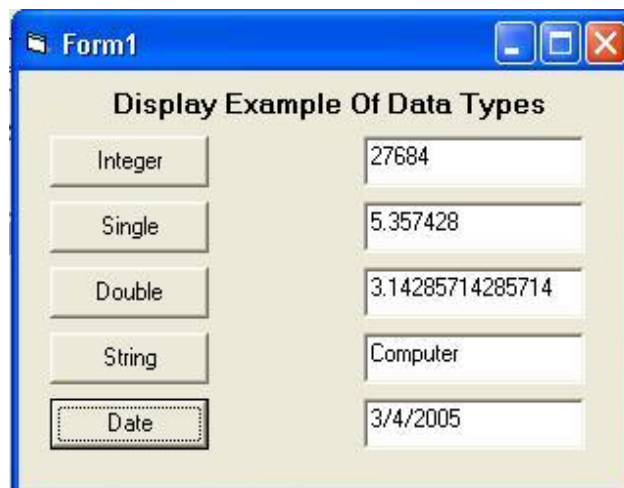
Private Sub cmdSingle_Click()
var2 = 5.357428
txtSingle.Text = var2
End Sub

Private Sub cmdDouble_Click()
var3 = 3.14285714285714
txtDouble.Text = var3
End Sub

Private Sub cmdDate_Click()
var5 = #3/4/2005#
txtDate.Text = var5
End Sub

Private Sub cmdString_Click()
var4 = "Computer"
txtString.Text = var4
End Sub
```

Run the program and see the results



Display Example Of Data Types	
Integer	27684
Single	5.357428
Double	3.14285714285714
String	Computer
Date	3/4/2005

Note-1:

The Date type variables must be enclosed by two Hash symbols (#)

Note-2:

If we set float number in a variable we declare it as Integer type, this number will be approximated to the nearest integer.

Note-3:

If we set Double type number in a variable we declare it as Single type, this number will be approximated to the nearest 6 floating point.

3-9: Print- statement

The **print** statement used in the program to print any value or any expression. Usually the print result will be appeared on the Form. This statement may take one of the following formats:

- 1) To print a specific values (strings, numbers, symbols, mathematical or logical expressions) use the following formula:

Print Value

- 2) To print a specific values separated by a single spaces use (;), as in the following formula:

Print Value-1 ; Value-2 ; ; Value-n

If we want to separate these values by 8 spaces use (,) instead of (;)

- 3) If we want to control the spaces number between the values we use the functions **Spc(n)** and **Tab(n)**. Where Spc(n) used when we want

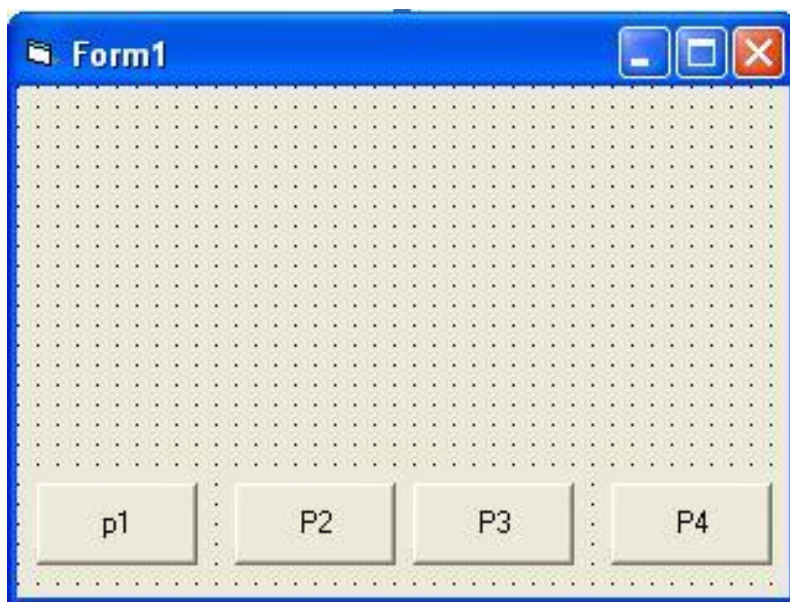
to separate values by n spaces starts from the end of the first value. Tab(n) is used to separate values by n spaces starts from the beginning of the form. Use the following formula:

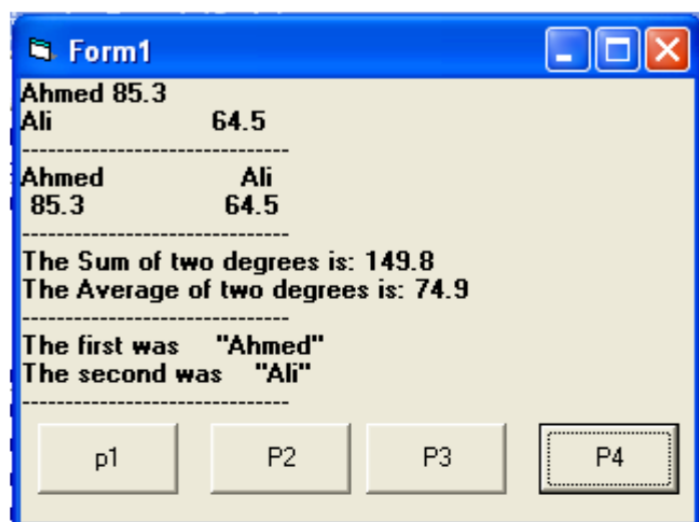
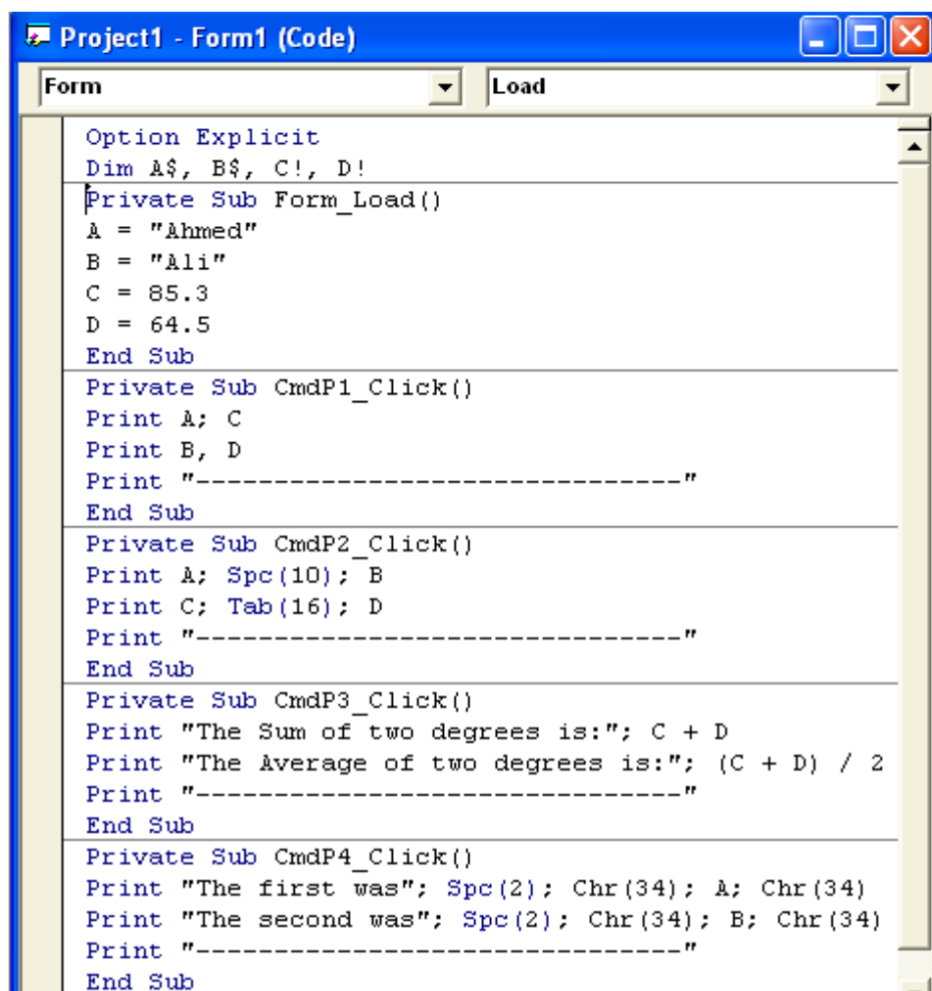
Print Value-1; (or ,) **Spc(n)** (or **Tab(n)**) ; (or ,) Value-2

Example

What would appear on the form after writing the below code if you know that this form contains four commands named "P1", "P2", "P3", and "P4".

Solution:





Note-1:

The event **Load** is used to store initial values in the used variables.

Note-2:

The function **Chr(m)** is used to obtain the character in the **ASCII table** that its number is (m). So Chr(34) gives the character that its value is 34 in ASCII table which is the double quotation ("). For more ASCII codes see **Appendix-B**.

3-10: Cls- statement

This statement is the inverse of the Print statement which is used to clear the form from any printing values. Cls statement may be written alone in the program or may be written as one of the following two formats:

Form1.Cls

Me.Cls

3-11: End- statement

As you see in the previous chapter, this statement is used to end the execution and exit from the application. Like **Cls** statement we can use it lonely without any control object.

3-12: Date- statement

This statement returns the computer date formats. You can use this statement can not be used lonely (like Cls statement) but as a value for other control objects.

3-13: Time- statement

This statement returns the computer time formats. As Date statements You can not use this statement lonely but as a value for other control objects.

3-14: Now- statement

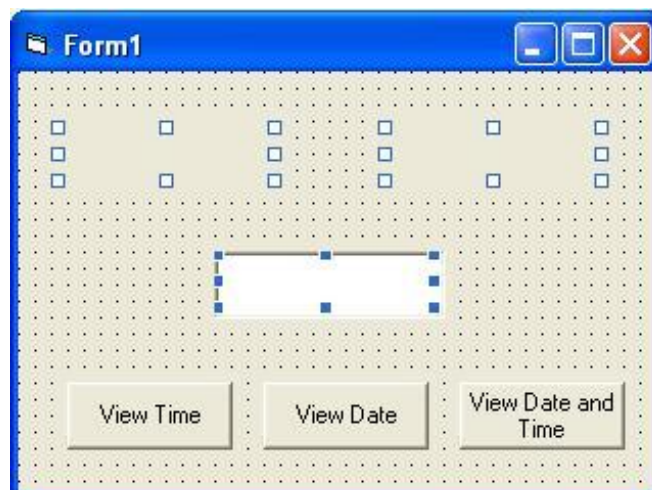
This statement return the computer time and date formats and it can not be used lonely but as a value for other control objects.

Example:

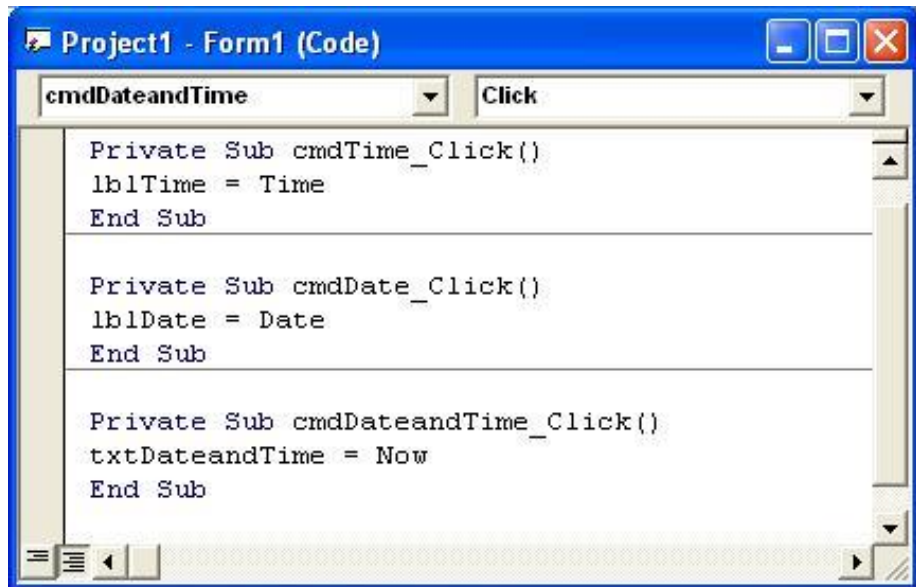
Write a VB program that displays the time on a Label Box when we click over the command button "View Time" and the date on another Label Box when we click over the command button "View Date". Add a text Box to the form to view date & time when we click over the command "View Date and Time".

Solution

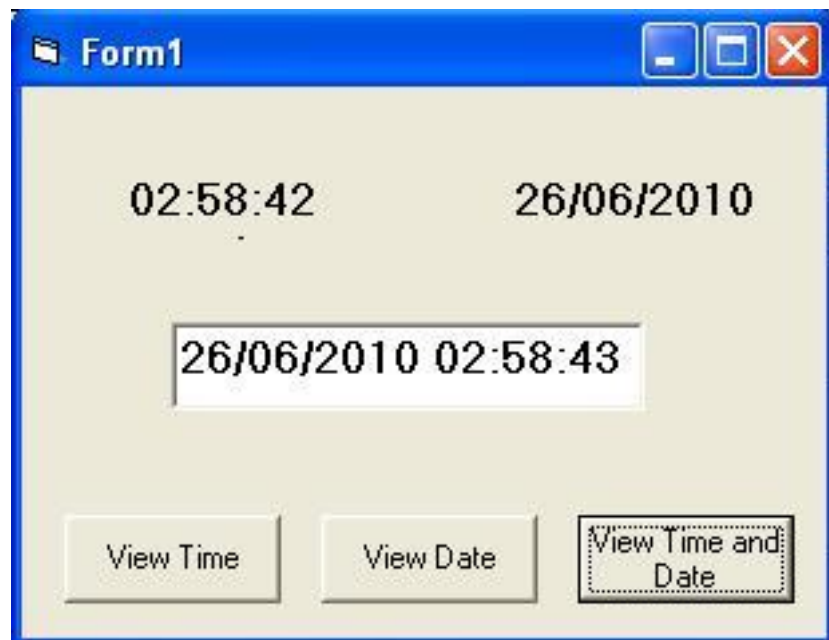
we need the following control objects shown below:



Double click over the commands sequentially to view the below code window that we will write our program:



Run the project then see the result. The result must be as shown below



3-15: Problems

1. Design a project contains a single form and four commands represent four colors. When we click any of these commands the background color for the form will be changed.
2. Design a project to enter a string from a text box then change the font type, font size, and font style (Italic, Bold) for this string when you click the three commands "Ftype", "Fsize", "Fstyle".
3. Design a project contains a label box having the below properties. When we click the form the label on the label box will enlarge to double size.

Name = lblLarger Alignment = 2-Center Caption = Larger AutoSize = True Font = Bold (10) Top = 360 Left = 1935

9. Write a VB program to change the below properties when we click the three commands "modify text" , "modify picture" and "hide picture" respectively:

- 1- Make the text box closer to the beginning of the form and display "Visual Basic" on it.
- 2- Change the color of the picture to red and hide the picture.
- 3- Hide the picture to be invisible.

- 3.** By using visual basic programming language implement a font editor contains five text boxes to enter the font type, size, and style (Bold and Italic) respectively. The fifth text box is used to enter the modified text. The operation will be executed according to double clicked a command box.
- 4.** Design a visual basic project that changes the form color to three basic colors (Red, Green, Blue) successively after 2, 4, and 6 seconds respectively.
- 5.** Design a VB project contains a label box, picture box and two command buttons named "Maximize" and "Minimize" when we click them the label box will enlarge to triple its size and the picture box will reduced to half its sizes.
- 6.** Design a project contains VSB, HSB and an Image Box. The two scroll bars ranges from 0 to 100. The Image box enlarged horizontally or vertically according to the two scroll bars.
- 7.** Explain the advantage of the following visual basic statements:

frmcolor. Backcolor = vbGreen

picflower. Visible = False

optselect. Value = True

imgbliss . Stretch = True

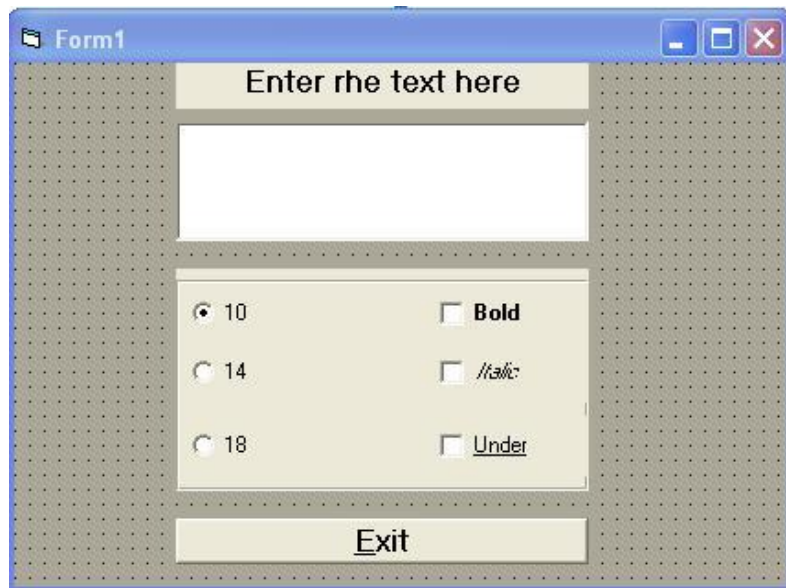
- 8.** By using visual basic programming language implement a digital clock contains two commands (Date and Time) and two text boxes. When we click these commands the daily date and time will appear on these text boxes.

4. Design a VB project contains a three Vertical Scroll Bars represents three colors (Red, Green, Blue) and a picture box. The three colors will be merged and appeared on the picture box background according to the selected colors. The scroll bars are scaled in the code from 0 to 255.
5. Design a project contains an image box, has the below properties, and 8 commands. These commands are represents eight operations on this image like; hiding, viewing, moving it in the four directions, maximizing and minimizing it. Use the movement steps 100 Twip and the maximizing and minimizing step 150 Twip.

Name = Image1
Stretch = True
Picture = **C:\My Documents\My Pictures\Bliss.Bmp**
Left = 0
Top = 0
Width = 1575
Height = 1335

13. Design a project have a single form labeled "Display Example of Data Types" contains five commands ("Integer", "Single", "Double", "String" and "Date"). When you click any of them a specific data type will appear on a five text boxes. Use a statement to view an error message if no variable was declared.

- 5.** Design a project contain the below form then program it to be a font editor project. After running the program enter any text then do operations viewed in the form.



- 15.** Design a project contain label box only then write a code that make this label box act as a digital clock.

Hint: Use Timer object and set its interval to 1000.

- 16.** What is data types are defined by suffix shown below:

Index%

Counter&

TaxRate!

Ratio#

CustomerName\$

- 17.** What is the purpose of the Const statement? How does it differ from a Dim statement? How is a Const statement written?
- 18.** Suppose a Print statement includes five output items, separated by commas. How can the statement be rewritten so that the output items appear on the same line, with minimum spacing between them?
- 19.** Suppose a Print statement includes five output items. How can the statement be rewritten so that the first three data items appear on one line and the remaining two data items appear on a second line?
- 20.** In Visual Basic, how does a named constant differ from a variable?
- 21.** Write a single (one-line) declaration for each of the following situations:
- 6.** Declare x1 and x2 as single-precision real variables.
 - 7.** Declare **CustomerName** and **Address** as string variables.
 - 8.** Declare **Counter** as an integer variable, and Sum and Variance as double-precision real variables.

 Declare City as a named string constant whose value is "New York".
- 22.** Write a VB program that displays the time on a Label Box when we click over the command button "View Time" and the date on another Label Box when we click over the command button "View Date". Add a text Box to the form to view date & time when we click over the command "View Date and Time".

CHAPTER FOUR

Operators and Functions

Introduction

Visual Basic supports numerous operators and functions. In this chapter we will list the most common operators and functions. We use these operators and functions in expressions when calculating and working with various types of data. Operators and functions manipulate data by combining or computing results.

4-1: Visual Basic Operators

Data values and controls are not the only kinds of assignments that you can make. With the Visual Basic operators, you can calculate and assign expression results to variables when you code assignment statements that contain expressions. So an operator is a word or symbol that does math and data manipulation, Operators are classified into four mainly types these are:

- 1. Mathematical Operators.**
- 2. String Operators.**
- 3. Conditional Operators.**
- 4. Logical Operators.**

4-1-1: Mathematical Operators

These types of Operators manipulate data by combining or computing results mathematically. Most mathematical operators are symbols, but some, such as Mod, look more like Visual Basic commands. Table (4-1) below illustrates these operators according to their predefined order

<i>Operator</i>	<i>Meaning</i>	<i>Example</i>	<i>Result</i>
) (Parentheses	$(2+3) * 7$	35
\wedge	Exponentiation	$2 \wedge 3$	8
*	Multiplication	$2 * 3$	6
/	Division	$6 / 2$	3
\	Integer Division	$11 \backslash 3$	3
Mod	Modulus	$11 \text{ Mod } 3$	2
+	Addition	$2 + 3$	5
-	Subtraction	$6 - 3$	3

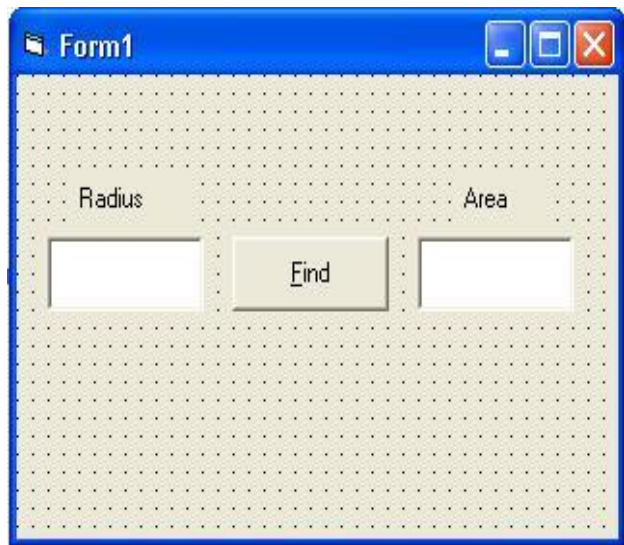
Tabel (4-1): VB Operators

Example-1:

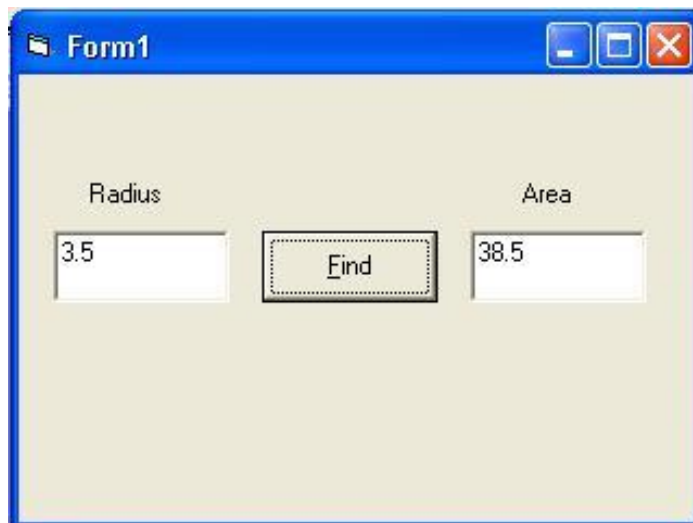
Design a project contains a command button "Find" and two text boxes. When we enter a number (represents the radius of a circle) from the first text box, the area of this circle will appeared on the second text box and when we click the command "Find".

Solution:

```
Private Sub cmdFind_Click()  
Dim R!, A!  
Const pie = 3.14  
R = Val  
(txtRadius.Text) A = R  
^ 2 * pie txtArea.Text =  
A End Sub
```



After running the project enter any value represents the radius for the circle then click the command "Find", the area of this circle will appeared on the other text box.



Note :

The text box always deals with texts (i.e data of type literal value) but if we want this text to deal with numbers we use the function **Val()** which converts the literal values to number of type variant.

4-1-2: String Operators

These operators are used in string operations and usually classified into two operations as shown in table (4-2) below:

<i>Operations</i>	<i>Operators</i>	<i>Example</i>	<i>Result</i>
Concatenation	+	"com"+"puter"	computer
	&	"com"&"puter"	computer
Comparing	Like	"abc"Like"adc"	False

Table (4-2): VB String Operators

Example-2:

What is the result of the following program?

```
Private Sub Form_Click)(
Dim A, B, C, D As String
Dim E, F, G As Boolean
A = "I will install"
B = "Visual Basic version 6.0"
C=A+B
D = A & B & "in my computer"
E = B Like "Visual Basic version 7.0"
F = B Like "Visual ?asic version 6.0"
```

```
G = B Like "Visual Basic version #.0"  
Print C; Print D  
Print E; F; G  
End Sub
```

Answer:

```
I will install Visual Basic version 6.0"  
I will install Visual Basic version 6.0 in my computer  
False True True
```

Note-1 : The question mark (?) is used to indicates to any character may take this place.

Note-2 : The hash symbol (#) is used to indicates to any number may take this place.

4-1-3: Conditional Operators

Conditional operators compare data. By comparing data and analyzing results, your Visual Basic program can decide an appropriate course of action based on data alone. By writing programs with conditional operators and statements, you let Visual Basic decide, at runtime, which statements to execute in a program. Through the conditional operators, you can learn if a value is less than, equal to, or greater than another value. Table (4-3) supports six conditional operators.

<i>Operator</i>	<i>Description</i>	<i>Example</i>	<i>Result</i>
=	Equal to	7 = 2	False
<	Greater than	6 > 3	True
>	Less than	5 < 11	True
<=	Greater than or equal to	23 >= 23	True
>=	Less than or equal to	4 <= 21	True
<>	Not equal to	3 <> 3	False

Table(4-3): VB Conditional Operators

Example-3:

What is the result of the following program?

```

Private Sub Form_Click()
Dim A, B As Integer
Dim A1, B1, C1, D1, E1, F1 As Boolean
A=54
B=80
A1=(A=B)
B1=(A>B)
C1=(A<B)
D1=(A<=B)
E1=(A>=B)
F1=(A<>B)
Print A1; Spc(2); B1; Spc(2); C1
Print D1; Spc(2); E1; Spc(2); F1
End Sub

```

Answer:

False False True

True False True

4-1-4: Logical Operators

The logical operators let you combine two or more sets of conditional comparisons. Like the Mod operator, the logical operators use keywords instead of symbols. Visual Basic supports three logical operators listed in the table (4-4) below:

<i>Operator</i>	<i>Description</i>	<i>Example</i>	<i>Result</i>
Not	Negates truth	Not (3 = 3)	False
And	Both sides must be true	(2 < 3) And (4 < 5)	True
Or	One side or other must be true	(2 < 3) Or (6 < 7)	True
Xor	One side or other must be true but <i>not both</i>	(2 < 3) Xor (7 > 4)	False

Table(4-4): VB Logical Operators**Example-4:**

What is the result of the following program?

```
Private Sub Form_Click()
Dim A, B, C, D As Boolean
Dim Res1, Res2, Res3, Res4 As Boolean
A = True
B = True
C = False
D = False
Res1 = Not D
Res2 = A Or B
```

```
Res3 = A And D
Res4 = B Xor C
Print "Res1="; Res1
Print "Res2="; Res2
Print "Res3="; Res3
Print "Res4="; Res4
End Sub
```

Answer

```
Res1= True
Res2= True
Res3= False
Res4= True
```

Note: Technically, the six conditional operators offer enough power to test for any condition, but you can greatly enhance their flexibility by combining the conditional operators with logical operators.

Note: The complete priority of VB operators is shown in table (4-5) below:

<i>Order</i>	<i>Operators</i>
1	Mathematical Operators
2	Conditional operators
3	String Operators
4	Logical operator

Table(4-5): VB Operators priority

Example-5

What is the result of the following program?

```
Private Sub Form_Click()
Dim A, B, C, D As Integer
Dim V1, V2, V3, V4, V5, V6 As Boolean
A=50:B=70:C=80:D=90
V1 = A > B And B < C
V2 = A <> B And B <> C
V3 = A > C Or D > A
V4 = V1 And V2 Or V3 Or A < C
V5 = A + B > A / B And D - C / 2 > B \ A
V6 = Not A + D ^ 2 = B / C ^ 2 Xor D ^ 2 = C \ A
Print V1; Spc(2); V2; Spc(2); V3
Print V4; Spc(2); V5; Spc(2); V6
End Sub
```

Answer

If we applied the priority shown in previous table table(4-5), the complete solution for (V1, V2, V3, V4, V5, and V6) will be:

```
V1 = (A > B) And (B < C)
V2 = (A <> B) And (B <> C)
V3 = (A > C) Or (D > A)
V4 = V1 And (V2 Or V3) Or (A < C)
V5 = ((A + B) > A / B) And (((D - C) / 2) > (B \ A))
V6 = Not ((A + D) ^ 2 = (B / C) ^ 2) Xor (D ^ 2 = (C \ A))
```

So the result will be:

False True True

True True True

4-2: Visual Basic Functions

Functions accept one or more arguments and do work with those arguments. The function then returns a single value. There are many built-in functions in Visual Basic that your programs can call. Programs that call functions must do something with the return values from those functions. A function might require one argument or more. If you send an argument list to a function, that function operates on those arguments and returns a single value based on the argument list. In general, functions can be classified into many types:

- 1. Input box function.**
- 2. Message box function.**
- 3. Mathematical functions.**
- 4. Conversion functions.**
- 5. String functions.**
- 6. Date and Time functions.**
- 7. User functions.**

4-2-1: Input Box Function

Input boxes are great to use when the user must respond to certain kinds of questions. Text boxes controls are fine for getting fixed input from the user, such as data values with which the program will compute. Input boxes are great for asking the user questions that arise only under certain conditions. Input boxes always give the user a place to respond with an answer. There are two **InputBox()** functions. Here are the formats of the **InputBox()** functions:

```
var = InputBox ("Message","Title")  
  
or:  
  
var = val (InputBox ("Message", "Title"))
```

Where:

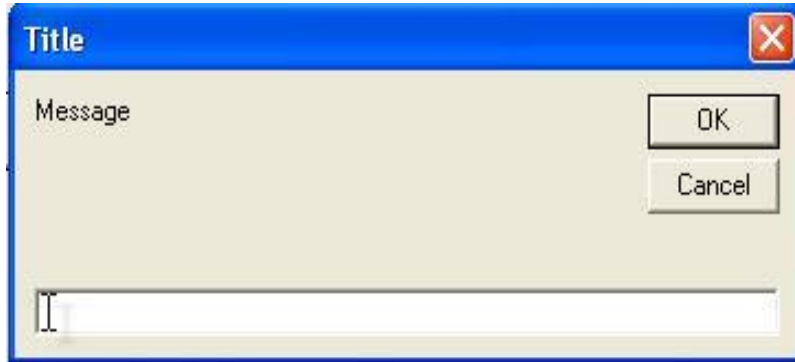
Var: Variable name returned value.

Message: Is a string (either variable or a string constant enclosed in quotation marks) and forms the text of the message displayed in the input box.

Title: Is an optional string that represents the text in the input box's title bar. If you omit the title, Visual Basic uses the project's name for the input box's title bar text.

The difference between the **InputBox()** functions lies in the return value. The first **InputBox()** function returns a string data type and the

second **InputBox\$()** function returns a variant data type. After applying the **InputBox()** forms above the following figure will appear:



Example-6

Write a VB program to enter four numbers then calculate the square and cube value for these numbers. Print these numbers with their square and cube values as a table.

Solution:

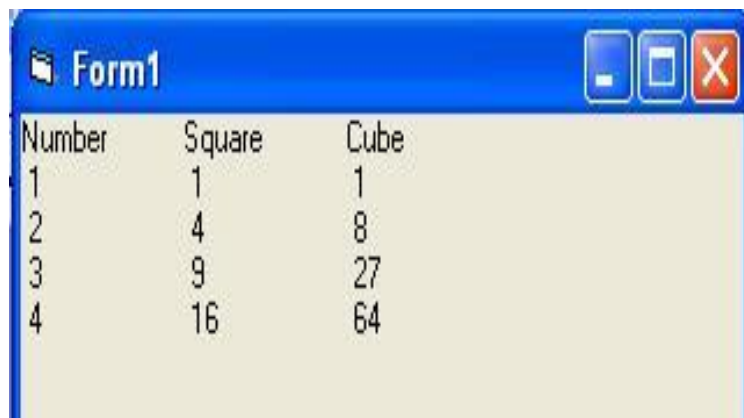
```
Private Sub Form_Click()  
Dim A!, B!, C!, D!  
Dim X1!, X2!, X3!, X4!  
Dim Y1!, Y2!, Y3!, Y4!
```

```
A = Val(InputBox("Enter the first number", "Number 1"))  
B = Val(InputBox("Enter the second number", "Number 2"))  
C = Val(InputBox("Enter the third number", "Number 3"))  
D = Val(InputBox("Enter the fourth number", "Number 4"))  
X1=A^2:X2=B^2:X3=C^2:X4=D^2  
Y1=A^3:Y2=B^3:Y3=C^3:Y4=D^3
```

```
Print "Number", "Square", "Cube"  
Print A, X1, Y1  
Print B, X2, Y2  
Print C, X3, Y3  
Print D, X4, Y4
```

```
End Sub
```

Finally: Run the program and click the form then enter the four numbers in the viewed input boxes respectively. The following results will appear (if the entered values was 1, 2, 3, and 4).



The screenshot shows a Windows form titled 'Form1' with a standard Windows XP-style title bar. The form contains a table with three columns: 'Number', 'Square', and 'Cube'. The table has four rows of data, corresponding to the numbers 1 through 4. The values in the 'Square' column are 1, 4, 9, and 16, and the values in the 'Cube' column are 1, 8, 27, and 64.

Number	Square	Cube
1	1	1
2	4	8
3	9	27
4	16	64

4-2-2: Message Box Function

There will be many times in programs when you'll need to ask the user questions or display error messages and advice to the user. Often, the controls on the form won't work well for such dialogs boxes aren't controls. Unlike controls that stay on the form throughout a program's entire execution cycle, a message box pops up on top of the form and

disappears when the user responds to the message box, usually by clicking the message box's OK command button.

There are two ways to produce message boxes. You can use the **MsgBox** statement or the **MsgBox()** function. The **MsgBox** statement displays messages for the user. In addition, the **MsgBox()** function displays messages but also provides a way for your program to display and check for multiple command button clicks on the message box window. Therefore, the statement following the statement executes when the user clicks OK. Here is the format of the **MsgBox** statement:

MsgBox "Message" , Symbol , "Title"

Where:

Message: is a string (either variable or a string constant enclosed in quotation marks) and forms the text of the message displayed in the message box.

Title: is an optional string that represents the text in the message box's title bar. If you omit the title, Visual Basic uses the project's name for the message box's title bar text

Symbol: is an optional numeric value, or VB constants, that describes the options you want in the message box. These options that you select determine whether the message box displays an icon, selection buttons as well as the button focusing. Tables (4-6), (4-7), and (4-8) contain the code values and VB constants that we will use it to form these three options respectively.

<i>Icons</i>	<i>Value</i>	<i>VB Constant Value</i>	<i>Description</i>
	16	VbCritical	Displays the halt sign icon
	32	VbQuestion	Displays the question mark icon
	48	VbExclamation	Displays the exclamation icon
	64	VbInformation	Displays the information icon

Table (4-6): Icon symbols and VB constants for the **MsgBox** function

<i>Value</i>	<i>VB Constant</i>	<i>Selection button</i>
0	VbOKOnly	OK
1	VbOKCancel	OK and Cancel
2	VbAbortRetryIgnor	Abort, Retry and Ignore
3	VbYesNoCancel	Yes, No and Cancel
4	VbYesNo	Yes and No
5	VbRetryCancel	Retry and Cancel

Table (4-7): Selection button VB constants for the **MsgBox** function

<i>Value</i>	<i>VB Constant</i>	<i>Focus</i>
0	VbDefaultButton1	Button 1 default
256	VbDefaultButton2	Button 2 default
512	VbDefaultButton3	Button 3 default
768	VbDefaultButton4	Button 4 default

Table (4-8): Button focusing VB constants for the **MsgBox** function

Note-1

We can use either the values or the VB constants to build any message box.

Note-2

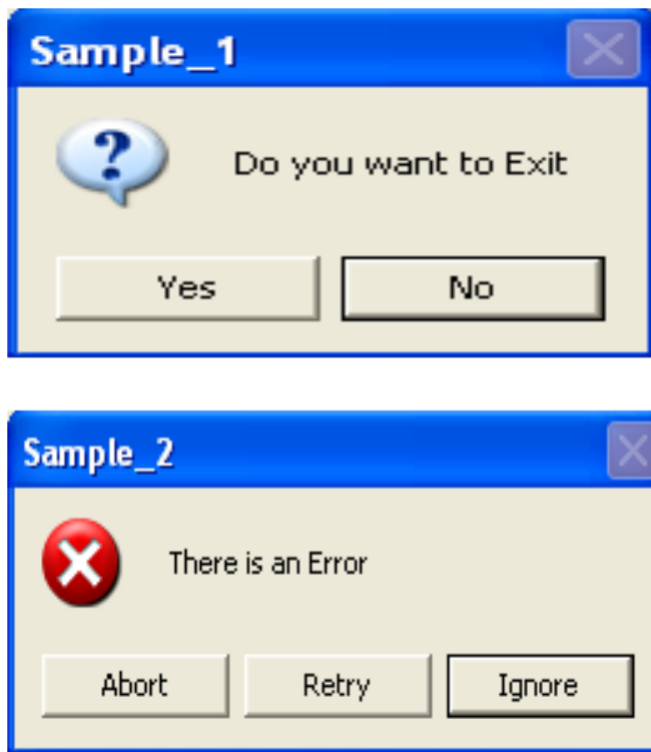
In **InputBox()** and **MsgBox()** if we want to write the "message" in two or more lines follow the following form:

"String1" + **Chr** (13) + "String2" +

Where the function **Chr()** is explained in the following few pages.

Example-7

What are the optional values that we set in the **MsgBox** to view the below message boxes?



Answer:

Symbol = vbQuestion + vbYesNo + vbDefaultButton2

MsgBox "Do you want to Exit", **Symbol**, "Sample_1"

Symbol = vbCritical + vbAbortRetryIgnore + vbDefaultButton3

MsgBox "There is an Error", **Symbol**, "Sample_2"

Example-8

What would appear when we execute the following codes?

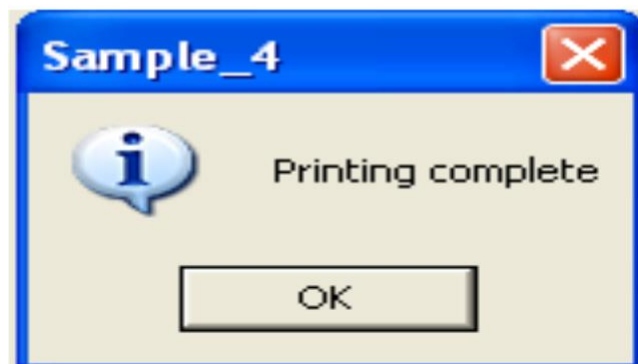
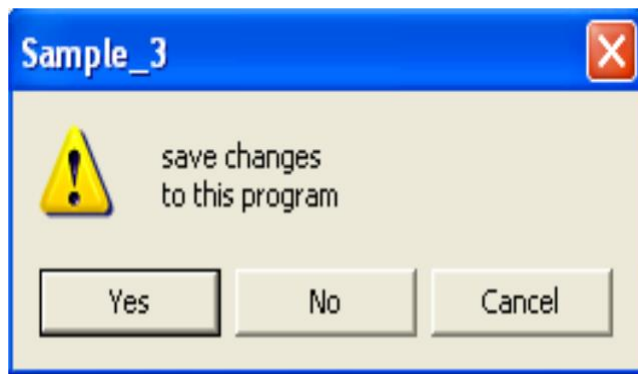
```
Symbol = vbExclamation + vbYesNoCancel
```

```
MsgBox "save changes"+Chr(13)+"to this program", Symbol, "Sample_3"
```

```
Symbol = vbInformation
```

```
MsgBox "Printing complete", Symbol, "Sample_4"
```

Answer:



MsgBox() function: The format of the **MsgBox()** function is almost identical to that of the **MsgBox** statement. Always assign a **MsgBox()** function to a variable represent the return value. Here is the format of the **MsgBox()** function:

Return value = MsgBox ("Message", Symbol, "Title")

Table (4-9) lists the possible return values for the **MsgBox()** function. In other words, the return variable may contain one of these values (Return value or Return VB Constant). A subsequent **If** statement (that we explain it in details in the next chapter) can then test to see which command button the user pressed.

Pressed button	Return value	Return VB constant
OK	1	VbOK
Cancel	2	VbCancel
Abort	3	VbAbort
Retry	4	VbRetry
Ignore	5	VbIgnore
Yes	6	VbYes
No	7	VbNo

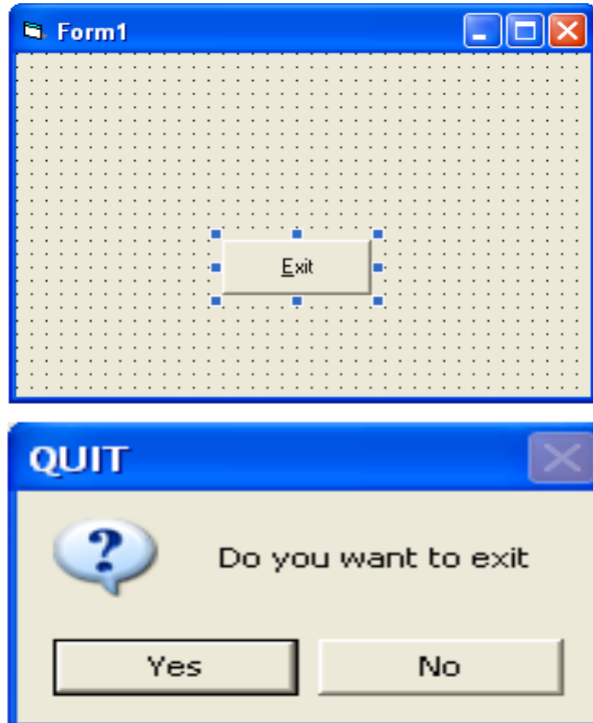
Table (4-9): The return Value for the pressed button

Example-9

Design a project contains a single command named "Exit" when we click it a message box will appeared. This message box contains two selection buttons Yes and No asked you to assert your existence and when you click yes the application will be stopped.

Solution:

```
Private Sub cmdExit_Click()  
Sym = vbQuestion + vbYesNo  
Ret = MsgBox("Do you want to exit", Sym, "QUIT")  
If Ret = vbYes Then End  
End Sub
```



4-2-3: Mathematical Functions

The mathematical functions will help you write programs when you need to include mathematical calculations in the code. Most of life applications used the mathematical functions to solve many complex problems contain various equations. In following few lines we will lists some of more used mathematical functions in Visual Basic program that may see it as a same spell in other programming languages.

1. Abs (n)

Returns the absolute value of the value (n)

Ex: Find $Y = |X|$ if you know that $X = -3.54$

$X = -3.54$

$Y = \text{Abs}(X)$

Print Y

Ans: $Y = 3.54$

2. Sqr (n)

Returns the square root of the value (n)

Ex: Find $Y = \sqrt{X}$ if you know that $X = 16$

$X = 16$

$Y = \text{Sqr}(X)$

Print Y

Ans: $Y = 4$

3. Exp (n)

Returns the base of the natural logarithm of the value (n) (Exponential value of (n)).

Ex: Find $Y = e^X$ if you know that $X = 1.4$

$X = 1.4$

$Y = \text{Exp}(X)$

Print Y

Ans: $Y = 8698888888818888$

4. Log (n)

Returns the natural logarithm of the value (n)

Ex: Find $Y = \text{Ln}(X)$ if you know that $X = 2.4$

$X = 2.3$

$Y = \text{Log}(X)$

Print Y

Ans: $Y = 0.8754687373539$

5. Sgn (n)

Return (1) if the sign of value (n) was positive and (-1) if it was negative

Ex: What is the result of the following VB program

$X = -3.5$

$Y = \text{Sgn}(-3.5)$

Print Y

Ans: $Y = -1$

6. Rnd

Returns a random number ranges from 0 to 1

Ex: Generate and print a random number in the interval [0, 20]

Ans:

```
Y= 20 * Rnd
```

```
Print Y
```

7. Randomize

Initialize the random number generator

Ex: Generate and print random number in the interval [0, 1] without repetition.

Ans:

```
Randomize
```

```
Y = Rnd
```

```
Print Y
```

8. Round (n)

Return an integer value represents the approximation of the value (n).

Ex: Approximate X to nearest integer if you know that X=12.5

Ans:

```
X = 12.5
```

```
Y = Round(X)
```

```
Print Y
```

9. Sin (n)

Returns the computed sine of the value (n) expressed in radians

Ex: Find $Y = \sin(X)$ if you know that $X = 2.3$ rad

$X = 2.3$

$Y = \sin(X)$

Print Y

Ans: $Y = 0.74570521217672$

10. Cos (n)

Returns the computed cosine of the value (n) expressed in radians

Ex: Find $Y = \cos(X)$ if you know that $X = 6.8$ rad

$X = 6.8$

$Y = \cos(X)$

Print Y

Ans: $Y = 0.88888888988818$

11. Tan (n)

Returns the computed tangent of the value (n) expressed in radians

Ex: Find $Y = \tan(X)$ if you know that $X = 2.5$ rad

$X = 2.5$

$Y = \tan(X)$

Print Y

Ans: $Y = -0.74702229723866$

12. Atn (n)

Returns the arctangent of the value (n) expressed in radians and it's also called arc length for the angle (n).

Ex: Find the arc length for the angle (X) in radian if you know that

X= 6.7 rad

X=6.7

Y = Atn(X)

Print Y

Ans: Y= 0.958113088595432

Note: If you compute trigonometric values on arguments expressed as degrees instead of radians, multiply the argument by π and divide by 180.

Ex: Find Y= sin (X) if you know that X= 3.7^o

Const pie =3.14

X = 3.7 * (pie / 180)

Y = Sin (X)

Print Y

Ans: Y= 6.44996385772737E-02

4-2-4: Conversion Functions

Visual Basic supplies many functions that convert the argument from a specific data type to another. When writing applications, you might need to round numbers down or up to their nearest integers or we need to convert a number temporary to a specific data type so we will use these functions. Table below contains the most well-known functions.

1. Asc (S)

Converts the string value (S) to an equivalent ASCII number>

Ex: Find the Ascii code for the dollar sign (\$).

S="\$"

Y = Asc(S)

Print Y

Ans: Y= 36

2. Chr (n)

Converts the argument number to an equivalent ASCII .

Ex: What is the equivalent ASCII symbol for the number (34)?

n = 38

Y = Chr (n)

Print Y

Ans: Y= &

3. CInt (n)

Rounds fractional values of 0.6 and more to the next highest integer (like Round())

Ex: What is the result after execution the following steps:

```
X= 4.8  
Y= CInt (X)  
Print Y
```

Ans: Y= 5

4. CLng (n)

Converts the argument (n) to an equivalent long integer data type

Ex: What is the result after execution the following steps:

```
X = 22747475.656  
Y = CLng(X)  
Print Y
```

Ans: Y= 22747476

5. CSng (n)

Converts the argument (n) to an equivalent single-precision data type

Ex: What is the result after execution the following steps:

```
X=22/7  
Y = CSng(X)  
Print Y
```

Ans: Y= 3.142857

6. CDbI (n)

Converts the argument (n) to an equivalent double-precision data type

Ex: What is the result after execution the following steps:

X=22/7

Y = CDbI(X)

Print Y

Ans: Y= 3.14285714285714

7. Hex (n)

Converts the decimal number equivalent hexadecimal

Ex: What is the result after execution the following steps:

X=15

Y = Hex (X)

Print Y

Ans: Y= F

8. Oct (n)

Converts the decimal number equivalent octal

Ex: What is the result after execution the following steps:

X = 9

Y = Oct (X)

Print Y

Ans: Y= 11

9. Int (n)

Rounds the number (n) down to the integer less than or equal to its arguments

Ex: What is the result after execution the following steps:

```
X = -8.1  
Y = Int (X)  
Print Y
```

Ans: Y= -9

10. Fix (n)

Truncates the fractional portion

Ex: What is the result after execution the following steps:

```
X=9.8  
Y = Int (X)  
Print Y
```

Ans: Y= 9

11. CStr (n)

Converts the argument (n) to an equivalent string data type

Ex: What is the result after execution the following steps:

```
X = Cstr(3)  
Y = Cstr(7)  
Z=X+Y  
Print Z
```


Ans: Z= 37

12. Val (s)

Converts the string argument to an equivalent numbering data type.

4-2-5: String Functions

There are several Visual Basic string functions that manipulate string data better than most other programming languages allow. The string functions allow you to strip away characters from string data and change strings in various ways. Table below lists these functions.

1. LCase (s)

Returns the string argument as a variant data type that's all lowercase letters

Ex: What is the result after execution the following steps:

X = "AL-MUSTANSIRIYA UNIVERSITY"

Y= LCase (X)

Print Y

Ans: Y= al-mustansiriya university

2. UCase (s)

Returns the string argument as a variant data type that's all uppercase letters

Ex: What is the result after execution the following steps:

```
X = "college of engineering"
```

```
Y = UCase(X)
```

```
Print Y
```

Ans: Y= COLLEGE OF ENGINEERING

3. Left (s, n)

Returns partial string argument contains (n) characters from the left of the main string

Ex: What is the result after execution the following steps:

```
X = "Electrical Engineering Department"
```

```
Y = Left(X, 22)
```

```
Print Y
```

Ans: Y= Electrical Engineering

4. Right (s, n)

Returns partial string argument contains (n) characters from the right of the main string

Ex: What is the result after execution the following steps:

```
X = "First Class"
```

```
Y = Right (X, 5)
```

```
Print Y
```

Ans: Y= Class

5. **Mid (s, n1, n2)**

Returns partial string argument contains (n2) characters starting from character number (n1) of string argument (s)

Ex: What is the result after execution the following steps:

```
X = "Electric"
Y = Mid(X, 3, 4)
Print Y
```

Ans: Y= ectr

6. **LTrim (s)**

Returns a string argument contains trimmed left spaces from the main string

Ex: What is the result after execution the following steps:

```
X = "  Engineer  "
Y = LTrim(X)
Print Y
```

Ans: Y= Engineer

7. **RTrim (s)**

Returns a string argument contains trimmed right spaces from the main string.

Ex: What is the result after execution the following steps:

```
X = "  Engineer  "
```

```
Y = LTrim(X)
```

```
Print Y
```

Ans: Y= □Engineer

8. Trim (s)

Returns a string argument contains trimmed spaces from the main string

Ex: What is the result after execution the following steps:

```
X = "□Engineer□"
```

```
Y = LTrim(X)
```

```
Print Y
```

Ans: Y= Engineer

9. Len (s)

Returns a number represents the string length.

Ex: What is the result after execution the following steps:

```
X = "Ministry of Higher Education"
```

```
Y = Len (X)
```

```
Print Y
```

Ans: Y= 28

10. Replace (s,s1,s2)

Returns a string contains the replaced string (s1) from the main string (s) by string (s2).

Ex: What is the result after execution the following steps:

X = "Physics Lectures"

X1 = "Physics"

X2 = "Computer"

Y = Replace(X, X1, X2)

Print Y

Ans: Y= Computer Lectures

11. StrReverse (s)

Returns a string argument reversed from the main string

Ex: What is the result after execution the following steps:

X="ABCDE"

Y = StrReverse(X)

Print Y

Ans: Y= E D C B A

4-2-6: Date and Time Functions

The date and time functions return values set inside your computer so that you can use and display those values in your program. Table below lists these functions.

1. Day(d)

Returns the day number in the date argument (d).

Ex: Write a program to find the day number for the date (8/7/2007)

```
X = # 8/7/2007#
```

```
Y = Day (X)
```

```
Print Y
```

Ans: Y= 8

2. Month(d)

Returns the month number in the date argument (d)

Ex: Write a program to find the month number for the date
(26/12/2011)

```
X = # 24/1/2013#
```

```
Y = Month (X)
```

```
Print Y
```

Ans: Y= 12

3. Year(d)

Returns the year number in the date argument (d)

Ex: Write a program to find the year for the date (1/7/2010)

```
X = # 24/1/2013#
```

```
Y = Year (X)
```

```
Print Y
```

Ans: Y= 2013

4. MonthName (n)

Returns the month name that its order is (n).

Ex: Write a program to find the month name for the forth month.

```
X = 4  
Y = MonthName(X)  
Print Y
```

Ans: Y= April

5. WeekDayName (n)

Returns the day name that its order is (n)

Ex: Write a program to find the day name for the fifth week day.

```
X = 5  
Y = WeekdayName(X)  
Print Y
```

Ans: Y= Wednesday

6. Second (t)

Returns the second number in the time argument (t)

Ex: Write a program to find the seconds for the time (01:25:30 AM)

```
X = #1:25:30 AM#  
Y = Second(X)  
Print Y
```

Ans: Y= 30

7. Minute (t)

Returns the minute number in the time argument (t)

Ex: Write a program to find the minutes for the time (10:22:10 PM)

```
X = #10:22:10 PM#
```

```
Y = Minute(X)
```

```
Print Y
```

Ans: Y= 22

8. Hour (t)

Returns the hour number in the time argument (t)

Ex: Write a program to find the Hours for the time (03:12:40 AM)

```
X = #03:12:40 AM#
```

```
Y = Hour(X)
```

```
Print Y
```

Ans: Y= 3

Note :-As we enclosed the string values by double quotation marks "-", we enclosed the date and time values by double hash marks # - #.

4-2-7: User Functions

These types of functions are not found in the visual basic library because these types are built according to special calculations. The programmer will build these functions according to its requisites with the use of the following form:

```
Private Function FName (V-1 As Type-1,..., V-n As Type-n) As function Type
```

```
    Function body
```

```
End Function
```


Then we can call the function using the following form:

Vname = FName (parameters)

Where:

FName: The name of the function you want to make it (may take any symbol or any string)

V-1 to V-n: The function variables (parameters) used in the program.

Type-1 to Type-n: The data types for the function variables (inputs).

Function Type: The data types for the function output.

Function body: Any relation between function name and their variables.

Vname: The resulted variable after calling the function and applying their parameters.

Example:

As we know the trigonometric functions Sin, Cos,...etc, returns a suitable argument according to the applied angle in radian unit. Implement the function "Sec" then enter any value through the main program then find Sec for this number.

Solution

Function Sec(n As Double) As Double

Sec = 1 / Cos(n)

End Function

```

Private Sub Form_Click()
Dim X#, Y#
X = Val(InputBox("Enter any value"))
Y = Sec(X)
Print Y
End Sub

```

Example:

Implement the a function named "Sum" to evaluate a summation of three student degrees (X, Y, Z). Enter the degrees and print the result through the main program.

Solution

```

Private Function Sum (X, Y, Z As Single) As Single
Sum = X + Y + Z
End Function

```

```

-----
Private Sub Form_Click ()
Dim deg1!, deg2!, deg3!, S!
deg1 = Val (InputBox ("enter the first degree"))
deg2 = Val (InputBox ("enter the second degree"))
deg3 = Val (InputBox ("enter the third degree"))
S = Sum (deg1, deg2, deg3)
Print "The sum is: "; S
End Sub

```

Note: For more functions you can see **Appendix-C** sorted in alphabetic manner

4.3 Problems

1- Design a project contains a command button "Find" and two text boxes. When we enter a number (represents the radius of a circle) from the first text box, the area of this circle will appeared on the second text box and when we click the command "Find".

2- What is the result of the following program?

```
Private Sub Form_Click()
Dim A, B, C, D As String
Dim E, F, G As Boolean
A = "I will install"
B = "Visual Basic version 6.0"
C=A+B
D = A & B & "in my computer"
E = B Like "Visual Basic version 7.0"
F = B Like "Visual ?asic version 6.0"
G = B Like "Visual Basic version #.0"
Print C; Print D
Print E; F; G
End Sub
```

3- What is the result of the following program?

```
Private Sub Form_Click()
Dim A, B, C, D As Boolean
Dim Res1, Res2, Res3, Res4 As Boolean
```

```
A = True
B = True
C = False
D = False
Res1 = Not D
Res2 = A Or B
Res3 = A And D
Res4 = B Xor C
Print "Res1="; Res1
Print "Res2="; Res2
Print "Res3="; Res3
Print "Res4="; Res4
End Sub
```

4- What is the result of the following program?

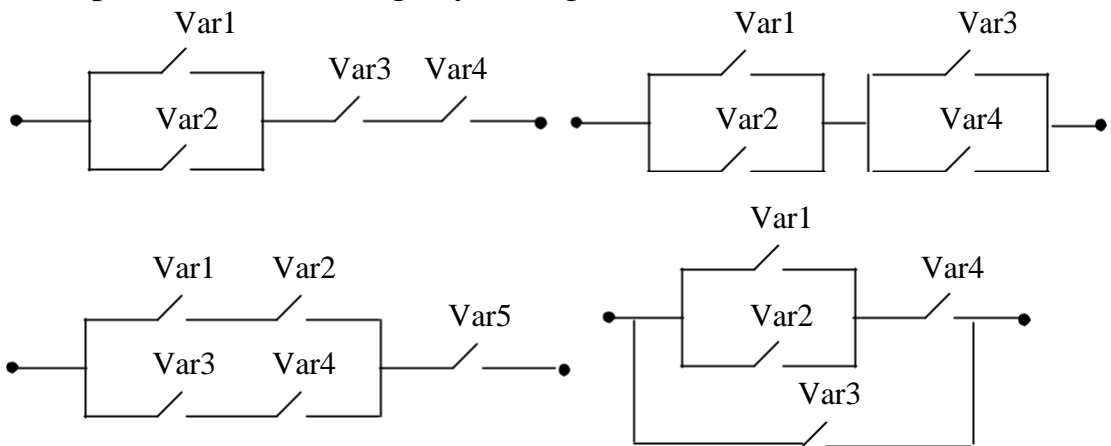
```
Private Sub Form_Click()
Dim A, B, C, D As Integer
Dim V1, V2, V3, V4, V5, V6 As Boolean
A=50:B=70:C=80:D=90 V1 = (A > B)
And (B < C)
V2 = (A <> B) And (B <> C)
V3 = (A > C) Or (D > A)
V4 = V1 And (V2 Or V3) Or (A < C)
V5 = ((A + B) > A / B) And (((D - C) / 2) > (B \ A))
V6 = Not (((A + D) ^ 2 = (B / C) ^ 2) Xor (D ^ 2 = (C \ A)))
Print V1; Spc(2); V2; Spc(2); V3
Print V4; Spc(2); V5; Spc(2); V6
End Sub
```

5. Use two Text Boxes to enter two values and display their product on a third Text.

6. Use three Text Boxes to enter three values and display their sum on a fourth Text.
7. Write a VB program to find and view the area of a triangle on a text box when we enter the base and height using two text boxes.
8. Write a VB program to enter three strings through three text boxes then concatenate and display them on a label box.
9. What is the result after executing the following program?

```
Private Sub Form_Click()
Dim A, B, C, D As Integer
Dim E, F, G, H As Boolean
A=100:B=100:C=140:D=133
E = (A = B) And (Not ((B = C)))
F = ((A < D) Xor (C <> D)) Or (C >= B)
G = ((A / 100) = ((C - 40) / 100)) And (F Or E)
H = G And ((E Or F) Or (A ^ 2 <= 100 * B))
Print E; Spc(2); F; Spc(2); G; Spc(2); H
End Sub
```

6. Represent the following keys in logical relations:

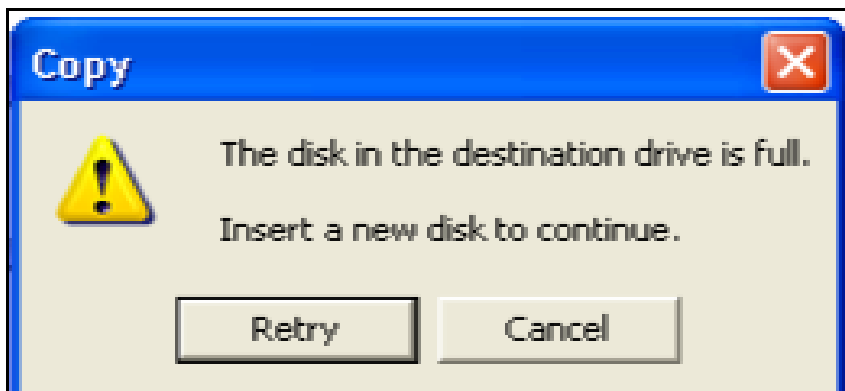


- Write a VB program to enter two numbers through two text boxes then display their division, sum, subtract using three label boxes.
- Use two input boxes named "First" and "Second" to enter two numbers when we click the command "Loading Numbers". Another two commands "Addition" and "Subtraction" are used to print the addition and subtraction results for these two numbers when we click them respectively.
- If you know that the radius of the circle inscribed in triangle with sides a, b, c is:

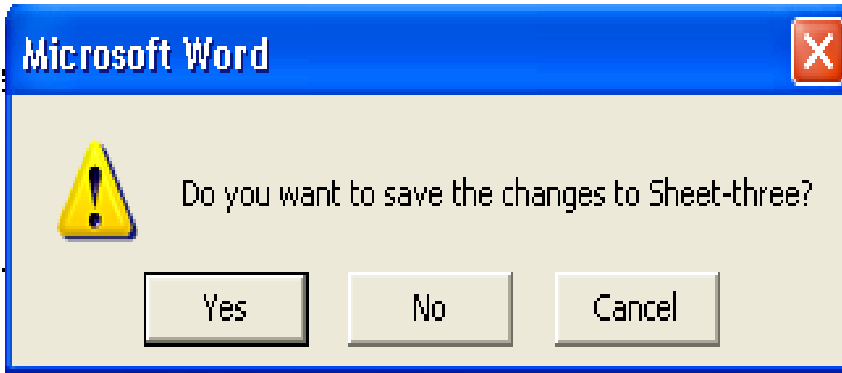
$$r = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}} \quad \text{where } s = \frac{1}{2}(a+b+c)$$

Input the triangle sides by using three input boxes then find and display the radius (r).

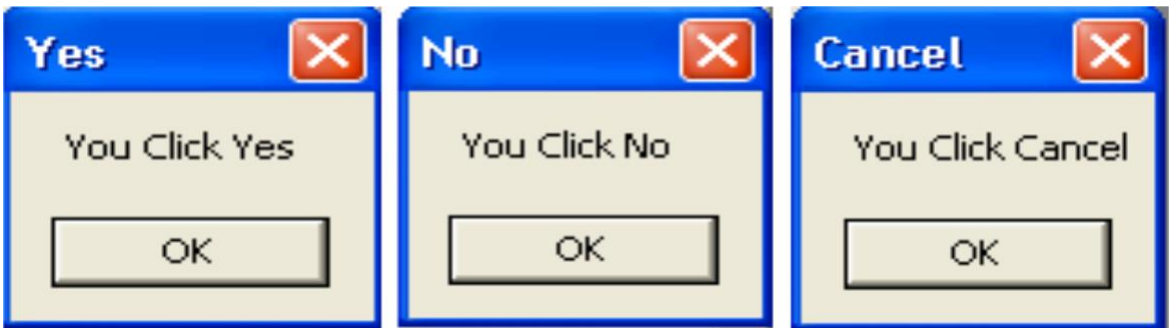
4. Write a VB program asks the user to enter his name and age then display them on two message boxes.
5. Write a VB program to view the below message box when you click the command "Disk".



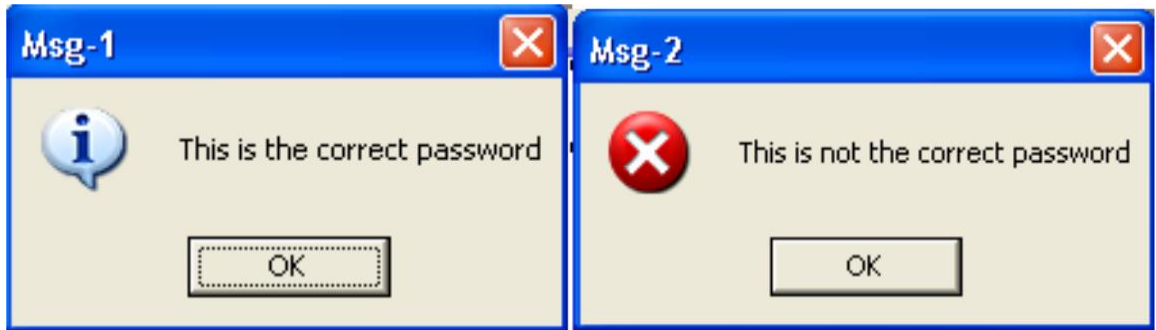
- Write a VB program to view the below message box when you click the form



Then view the following message boxes when we click the above selected button respectively.



- Write a VB program to make a password. Enter any word by using an input box then compare it with a previously stored word (let it be "VBasic"). If the two words are matched the below left message box will appear, else the below right box appeared.



- Write a VB program to enter any two numbers (x and y) via an input box then find and print the value of Z from the following equations:

$$Z = \sin(x^2) + \tan^3\left(\frac{y}{3}\right)$$

$$Z = \ln\left(\frac{x}{y}\right)^4 - \sqrt{x^2 + y^2}$$

- Write a VB program to enter any character in a text box then display the equivalent ASCII value in another text box after pressing the command "Convert".
- Write a VB program to display the present time and date on two text boxes then split these information to six values and display them on six text boxes after pressing the commands "Time" and "Date".
- As you know that the function Log(x) returns the natural logarithm for argument (x) (i.e. Ln(x)). Write a VB program to implement the function LogEx(x,b) to find the logarithm for the argument (x) with any base (b) then find and print Log₄(9).

Hint: Use the relation $\text{Log}_b(x) = \frac{\ln(x)}{\ln(b)}$

22. Write a VB program to implement the function NthRoot (x, N) to return the Nth root for the argument (x) (i.e. $\sqrt[N]{x}$). Test the function by finding $\sqrt[3]{16}$.

□ As you know that the statement Rnd generates a random number in the interval [0,1], write a VB program to implement the function RndMinMax(Min, Max) to generate a random number limited by the interval [Min, Max]. Test the function by generating a number in the interval [2, 10].

Hint: Use the relation $\text{Int}((\text{Max}-\text{Min}+1)*\text{Rnd}+\text{Min})$

□ Write a VB program to enter any number then print the triple of this number. Use the function Triple(n) to find the triple of any number.

Write a VB program to evaluate the following by the use of user functions:

$$\square = \begin{cases} 3 \sinh^{-1} \left(\frac{x}{2} \right) + x \cosh^{-1} \left(\frac{\sqrt{x^3}}{2} \right) \\ x \sinh^{-1} \left(\frac{1}{2} \right) + 2 \cosh^{-1} \left(\sqrt{\frac{3}{2}} \right) \end{cases}$$

$\left| \cos(2 \cosh^{-1}(z^2)) \right|$

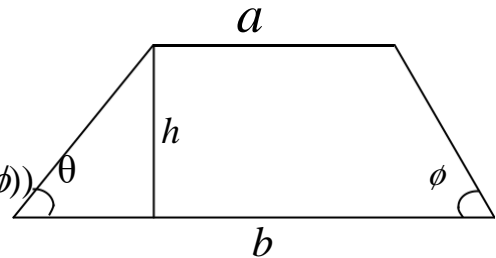
Find y for each entered value for the three equations at the same time.

Hint: $\sinh^{-1}(u) = \ln(u + \sqrt{u^2 + 1})$ and $\cosh^{-1}(u) = \ln(u + \sqrt{u^2 - 1})$

- Write a VB program to find and print the area and the perimeter (circumference) for the shape below from the following equations:

$$Area = \frac{1}{2}h(a + b)$$

$$Perimeter = a + b + h(\csc(\theta) + \csc(\phi))$$



Enter the unknown variables via an input boxes then print the result on two message boxes. Use the user function to define any unknown functions.

- 2)** Write a VB program to find and print Y from the following equation:

$$Y = \frac{\cosh(x) + \sinh^2(x) - 4 \cosh(x) \cdot \sinh(x)}{4x^2 + 2 \sinh(x) - \cosh^2(x) - 6}$$

Use the user function to define the sinh() and cosh() functions using the relations:

$$\text{Sinh}(x) = \frac{e^x - e^{-x}}{2} \quad \text{and} \quad \text{Cosh}(x) = \frac{e^x + e^{-x}}{2}$$

CHAPTER FIVE

Conditional Statements and Functions

Introduction

Computers cannot think on their own, but with your help they can be taught to make decisions based on values contained in controls and variables. Visual Basic's decision-making capability enables it to calculate applications based on certain conditions, to print exception reports, and to check user responses by means of the form's controls. In this chapter, you will learn some new programming statements and functions that you can use along with the ones you already know to write programs that make data-based decisions. We need the comparison Operators (Relational and Conditional operators) listed in the previous chapter to accomplish the conditional statements and functions.

5-1: Conditional Statements

The relational operators are sometimes called the conditional operators because they test conditions that are either true or false. The conditional statements compare values against one another. You can compare for equality, inequality, and size differences. We can also use

the logical operators to connect two or more conditional operators. In this section we will illustrate four types of conditional statements:

1. **IF- Statement.**
2. **IF_Else- Statement.**
3. **IF_Else If- Statement.**
4. **Select Case- Statement.**

5-1-1: IF- Statement

The **If-statement** uses the relational operators to test data values. It performs one of two possible code actions, depending on the result of the test. In other words, the If statement uses the relational operators to test data and might execute one or more lines of subsequent code, depending on the results of the test. The **If-statement** makes decisions. If a relational test (**condition**) is true, the body of the **If-statement** (**comment**) will be executed. Here is the format of **If-statement** :

```
If (Condition) Then  
    Comment  
End If
```

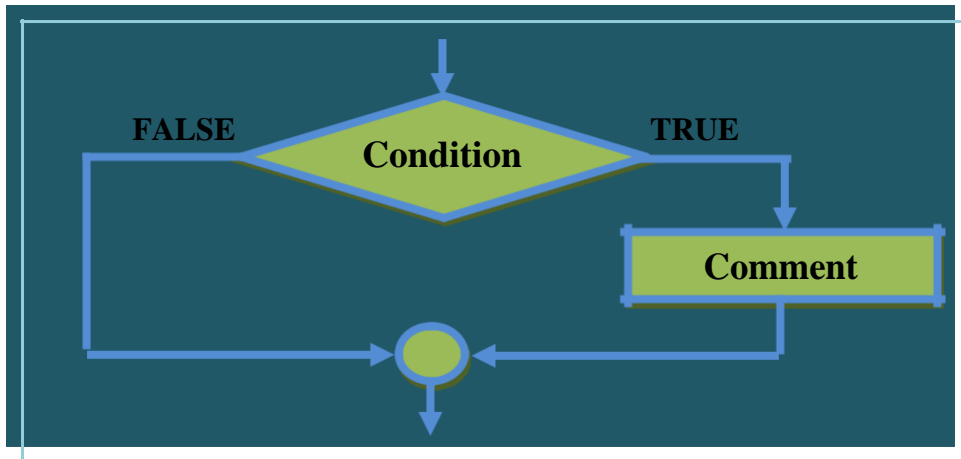
The **End If** statement informs Visual Basic where the body of the If statement ends.

There is a shortcut form of If statement that you might run across.

The single-line If statement has a format that looks like this:

```
If (Condition) Then Comment
```

The single-line If does not require an End If statement because relational test and the body of the If reside on the same line. The flow chart that represents the If statement is shown in **figure (5-1)**.



Figure(5-1): If-Statement flowchart

Example-1

This program will learn you how to deal with keyboard keys. Write a VB program to display the message ("You Click F9 Button") if you click the key F9 from the keyboard and the message ("You Click F10 Button") if you click the key F10.

Solution

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
```

```
    If KeyCode = 120 Then
```

```
        MsgBox "You Click F9 Button"
```

```
    End If
```

```
    If KeyCode = 121 Then
```

```
        MsgBox "You Click F10 Button"
```

```
    End If
```

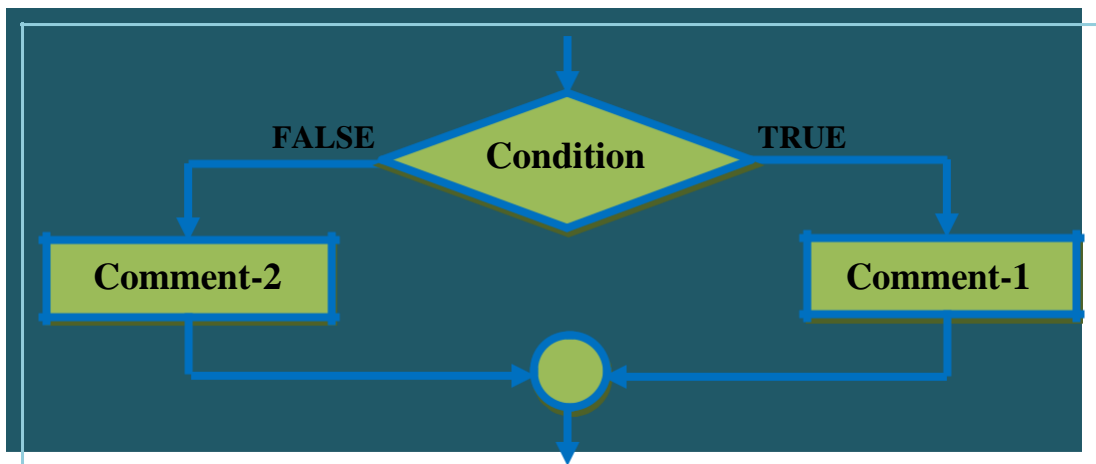
```
End Sub
```

5-1-2: IF_Else - Statement

Whereas **If**-statement executes code based on the relational test's true condition, the **Else** statement executes code based on the relational test's false condition. **Else** is actually part of the **If**-statement. Here is the complete format of the **If-Else** statement :

```
If (Condition) Then  
    Comment-1  
Else  
    Commment-2  
End If
```

The flow chart that represents the **If_Else**- statement is shown in figure (5-2) below:



Figure(5-2): If_Else- Statement flowchart

Example-2

Write a VB program to find a solution for the following equation:

$$Z = \frac{\sqrt{X+Y}}{X} \quad \text{Where } X+Y \geq 0, X > 0$$

Enter X and Y and print Z then display the message "Wrong Values" on a message box if the two conditions above are not satisfied.

Solution

```
Private Sub Form_Click()  
Dim X!, Y!, Z!  
X = Val (InputBox ("Enter the value of X"))  
Y = Val (InputBox ("Enter the value of Y"))  
  
If (X + Y) >= 0 And X > 0 Then  
Z = (Sqr (X + Y)) / X  
Print "The value of Z is: "; Z  
Else  
MsgBox "Wrong Values", vbCritical, "Error"  
End If  
  
End Sub
```

Example-3

Design a VB project labeled "Lucky Seven" contains the command "Spin" and four text boxes. When we click the command, a three randomly generated numbers will appear on the three text boxes. If one of these boxes contains number 7 the string "You Win" will appear on the fourth text box else the string "You Loss" appeared.

Solution

```
Private Sub cmdSpin_Click()
```

```
Dim A%, B%, C%
```

```
A = 10 * Rnd
```

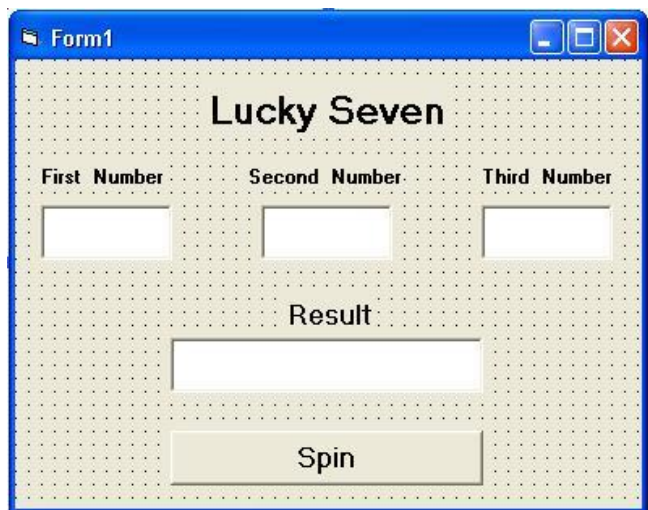
```
B = 10 * Rnd
```

```
C = 10 * Rnd
```

```
Txt1.Text = A
```

```
Txt2.Text = B
```

```
Txt3.Text = C
```



```
If (A = 7) Or (B = 7) Or (C = 7) Then
```

```
txtResult.Text = "You Win"
```

```
Else
```

```
txtResult.Text = "You Loss"
```

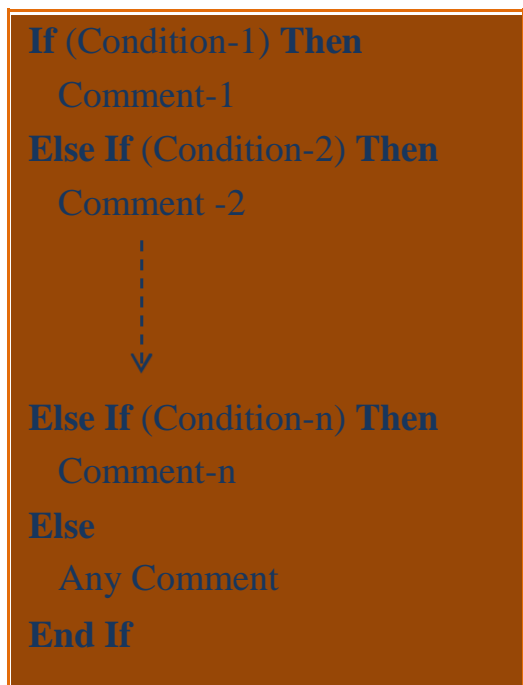
```
End If
```

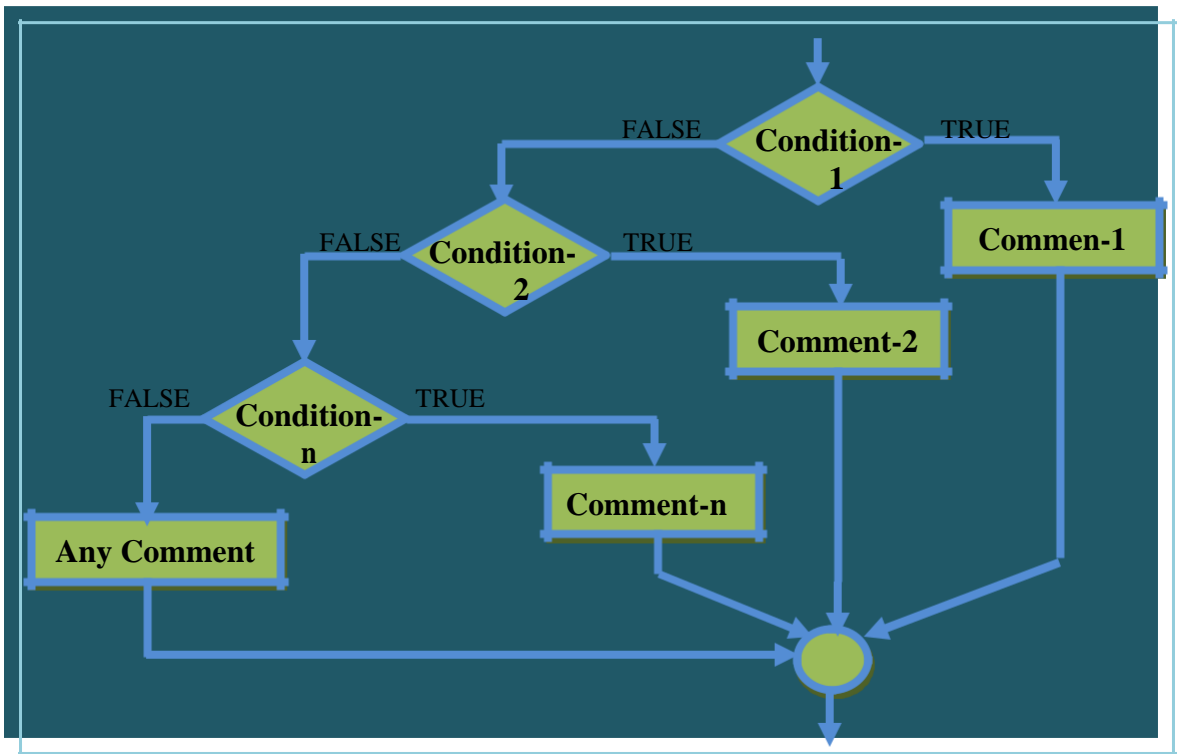
```
End Sub
```


5-1-3: IF_Elself- Statement

The **If** and **If-Else** statement is great when you must test against more than two conditions, however, the **If** and **If-Else** becomes difficult to maintain. Although the logic of the **If-Else** statement is simple, the coding is extremely difficult to follow. Visual Basic supports a statement, called **If-ElseIf**, which handles such multiple-choice conditions better than **If-Else**.

Here is the format of the **If-ElseIf** statement and also its flowchart shown in figure (5-3):





Figure(5-3): If_Else If- Statement flowchart

Example-4

Design a VB project contains three text boxes. The first is used to enter a number represents a centigrade degree. The second text box displays the Fahrenheit degree that generated from the first degree according to the relation $(F = (9/5) * C + 32)$. The third text box is used to display the below phrases according to their equivalent Fahrenheit degree:

“Cold” when $F \leq 41$.

“Nice” when $41 < F \leq 77$.

“Hot” when $F > 77$.

Solution

```
Private Sub Form_Click()  
Dim C!, F!  
C = Val (txtC.Text)  
F=(9/5)*C+32  
txtF.Text = F  
If (F <= 41) Then  
txtResult.Text = "Cold"  
ElseIf (F > 41) And (F <= 77) Then  
txtResult.Text = " Nice"  
ElseIf (F > 77) Then  
txtResult.Text = "Hot"  
End If  
End Sub
```

5-1-4: Select Case- Statement

The **Select Case** statement is a good substitute for long, nested **If-ElseIf** conditions when one of several choices is possible. You set up your Visual Basic program to execute one set of Visual Basic statements from a list of statements inside **Select Case**.

Select Case can have three Case value sections based on three kinds of matches:

1. An exact Case match to Select Case's Expression
2. A conditional Case match to Select Case's Expression
3. A range of Case matches to Select Case's Expression

First format: An exact Case match to Select Case's Parameter. The format of this case is as shown:

```
Select Case Parameter
```

```
Case no.1
```

```
    Comment-1
```

```
Case no.2
```

```
    Comment-2
```

```
    .      .
```

```
    .      .
```

```
    .      .
```

```
Case no.n
```

```
    Comment-n
```

```
Case Else
```

```
    Any Comment
```

```
End Select
```

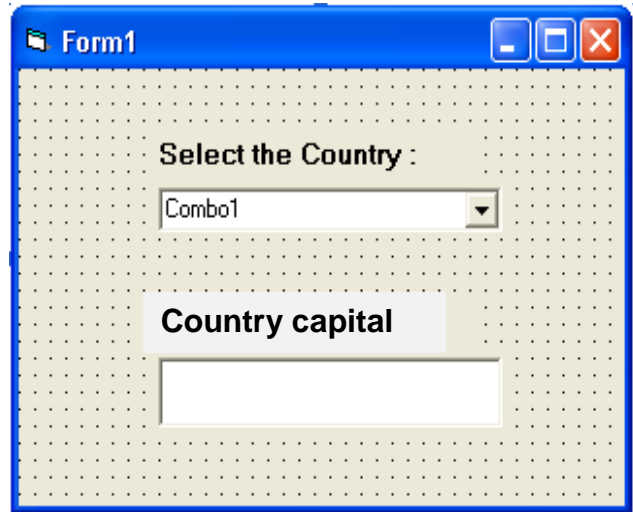
Where **no.1** to **no.n** are integer numbers

Example-5

Design a VB project contains a combo box and a text box. The combo box contains a list of countries (Iraq, Germany, Lebanon, Egypt, France) added through the code. The capital of this country will be displayed on the text box when you select any of these countries from the combo box.

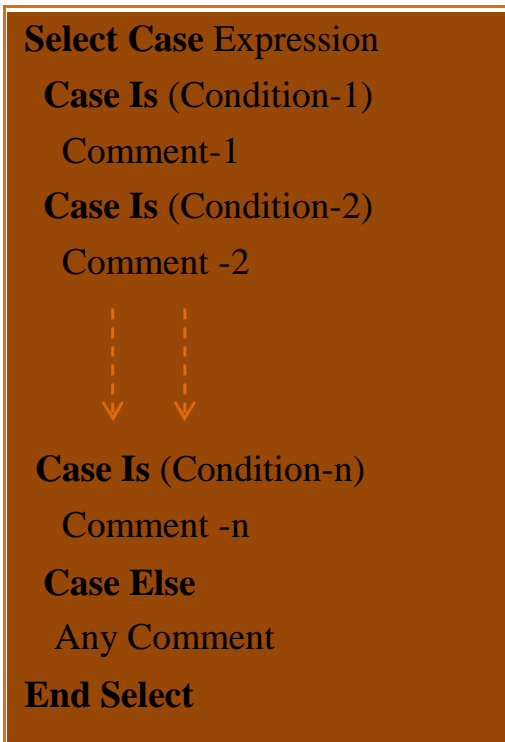
Solution

```
Private Sub Form_Load()  
    Combo1.AddItem "Iraq"  
    Combo1.AddItem "Germany"  
    Combo1.AddItem "Lebanon"  
    Combo1.AddItem "Egypt"  
    Combo1.AddItem "France"  
End Sub
```



```
Private Sub Combo1_Click()  
    Select Case Combo1.ListIndex  
        Case 0  
            txt1.Text = "Baghdad"  
        Case 1  
            txt1.Text = "Berlin"  
        Case 2  
            txt1.Text = "Beirut"  
        Case 3  
            txt1.Text = "Cairo"  
        Case 4  
            txt1.Text = "Paris"  
    End Select  
  
End Sub
```

Second format: A conditional Case match to Select Case's Expression. The format of this case is as shown:



Example-6

Write a VB program to solve the below equations depending on the entered X value.

$$\begin{cases}
 X^3 + 3e^{-X} & , X \leq 0 \\
 \sin(X) + \cos(X) & , 0 < X \leq 3\pi \\
 \ln(X) + \tan(X) & , 3\pi < X \leq 9\pi \\
 \sin^2(X) + X^3 & , X > 9\pi
 \end{cases}$$

Solution

```
Private Sub Form_Click()  
Dim X!, Y!  
Const Pie = 22 / 7  
X = Val (InputBox ("Input the X value ", "X"))  
  
Select Case X  
Case Is <= 0  
Y = X ^ 3 + 3 * Exp(-X)  
Case Is > 0 And X <= 3 * Pie  
Y = Sin(X) + Cos(X ^ 2)  
Case Is > 3 * Pie And X <= 9 * Pie  
Y = Log (X) + Tan (X)  
Case Is > 9 * Pie  
Y = (Sin(X)) ^ 2 + X ^ 3  
End Select  
Print "Y=";Y  
  
End Sub
```

Third format: A range of Case matches to Select Case's Expression.

The format of this case is as shown:

```

Select Case Expression
  Case Val.1 to Val.2
    Comment -1
  Case Val.3 to Val.4
    Comment -2
    ↓      ↓
  Case Val.n to Val.n+1
    Comment -n
Case Else
  Comment-n+1
End Select

```

Where Val.1 to Val.n+1 are any numbers.

Example-7

Write a VB program to view the student Grade in a message box from the below table when we enter its degree in an input box and when we click the command "Show".

Degree	Grade
0-49	Weak
50-59	Fair
60-69	Medial
70-79	Good
80-89	Very Good
90 - 100	Excellent

Solution

```
Private Sub cmdShow_Click()
```

```
Dim D As Single
```

```
D = Val (InputBox("Enter the degree", "Degree"))
```


Select Case D

Case 0 To 49

MsgBox "The Grade is Weak"

Case 50 To 59

MsgBox " The Grade is Accept"

Case 60 To 69

MsgBox " The Grade is Medial"

Case 70 To 79

MsgBox " The Grade is Good"

Case 80 To 89

MsgBox " The Grade is Very Good"

Case 90 To 100

MsgBox "Excellent"

Case Else

MsgBox " The Grade is Wrong degree"

End If

End Sub

Note: We can use all Select Case formats in the same problem.

5-2: Conditional Functions

In addition to Conditional statements we can use the Conditional function to represent the problems that has many conditions. The advantage of these function is simplicity to the use in problems in a single line step. The only difference between the conditional statements and functions is that the second contains many parameters enclosed by brackets but the first did not contain any brackets. So we can divide the conditional function to three types:

5. IIF- Function

6. Choose- Statement

7. Switch- Statement

5-2-1: IIF- Function

This function is the same as **If_Else**- Statement, so its contain a single condition and two comments as shown in its form:

`Var = IIF (Condition , Comment-1 , Comment -2)`

Where:

Var: The restore value from the return comparison result.

Comment-1: If **condition** is satisfied.

Comment-2: If condition is not satisfied.

Example-8

Write a VB program to enter any number (X) then number. Display the result on the form and on a message box.

Solution

```
Private Sub Form_Click()  
Dim X!, Y$  
X = Val (InputBox ("Enter any number"))  
Y = IIf (X Mod 2 = 0, "even number", "Odd number")  
Print Y  
MsgBox Y  
End Sub
```

5-2-2: Choose- Function

This function is the same as **Select Case**-Statement (First format), so its contain many comments (Comment-1 to Comment-n) that can be selected according to agreement of the parameter value which is varies from 1 to n according to the comment number selection as shown in its form:

```
Var = Choose (Parameter , Comment-1 , Comment-2 , ..... , Comment-n)
```

Example-9

Write a VB program to enter two numbers (X and Y) then enter the operation number (N) according to the following: 1. Addition 2. Subtraction 3. Multiplication 4. Division. Display the result on a message box.

Solution

```
Private Sub Form_Click()
Dim X!, Y!, N%, Sol!
X = Val(InputBox("Enter the first number"))
Y = Val(InputBox("Enter the second number"))
N = Val(InputBox("Enter your operation choice from 1 to 4"))
Sol = Choose(N, X + Y, X - Y, X * Y, X / Y)
MsgBox Sol
End Sub
```

5-2-3: Switch- Function

This function is the same as Select Case- Statement (second format) or If_Else If- Statement, so its contain many conditions (Condition.1 to Condition.n) and many comments (Comment-1 to Comment-n) that can be selected successively according to the following form.

```
Var = Switch (Condition-1 , Comment-1 _
              Condition-2 , Comment-2 _
              . . .
              . . .
              Condition-n , Comment-n)
```

Note: In Visual Basic, the Under score symbol (_) represent a completion for this line in the next line.

Example-10

If you know that the Tax office takes a tax for the imported goods according to their degrees as shown:

Goods degree	Tax
"A"	40%
"B"	10%
"C"	5%
"D"	2%
"E"	1%

Write a VB program to enter the goods price and its degree then print the tax according to the table above.

Solution

```
Private Sub Form_Click()
Dim G$
Dim Tax As Currency
Dim Price As Currency
Price = Val (InputBox ("Enter the goods price"))
D = UCase (InputBox ("Enter the tax degree"))
Tax = Switch (D = "A", 0.4 * Price, _
D = "B", 0.1 * Price, _
D = "C", 0.05 * Price, _
D = "D", 0.02 * Price, _
D = "E", 0.01 * Price)
Print "Tax="; Tax
End Sub
```

5.3: Problems

Note: Use several methods to solve the same question.

8. Write a VB program to enter two numbers then compare them and display the comparison result on a message box.
9. Write a VB program to enter a number then display the message "Even Number" if the entered number is even and the message "ODD Number" if it is odd.
10. Write a VB program to enter a string then display the message boxes "Greater than 6 characters", "Less than 6 characters", and "Equal to 6 characters" if this number was greater, less, and equal to six characters respectively.
11. Write a VB program to enter a character represent the person gender (M: for male, F: for female) and a number represents person length ((L) in inch) then find and print its perfect weight ((W) in pound) according to the following relations:

For male (M) : $W = (L \times 4) - 125$

For female (F): $W = (L \times 3.5) - 108$

6. Suppose the random bank offers 9% interest on balances of less than \$5000, 12% for balances of \$5000 or more but less than \$10000, and 15% for balances of \$10000 or more. Write a VB program to enter a person balance then calculates a customer's new balance after one year.

7. Write a VB program to find W from the equations:

$$W = \begin{cases} X+Y & : X>0, Y<0 \\ \frac{X^2 + Y^3}{2} + 5X & : X=0, Y>0 \\ \frac{2X}{3} + 4Y & : X<0, Y=0 \end{cases}$$

Print W for each input values of X and Y.

10. Write a VB program to find the roots (X_1, X_2) for the quadratic equation: $ax^2 + bx + c$ using the formula:

$$X_1, X_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Enter the constants (a, b, c) then check the following points:

- if $a = 0$ display the message "Divided by zero".
- if $b^2 < 4ac$ display the message "No real roots".
- if $b^2 = 4ac$ display the message "Equal roots" then find the roots from the formula above.
- if $b^2 > 4ac$ display the message "real roots" then find the roots from the formula above.

6. Write a VB program to enter a number represents a person age then display the following on a message boxes:

- "Wrong age" if the age less than or equal to 0 years.
- "Child" if the age less than 8 years.
- "Boy" if the age greater than or equal to 8 years.
- "Young" if the age greater than or equal to 18 years.
- "Old" if the age greater than or equal to 35 years.
- "Very Old" if the age greater than or equal to 65 years.

7. Write a VB program to find z from the below equations.

$$z = \begin{cases} 2k - \sin(k) & \square & k = 1 \\ k - 2 & \square & k = 2 \\ k & \square & k = 3 \\ k & \square & k < 1 \text{ or } k > 3 \end{cases}$$

Print z for each input value of k.

- Write a VB program to find x from the below equations according to your choice entry from 1 to 4.

$$x = \sin(t) + \tan(t)$$

$$x = \cosh(t^3 + 2t)$$

$$x = \sec^2(4t) + t^2$$

$$x = 0.25 t^4 + t^2$$

Print x for each input value of t. Define the unknown functions.

- Write a VB program to find and print T from the below equations where a and b are input variables.

$$T = \begin{cases} a^2 + ab + \sqrt{ab} & \mathbf{a < b \text{ AND } a < 10} \end{cases}$$

$$\begin{cases} ab^2 - 2a + 5b & \mathbf{a = b \text{ OR } b > 10} \end{cases}$$

- By using Switch function write a VB program to enter any number then display the following on a message box:

- 1- "Divisible by 7" if the entered number can be divisible by 7.
- 2- "Odd number" if the entered number was odd number.
- 3- "Even number" if the entered number was even.

- Define f(x) as follows:

$$f(x) = \begin{cases} 3 - x & \text{if } x < 2 \\ 2 & \text{if } x = 2 \\ x/2 & \text{if } x > 2 \end{cases}$$

Write a VB program to calculate f(x) from the above equation then print f(x) for any input value of x.

CHAPTER SIX

Looping Statements and Arrays

Introduction

We need loops in the program to repeat an operation or group of operations to specific (or non specific) number of times. These statements are used to create the counters and arrays. This chapter describes how you can add looping to Visual Basic programs so that the programs can process several data values using looping statements. Loops also enable you to correct user errors and repeat certain program functions when the user requests a repeat. The second half of this chapter we will learn the arrays which gives you another way to reference data than by using a different variable name. Each element in the array, however, is a unique variable known by its array name and subscript.

6-1: Looping Statements

In this chapter we will discuss four types of looping statements used to build the counters:

8. For-Next statement

9. Do While- Loop statement

10. While- Wend statement

11. Do Until- Loop statement

6-1-1: For _Next- Statements

The For-Next statement repeats statements for a specified number of times. The format of this statement always begins with the For statement and ends with the Next statement as shown below:



Where:

CounterVar: A variable represents the counter.

StartVal: A number represents the starting value for the counter.

EndVal: A number represents the Ending value for the counter.

Step-size: A number represents the increasing (or decreasing) value for the counter.

Example-1

Write a VB program to find the factorial of any integer number (N!)

Hint: Factorial (N) = $N! = 1 \times 2 \times 3 \times \dots \times N$

Solution:

```
Private Sub Form_Click()
Dim N%, I%, F#
N = Val (InputBox ("Enter any number"))
F = 1
For I = 1 To N
F=F *I
Next I
Print "Factorial of"; N; "="; F
End Sub
```

Example-2

Write a VB program to find the value of S from the following series:

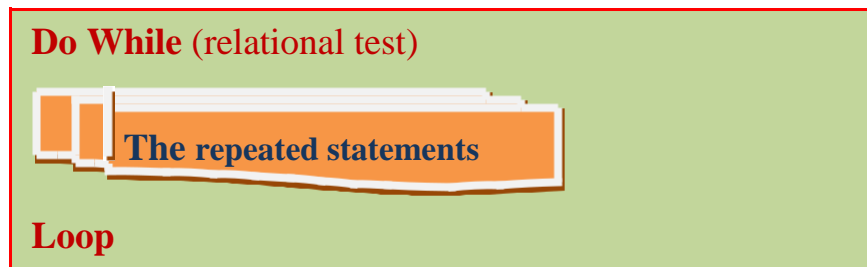
$$S = \sum_{I=1}^N \frac{1}{I} (X + N)$$

Solution:

```
Private Sub Form_Click()  
Dim X!, N%, I%, S#  
N = Val (InputBox ("Enter N"))  
X = Val (InputBox ("Enter X"))  
S = 0  
For I = 1 To N  
S=S+(1/I)*(X+N)  
Next I  
Print S  
End Sub
```

6-1-2: Do While_Loop- Statement

The Do While statement works with relational expressions just as the If statement does. Therefore, the six relational operators work as expected here. Like the If statement that ends with an End If statement, a loop will always be a multi-line statement that includes an obvious beginning and ending of the loop. Here is the format of the Do While loop:



To modify the For-Next statement to the Do While so to work as a counter we will use the following format according to the previously defined values as shown below:

```
CounterVar = StartVal
```

```
Do while (CounterVar <= EndVal)
```



The repeated statements

```
CounterVar = CounterVar + step_size
```

```
Loop
```

Example-3

Write a VB program to find S from the following series. Use the Do While_Loop- statement.

$$S = \frac{1}{a+b} + \frac{2}{(a+b)^2} + \frac{3}{(a+b)^3} + \dots + \frac{n}{(a+b)^n}$$

Find the value of S for each input values of a, b and n.

Solution:

```
Private Sub Form_Click()
```

```
Dim a!, b!, n%, I%, s#
```

```
a = Val (InputBox ("Enter the value of a"))
```

```
b = Val (InputBox ("Enter the value of b"))
```

```
n = Val (InputBox ("Enter the value of n"))
```

```

s = 0
I = 1
Do While (I <= n)
s = s + (I / ((a + b) ^ I))
I=I+1
Loop
Print a; b; n; s
End Sub

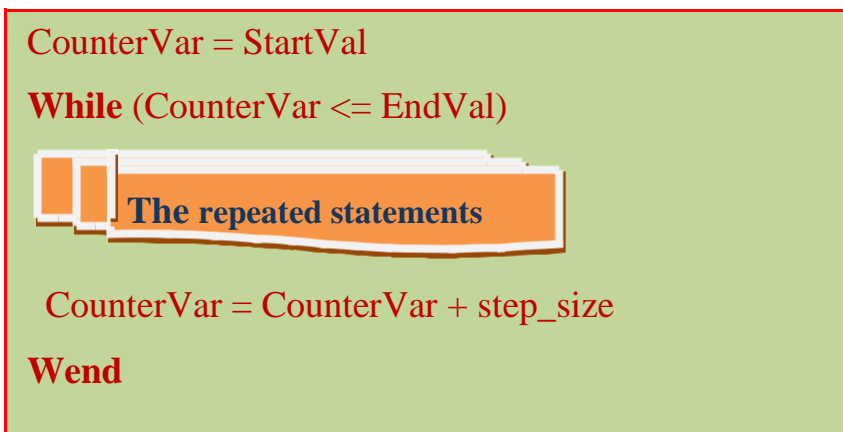
```

6-1-3: While_Wend- Statement

The While-Wend statement works with relational expressions just as the Do While statement does. The format of the While-Wend loop is:



And to make it work as a counter use the following format:



Example-4

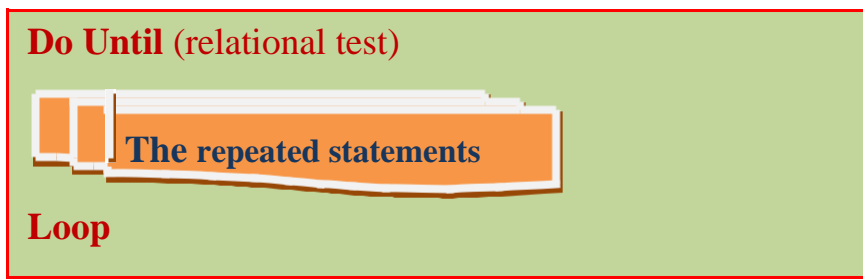
Write a VB program ask the officer to enter his name (N), age(A), and his first salary (F). If you know that this salary is increased 5% per year. Using (While_Wend- Statement, find and print the overall salaries that this officer can get it until he reach the retirement age (65) year.

Solution:

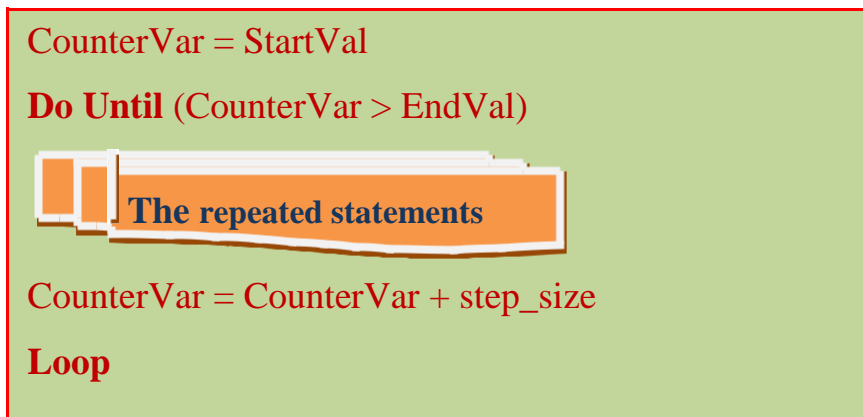
```
Private Sub Form_Click()  
Dim N$, A!, F#  
N = InputBox("enter officer name")  
A = Val (InputBox("enter officer age"))  
F = Val (InputBox("enter officer first salary"))  
While (A <= 65)  
F=F+0.05*F  
A=A+1  
Wend  
Print N; F  
End Sub
```

6-1-4: Do Until_ Loop- Statement

Whereas the Do While-Loop continues executing the body of the loop as long as the relational test is true, the Do Until-Loop executes the body of the loop as long as the relational test is false. The format of the Do Until is as shown:



And the counter format is:



Example-5

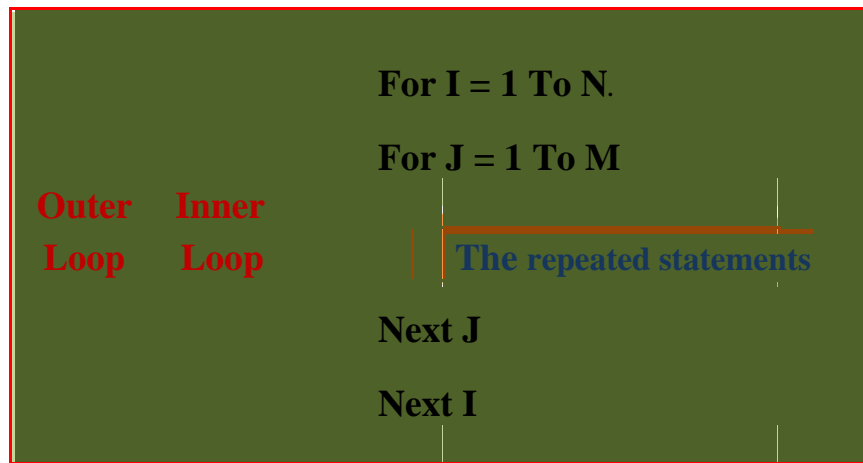
Two cars (C1 and C2), at time 0 hour the first car speed was 150 km/h and the second was 50 km/h. If the first car speed increased by a rate of 2% per hour and the second 6% per hour. At what time will the second car reach the first? Write a VB program to clear that (Use Do Until_Loop- Statement).

Solution

```
Private Sub Form_Click()  
Dim T!, C1!, C2!  
T=0:C1=150:C2=50  
Do Until (C2 >= C1)  
C1=C1+0.02*C1  
C2=C2+0.06*C2  
T=T+1  
Loop  
Print "At Time: "; T; "Hours"; "C2 will reach C1"  
Print C1  
Print C2  
End Sub
```

6-2: Nested Loops

Its so called because it contains many loop statements nested together without any intersections. As shown in figure below there are many loops,the Do While loop is called the outer loop the other loops is called the inner loops. The outer loop is closed finally and the last inner loop is closed firstly. The advantage of these loops is to build the Arrays that we will study it in the next section.



Example-6

Write a VB program to find and print the multiplication table from 1 to 10.

Solution:

```

Private Sub Form_Click()
Dim I%, J%, P%
  
```

```

For I = 1 To 10
For j = 1 To 10
P=I*J
Print P;
Next J
Print
Next I
End Sub

```

Example-7

Write a VB program to find the value of S from the following equation:

$$S = \sum_{I=1}^5 \sum_{J=1}^7 (I+J)$$

Solution:

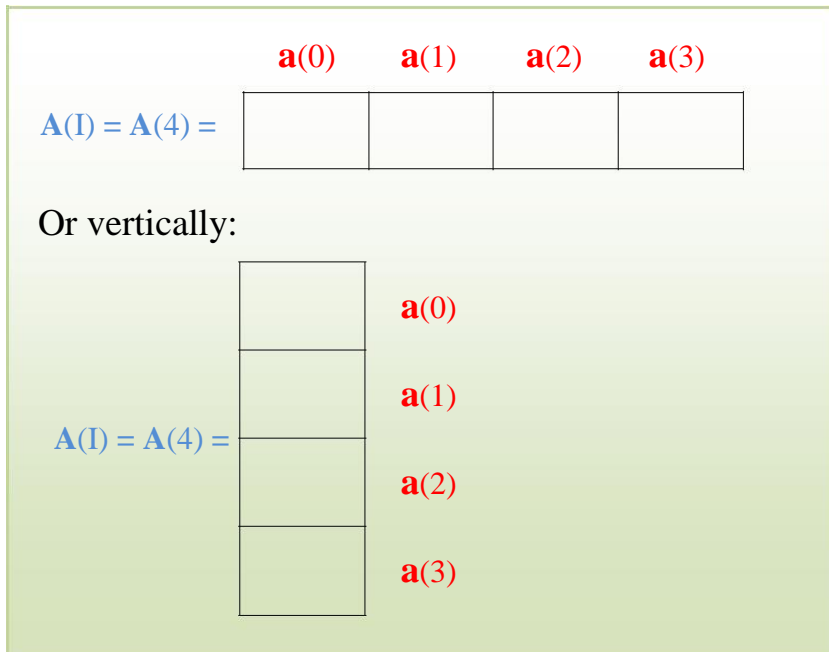
```

Private Sub Form_Click()
Dim I%, J%, M%, S%
M=0:S=0
For I = 1 To 5
For J = 1 To 7
M=M+(I+J)
Next J
S=S+M
Next I
Print "S="; S
End Sub

```

6-3: Arrays

Variable can reserve a one cell from the computer memory so we can reserve many variables that are similar in name and data type in one variable contain the same name but many locations. The resulted variable is called array and these locations represented by a single loop called subscript. We can work with individual array elements as if they were stand-alone variables by accessing their individual subscripts. The array subscript enables you to access one or more elements from the array. The big advantage that arrays provide is enabling your program to step through the entire array using a For loop's control variable as the subscript. As an example below illustrates an array named (A) contains four elements and subscript named (I). This array may be written in horizontal or vertical manner.



Ex : An array N(5) with numeric elements:

	n(0)	n (1)	n(2)	n(3)	n(4)
N(5) =	11	20	45	67	83

Ex : An array S(4) with string elements:

	s(0)	s(1)	s(2)	s(3)
S(4) =	Name	Degree	Sum	Average

6-3-1: Declaring Arrays

We usually use the **Dim**-statement to declare a standard array using the form:

Dim Array-Name (Index) **As** Data-Type

If we want define the array using any starting and ending values for index:

Dim Array-Name (Initial to Final (index)) **As** Data-Type

6-3-2: Read, Generate, Input, and Print Arrays

Reading arrays is used when the elements are known. The value of any element is entered directly.

Ex : Write a VB program to read the below array B(4).

$$B(4) = \begin{bmatrix} 32 \\ 48 \\ 67 \\ 18 \end{bmatrix}$$

Sol:

```
Private Sub Form_Click)(
Dim B(1 To 4) As integer
B(1)=32: B(2)=48: B(3)=67: B(4)=18
End Sub
```

Generating arrays can be done by the use of the statement Rnd which is used when we want to generate numbers randomly in the interval [0, 1]. We can use this statement to generate numbers in the interval [0, N] only we have to product N by this statement. Loops are needed in such case.

Ex : Write a VB program to generate the elements randomly in the interval [0, 10] for the array C(8).

Sol:

```
Private Sub Form_Click()  
Dim C(1 To 8) As integer  
For I = 1 To 8  
C(I) = 10* Rnd  
Next I  
End Sub
```

Inputting arrays is used when we do not have the values of the array elements or if we have to enter its elements in the program. Loops are needed for inputting arrays.

Ex : Write a VB program to input the elements for the array A(4).

Sol:

```
Private Sub Form_Click()  
Dim A(1 To 4) As Single  
For I = 1 To 4  
A(I) = Val(InputBox("Enter elements of A", I((  
Next I  
End Sub
```

Finally, printing arrays is used when want to view our resultant array on the form.

Ex : Write a VB program to print the array D(6) horizontally.

Sol:

```
For I = 1 To 6
Print D(I);
Next I
```

Note :

If you want to print it vertically remove the semicolon symbol (;)

Example-8

Write a VB program to enter the array elements A(6) then display the array B(6) which contain the reversed A elements.

Solution

```
Private Sub Form_Click()
Dim A(1 To 6), B(1 To 6) As Single, I%
For I = 1 To 6
A(I) = Val (InputBox("Enter (A) element", I))
Next I
For I = 1 To 6
B(I) = A(7 - I)
Next I
Print "A(I)"; Tab(8); "B(I)"
For I = 1 To 6
Print A(I); Tab(8); B(I)
Next I
End Sub
```


Example-9

Write a VB program to enter the elements of two lists. Each list contain (14) elements then display the array C(14) contain the mean between opposite numbers in the two lists.

Solution

```
Private Sub Form_Click()  
Dim A(1 To 14), B(1 To 14), C(1 To 14) As Single, I%  
  
For I = 1 To 14  
A(I) = Val(InputBox("Enter elements of A", I))  
B(I) = Val(InputBox("Enter elements of B", I))  
C(I) = (A(I) + B(I)) / 2  
Next I  
  
Print "A(I)"; Tab(8); "B(I)"; Tab(8); "C(I)"  
  
For I = 1 To 14  
Print A(I); Tab(8); B(I); Tab(8); C(I)  
Next I  
  
End Sub
```

6-3-3: Array Operations

In this section we will discuss the most used operations on arrays. These operations can be summarized in the following points:

- Finding Maximum and Minimum value.
- Sorting (ascending or descending)
- Summing elements.
- Replacing elements.

6-3-3-1: Finding Maximum and Minimum value

To find the maximum or minimum element in the array, follow these steps:

Step-1: set the first element to a variable named "Max" (or Min)

Step-2: open a loop then compare this variable with the other array elements.

Step-3: if the compared elements greater that (or smaller) than "Max" (or "Min") then set this element as the new maximum (or minimum) value.

Step-4: if the condition in step-2 is satisfied, the old maximum (or minimum) value will not change.

Step-5: print the maximum (or minimum) element which is stored in the variable Max (or Min).

Example-10

You have the array A(10), write a VB program to find and print the maximum and minimum value in this array and locate its position.

Solution

```
Max = A(1)
Min = A(1)
For I = 1 To 10
If A(I) > Max Then
Max = A(I)
L1=I
End If
If A(I) < Min Then
Min = A(I)
L2=I
End If
Next I
Print "Maximum = "; Max; "At Location"; L1
Print "Minimum = "; Min; "At Location"; L2
End Sub
```

6-3-3-2: Sorting in ascending or descending manner

The other important application used in arrays is to sort its elements in ascending or descending manner. To do this we must follow the following steps:

Step-1: open two loops (named I and J). Loop (I) begun from 1 to (N-1) and (J) from I+1 to N, where N is an integer number represents the array size.

Step-2: compare each element in the array with the other remaining elements, that is satisfied in the relation ($A(I) > A(J)$) for ascending and ($A(I) < A(J)$) for descending.

Step-3: if the condition in step-2 is satisfied then interchange these values. Not that with interchanging you must use a variable (say T) to work as a memory to avoid losing the values, as $T = A(I)$; $A(I) = A(J)$; $A(J) = T$

Step-4: if the condition in step-2 is not satisfied, the elements will stay without any change.

Step-5: print the same array (after sorting).

Example-11

Write a VB program to generate randomly the array B(20) then sort and print this array in ascending manner.

Solution

```
Dim B!(1 To 20)
Randomize
For i = 1 To 20
    B(i) = 10 * Rnd
Next i
For i = 1 To 19
    For J = i + 1 To 20
        If B(i) > B(J) Then
            T = B(i): B(i) = B(J): B(J) = T
        End If
    Next J
Next i
For i = 1 To 20
    Print B(i)
Next i
```

6-3-3-3: Summing array elements

To find the sum we must first name the variable (say Sum) as the sum of all elements. Then follow these steps: **Step-1:** set the initial value for the variable (Sum) before open the loop.

Step-2: put the cumulative sum equation ($\text{Sum} = \text{Sum} + \text{Array elements}$) for the Sum variable.

Step-3: print the variable (Sum) after the next of the loop.

Note: We can follow these steps in multiplication operation but change the variable name to Prod and set its initial value to (1) then convert the plus sign in step-3 (+) to prod sign (*).

Example-12

You have the array A(10), write a VB program to find and print the sum and product of all array elements.

Solution

```
Sum = 0: Prod=1
For I = 1 To 10
Sum = Sum + A(I)
Prod = Prod * A(I)
Next I
Print "Sum of A elements="; Sum
Print "Prod of A elements="; Prod
```

6-3-3-4: Replacing elements

This operation is used if desired to replace an array elements with another elements in the same array. To do this we must follow these steps:

Step-1: first, we must limit the replacing elements by the use of loops. Open loop (I) to limit the replacing elements.

Step-2: before replacement we must find a relation between the interchanging elements (usually addition by a constant number) then use the previously defined variable (T) to work as a memory, as $T = A(I)$: $A(I) = A(I+const)$: $A(I+const) = T$ **Step-3:** print the resultant array after replacing its elements.

Example-13

Write a VB program to generate randomly the element of array S(30), then replace the first five elements with the last five and print the two arrays.

Solution

```
Dim S%(1 To 30), I%
```

```
Randomize
```

```
For I = 1 To 30
```

```
    S(I) = 10 * Rnd
```

```
Print S(I);
```

```
Next I
For I = 1 To 5
T = S(I): S(I) = S(I + 25): S(I + 25) = T
Next I
Print
For I = 1 To 30
Print S(I);
Next I
```

6-3-4: Control Array

A control array is nothing more than a list of controls, just as a variable array is a list of variables. The advantage to using a control array is the same as for using a variable array: You can step through several variables using a loop instead of having to name each individual control.

Visual Basic supports one technique for control arrays that you'll find yourself using a lot, even though collections are always available to you. When you copy a control and paste that control back onto the form, Visual Basic displays the message box shown below:



You might wonder why you'd ever copy and paste a control, but if you need to place several commands buttons or labels that all have the same format—perhaps the same font size and caption alignment—it's a helpful technique. You just create one control, set all its properties. As soon as you paste the copied control, Visual Basic displays the message box shown previously. If you answer Yes, Visual Basic automatically creates a control array with a name that matches the first control.

For example, if the first control is a command button named `Command1`, the array is named `Command1`, and the elements begin at `Command1(0)` and increment as long as you keep pasting the control. Your code then can step through all the control array elements from `Command1(0)` through `Command1(n)`, where `n` is the total number of `Command1` controls on the form, and set properties for them.

Example-14

By using control array, write a VB program to design the calculator shown below with their objects property. Use the control array to define the command numbers and another array to define the four basic operations.

Solution

```
Dim a As Double
```

```
Dim b As Double
```

```
Dim Operator As String
```

```
Dim result As Double
```

```
Private Sub cmdclear_Click()
```

```
lblview.Caption = ""
```

```
End Sub
```

```
Private Sub cmdnumbers_Click(Index As Integer)
```

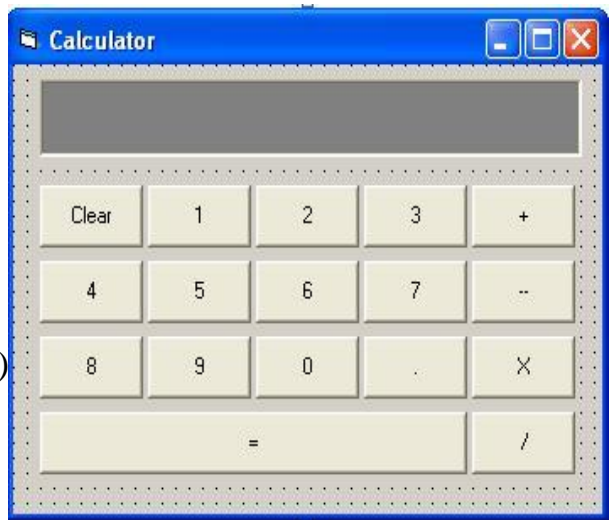
```
lblview.Caption = lblview.Caption + cmdnumbers(Index).Caption
```

```
End Sub
```

```
Private Sub cmdmMathOp_Click(Index As Integer)
```

```
a = Val(lblview)
```

```
lblview.Caption = ""
```



Select Case Index

Case 0

Operator = "+"

Case 1

Operator = "-"

Case 2

Operator = "*"

Case 3

Operator = "/"

End Select

End Sub

Private Sub cmdequal_Click()

b = Val(lblview.Caption)

Select Case Operator

Case "+"

result = a + b

Case "-"

result = a - b

Case "*"

result = a * b

Case "/"

result = a / b

End Select

lblview.Caption = result

End Sub

Label Box: lblview

Command Box: cmdclear

Command Box: cmdresult

Command Array(10): cmdnumbers

Command Array(4): cmdMathOp

6.4 Problems

- Write a VB program to enter 20 degrees then calculate and print the percentage rate for the degrees greater than 60% but less than 70%.
- Write a VB program to enter 25 numbers then calculate and print their sum if one of them was negative and their product if one of them was even.
- Write a VB program to enter 60 numbers then compute and print:
 - 1- The sum of all integers between 2 and 60 those are divisible by 2.
 - 2- Sum of odd numbers.
 - 3- Number of even numbers.

A list contains the degrees of 80 students in computer course. Write a VB program to find the succeeding rate after entering the student's degrees.

Hint: Succeeding rate = (number of success students / total number) × 100 %

- Write a VB program to find the value of X where:

$$X = A^3 * B^2 / A^3 + B^2$$

For 10 entered values of A and B, use functions ("cube" and "square") to define A^3 and B^2 respectively. Display the results on a message box.

Write a VB program to enter a name and age for 100 persons then find and print:

- 1- The ages between 6 and 12 years, then display the message "he (she) is in the primary school" on another message if the age lies in the mentioned range.
- 2- Display the last three letters of the first name, if the name consists of three letters or more.

- Write a VB program to find and print the value of m from the following series:

$$= e^{-1} \cdot e^{-2} \cdot e^{-3} \dots e^{-18}$$

- Write a VB program to find and print the value of m from the following equation:

$$a = \sum_{i=1}^m x^i y^{-i}$$

Where $x = 0.7$, $y = 1.5$, and $m = 12$.

- 3) Write a VB program to find and print $\tan^{-1}(x)$ by the use of the following series:

$$\tan^{-1}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \dots + \frac{x^{99}}{99}$$

Input x then find and print $\tan^{-1}(x)$ from the above series.

- 3) Write a VB program to find $e^{-1}(x)$ using the following series up to 100 terms:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$$

Input x then find and print e^x from the above series.

11. Write a VB program to find and print the following series:

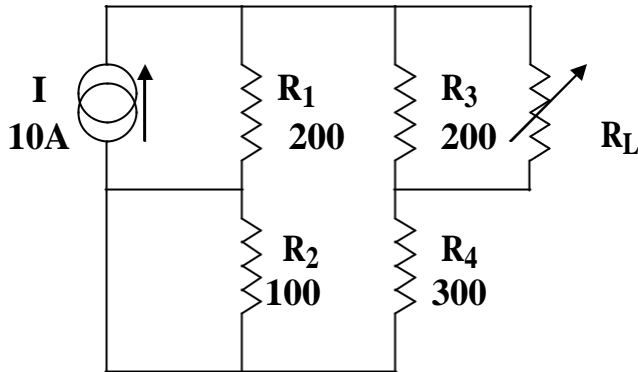
$$series(x) = x + \frac{x^3}{6} + \frac{x^5}{120} + \frac{x^7}{5040} + \dots + \frac{x^{21}}{5.1 \text{ E}19}$$

12. Write a VB program to find and print P from the following equation where r= 1, 2, 3, 4, 5, 6 and n=6. Implement the function fac(x) to find the factorial of any integer number.

$$p = \frac{n!}{(n-r)!r!}$$

In the year 1987 the Iraq census was 17, 000, 000 and the Egypt census was 50, 000, 000, if you knew that the population increment rate for Iraq 4% per year and population increment rate for Egypt was 3% per year. In what year the Iraqi census can exceed the Egypt census? Write a VB program to solve this problem.

4. Write a VB program to find the current in any branch and the voltage for any resistance in the circuit below if you knew that R_L changes from 100Ω to 1000Ω in steps of 100Ω . Print the current and voltage in branches for each value of R_L



10. Write a VB program to enter and print the two arrays $M(30)$ and $N(30)$, then find and print the value of P from the following equation:

$$9. = 2\lambda \sum_{i=1}^{30} M(i) * N(i)$$

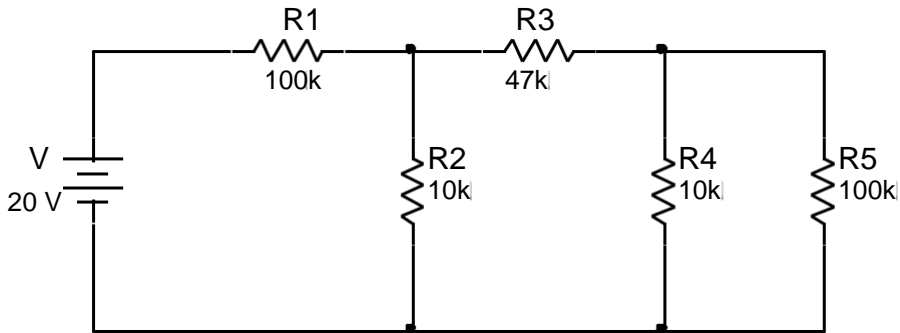
where $\lambda = 2.15 * 10^{-22}$

6. Write a VB program to enter 120 degrees for 60 students in two objects (physics and mathematics). Store these degrees in two one-dimensional arrays (P and M) then find and print the following:

6. The mean between two opposite degrees.

7. The succeeding rate in physics

9. For figure shown below write a VB program to store the 5 values for the 5 resistors in array $R(5)$ then calculate the current flows in each resistor and store it in another array $I(5)$. Find also the power dissipated in each resistor and display it on a third array $P(5)$. Print all the resulted arrays.



7. Write a VB program to find and print the value of (S) from the following series:

$$S = \sum_{i=0}^{m-1} a_i \cdot 2^i \times \sum_{j=0}^{n-1} b_j \cdot 2^j$$

Enter any undefined variables.

8. Write a VB program to enter and print horizontally the array A (20), then find and print:

1- The first three maximum numbers.

2- The last two minimum numbers.

9. N charges placed on a plate. Write a VB program to enter the value for the ith charge in an array (Q_i) and the distance from the center to the ith charge in another array (r_i) then find and print the electrical potential (E) in the center of this plate according to the following equation:

$$E = \frac{1}{4\pi\epsilon_0} \sum_i \frac{Q_i}{r_i}, \text{ where } \epsilon_0 = 8.85 \times 10^{-12}$$

10. The approximation format to find the square root for number (R) is:

$$x(i+1)=0.5*(x(i)+R/x(i))$$

Where $x(0) = R$ and $x(1), x(2) \dots$, etc are the approximated values for the square root of R. Write a VB program to find and print approximate value for the entered number (R). The program will stopped when the difference between $x(i)$ and $x(i+1)$ less than (0.001).

22. Write a VB program to store randomly generated elements in array X_i then find the value of Z from the following equations:

$$Z = \frac{1}{n} \sum_{i=1}^n (X_i - D)^4 \quad \text{Where} \quad D = \frac{1}{n} \sum_{i=1}^n X_i$$

23. Design a VB project contains ten Check Boxes as a control array and a single List Box. When you click any of Check Boxes their names will be displayed on the list box

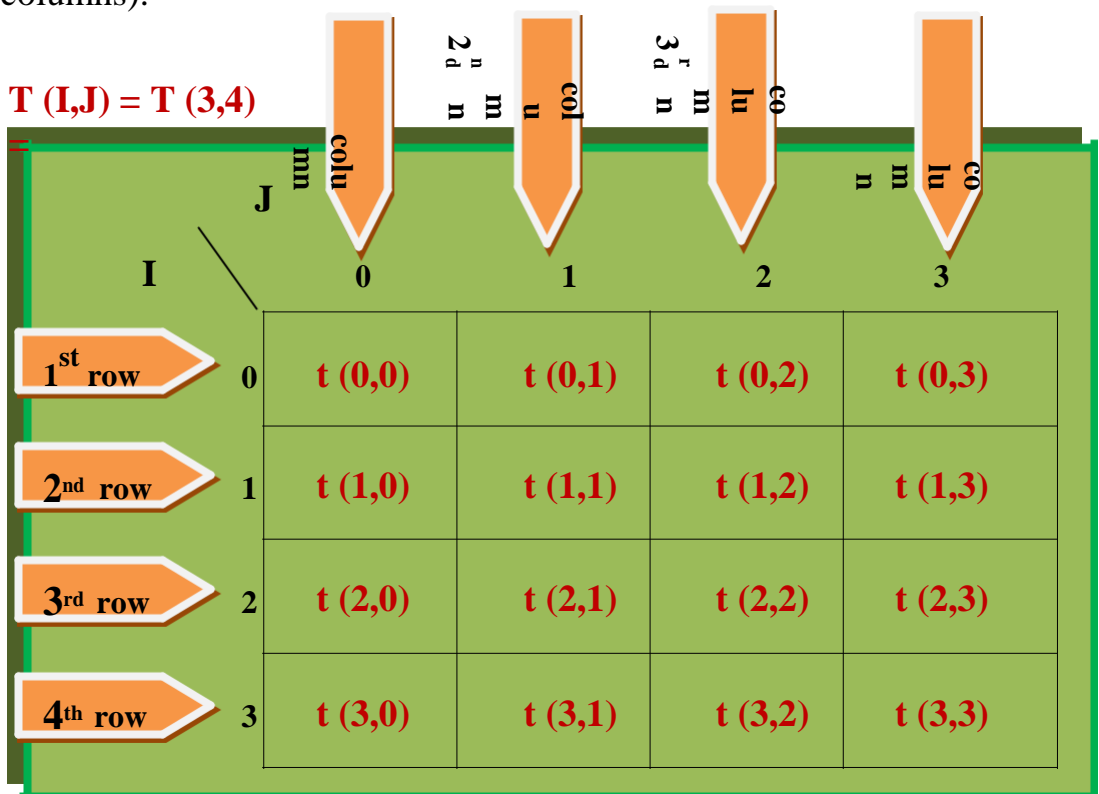
24. Design a VB project contains 26 commands array represent the English alphabets. The project also contains a text box and two command boxes "Erase" and "Space". Add a string of alphabets by the use of commands array through the text box. If you click the command "Space" a space will added into the string and if you click the command "Erase" the string in the text box will be erased.

CHAPTER SEVEN

MATRICES

Introduction

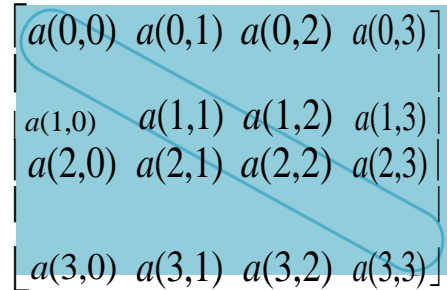
The Matrix is another way to format data as rows and columns if they have the same type of data. So the matrix is a two dimensional array contain two subscripts I and J represents the row and column indices and has a name as a capital-letter. Example below illustrates a two dimensional array named (T) contains (12) elements (three rows and four columns) and two subscripts (I for rows and J for columns).



7-1: Square Matrix

When number of rows is equal to number of columns (say N) the matrix is called "Square Matrix". This type of matrices has the following properties:

1- Main diagonal: The group of the elements that lies between the first element in the first row and the last element in the last row. As shown in the shaded area:



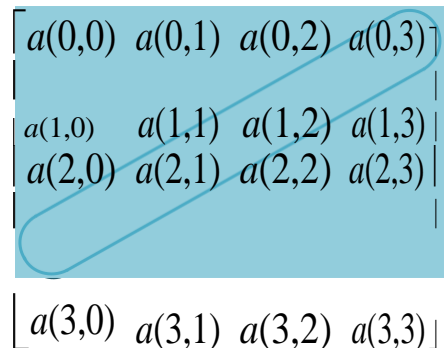
$a(0,0)$	$a(0,1)$	$a(0,2)$	$a(0,3)$
$a(1,0)$	$a(1,1)$	$a(1,2)$	$a(1,3)$
$a(2,0)$	$a(2,1)$	$a(2,2)$	$a(2,3)$
$a(3,0)$	$a(3,1)$	$a(3,2)$	$a(3,3)$

So: $\mathbf{I} = \mathbf{J}$

where:

\mathbf{I} is the row index and \mathbf{J} is the column index.

2- Secondary diagonal: The group of the elements that lies between the last element in the first row and the first element in the last row. As shown in the shaded area:



$a(0,0)$	$a(0,1)$	$a(0,2)$	$a(0,3)$
$a(1,0)$	$a(1,1)$	$a(1,2)$	$a(1,3)$
$a(2,0)$	$a(2,1)$	$a(2,2)$	$a(2,3)$
$a(3,0)$	$a(3,1)$	$a(3,2)$	$a(3,3)$

So: $\mathbf{I} = (\mathbf{N}-1) - \mathbf{J}$, if the indices (\mathbf{I} and \mathbf{J}) started from 0.

and: $\mathbf{I} = (\mathbf{N}+1) - \mathbf{J}$, if the indices (\mathbf{I} and \mathbf{J}) started from 1.

3- Upper triangle: The group of the elements that lies above the main diagonal as shown in the shaded area:

$$\begin{bmatrix} a(0,0) & a(0,1) & a(0,2) & a(0,3) \\ a(1,0) & a(1,1) & a(1,2) & a(1,3) \\ a(2,0) & a(2,1) & a(2,2) & a(2,3) \\ a(3,0) & a(3,1) & a(3,2) & a(3,3) \end{bmatrix}$$

So: $\mathbf{I} < \mathbf{J}$

4- Lower triangle: The group of the elements that lies below the main diagonal as shown in the shaded area:

$$\begin{bmatrix} a(0,0) & a(0,1) & a(0,2) & a(0,3) \\ a(1,0) & a(1,1) & a(1,2) & a(1,3) \\ a(2,0) & a(2,1) & a(2,2) & a(2,3) \\ a(3,0) & a(3,1) & a(3,2) & a(3,3) \end{bmatrix}$$

So: $\mathbf{I} > \mathbf{J}$

Ex: Matrix $M(3,3)$ with numeric elements:

$$M(3,3) = \begin{bmatrix} \mathbf{16} & \mathbf{48} & \mathbf{52} \\ \mathbf{6} & \mathbf{12} & \mathbf{3} \\ \mathbf{2} & \mathbf{10} & \mathbf{7} \end{bmatrix}$$

Ex : Matrix $S(4,4)$ with literal elements:

$$S(4,4) = \begin{bmatrix} \text{EM} & \text{AM} & \text{FG} & \text{HS} \\ \text{ENG} & \text{TL} & \text{KD} & \text{UT} \\ \text{AD} & \text{WXY} & \text{QP} & \text{WW} \\ \text{LDS} & \text{EIP} & \text{MD} & \text{ZG} \end{bmatrix}$$

7-2: Declaring Matrices

As in arrays we will use the Dim statement to declare a standard matrix having two dimensions (note that in Visual Basic, matrices can have up to 60 dimensions):

If this matrix started at index 0 the declaring format is:

Dim Array-Name (Row Index , Column Index) **As** Data-Type

Or if we want it start at any starting and ending values of indices we use this format:

Dim Array-Name (**s to e** (for **I**), **s to e** (for **J**)) **As** Data-Type

Where **s** and **e** are the starting and ending values for rows and columns indices (**I** and **J**) respectively.

7-3: Read, Generate, Input, and Print Matrices

We read the matrix when its elements are known (or given) and their values are constants during the execution of the program, so the value of any element is entered directly.

Ex: Write a VB program to read the matrix B(3,3).

$$B(3,3) = \begin{bmatrix} 48 & 32 & 29 \\ 67 & 87 & 23 \\ 18 & 60 & 44 \end{bmatrix}$$

Sol:

```
Private Sub Form_Click()  
  
Dim B(1 To 3, 1 To 3) As Integer  
B(1,1)=48: B(1,2)=32: B(1,3)=29  
B(2,1)=67: B(2,2)=87: B(2,3)=23  
B(3,1)=18: B(3,2)=60: B(3,3)=44  
End Sub
```

If the matrix was unknown but we want to generate its elements randomly use the statement **Rnd** which is used when we want to

generate numbers randomly in the interval $[0, 1]$. We can use this statement to generate numbers in the interval $[0, N]$ only we have to product N by this statement. Loops are needed in such case.

Ex: Write a VB program to generate random elements in the interval $[0, 10]$ for the matrix $C(8,8)$.

Sol:

```
Private Sub Form_Click()  
Dim C(1 To 8, 1 To 8) As integer  
For I = 1 To 8  
For J = 1 To 8  
C(I, J) = 10* Rnd  
Next J  
Next I  
End Sub
```

Inputting matrices is already used when we do not have the values of the matrix elements or its elements may be changed (not constant) in the program. Loops are needed for such case.

Ex : Write a VB program to input the elements for the two dimensional array A(4,4).

Sol:

```
Private Sub Form_Click()  
Dim A(1 To 4, 1 To 4) As integer  
  
For I = 1 To 4  
For J = 1 To 4  
A(I, J) = Val(InputBox("Enter elements of A", I((  
Next J  
Next I  
  
End Sub
```

Finally, printing matrices used when want to view our resultant array on the form. Two loops are used for printing.

Ex : If you have the matrix D(6,6), how can you print it? Write a VB program to clear that.

Sol:

```
For I = 1 To 6  
For J = 1 To 6  
Print D(I, J);  
Next J : Print : Next I
```


Example-1:

Write a VB program to enter the matrix A(4,4) then display:

12. The main diagonal elements.
13. The secondary diagonal elements.
14. The upper triangle elements.
15. The lower triangle elements.

Solution

```
Private Sub Form_Click()  
Dim A%(1 To 4, 1 To 4), I%, J%  
For I = 1 To 4  
For J = 1 To 4  
A(I, J) = Val(InputBox("Enter values of A elements", CStr(I) + CStr(J)))  
Print A(I, J);  
Next: Print: Next  
Print "The main diagonal elements are:"  
For I = 1 To 4  
For J = 1 To 4  
If I = J Then  
Print A(I, J)  
End If  
Next: Next
```

Print "The secondary diagonal elements are:"

For I = 1 To 4

For J = 1 To 4

If I = 5 - J Then

Print A(I, J)

End If

Next: Next

Print "The upper triangle elements are:"

For I = 1 To 4

For J = 1 To 4

If I < J Then

Print A(I, J)

End If

Next: Next

Print "The lower triangle elements are:"

For I = 1 To 4

For J = 1 To 4

If I > J Then

Print A(I, J)

End If

Next: Next

End Sub

Example-2:

Write a VB program to enter the elements of matrix F(3,5) then convert it to the array X(15) which has the same row-by-row elements.

Solution:

```
Private Sub Form_Click()  
Dim F%(1 To 3, 1 To 5), I%, J%, K%  
Dim X%(1 To 15)  
K = 0  
For I = 1 To 3  
For J = 1 To 5  
F(I, J) = Val(InputBox("Enter values of F elements", CStr(I) + CStr(J)))  
Print F(I, J);  
K=K+1  
X(K) = F(I, J)  
Next: Print: Next  
Print "-----"  
For K = 1 To 15  
Print X(K)  
Next  
End Sub
```

7-4: Matrix Operations

In the following few papers we will discuss the most used matrix operations. These operations are:

8. Finding Maximum and Minimum value.
9. Summation and production (Rows, Columns, All).
10. Replacement between matrix elements.

7-4-1: Finding Maximum and Minimum value

To find the maximum or minimum element in the matrix, follow these steps:

Step-1: set the first element to a variable named "Max" (for maximum) and "Min" (for minimum).

Step-2: open the two loops then compare this variable with the other elements.

Step-3: if the compared elements greater than "Max" (for maximum) or less than "Min" (for minimum), set this element as the new maximum (or minimum) value.

Step-4: if the condition in step-3 is not satisfied, the old maximum (or minimum) value will not change.

Step-5: print the maximum (or minimum) element which is stored in the variable Max (or Min).

Example-3

You have the matrix A(10,20), write a VB program to find and print the maximum and minimum value in this array and locate its position.

Solution

```
Max = A(1,1)
```

```
Min = A(1,1)
```

```
For I = 1 To 10
```

```
For J = 1 To 20
```

```
If A(I,J) > Max Then
```

```
Max = A(I,J)
```

```
L1=I:L2=J
```

```
End If
```

```
If A(I,J) < Min Then
```

```
Min = A(I,J)
```

```
L3=I,L4=J
```

```
End If
```

```
Next : Next
```

```
Print "Maximum = "; Max; "At Location I="; L1;"and Location J=";L2
```

```
Print "Maximum = "; Min; "At Location I="; L3;"and Location J=";L4
```

```
End Sub
```

7-4-2: Summation and production (Rows, Columns, All)

To find the sum we must first name the variables as shown: Sum1 as sum of rows elements, Sum2 as sum of columns elements, and Sum3 as sum of all elements. Then follow these steps:

Step-1: set the initial value for these variables to zero in different places as follow: Sum1=0 between (I) and (J) loops. Sum2=0 between (J) and (I) loops. Sum3=0 before (I) and (J) loops.

Step-2: put the cumulative sum equation (Sum=Sum + matrix elements) for Sum1, Sum2, and Sum3.

Step-3: print the variables but in different places as follow: print Sum1 between the next of (J) and (I) loops. print Sum2 between the next of (I) and (J) loops. print Sum3 after the next of the two loops.

Note: We can follow these steps in multiplication operation but change the names to Prod1, Prod2, and Prod3 and set them to (1). Convert the plus sign in step-3 (+) to a prod sign (*).

Example-4

You have the matrix $W(4,4)$, write a VB program to find and print the sum of rows, columns and the sum of all matrix elements.

Solution

```
For I = 1 To 4
Sum1 = 0
For J = 1 To 4
Sum1 = Sum1 + W(I, J)
Next J
Print "Sum of rows="; Sum1
Next I
For J = 1 To 4
Sum2 = 0
For I = 1 To 4
Sum2 = Sum2 + W(I, J)
Next I
Print "Sum of columns="; Sum2;
Next J
Sum3 = 0
For I = 1 To 4
For J = 1 To 4
Sum3 = Sum3 + W(I, J)
Next J : Next I
Print "Sum of array elements="; Sum3
End Sub
```

7-4-3: Replacement between matrix elements

This operation is used if desired to replace a row (or column) with another row (or column) or replacement between matrix halves or quarters. To do this we must follow these steps:

Step-1: first, we must limit the replacing area by using loops. Open loop (I) to limit the replacing rows and open loop (J) to limit the replacing columns (Fix one of these loops if the replacing area was one row or one column).

Step-2: before replacement we must find a relation between the interchanged areas (usually addition with constant number) then use the temporary storage variable (say T) as $T = A(I,J)$: $A(I,J) = A(I+const, J+const)$: $A(I+const, J+const) = T$

Step-3: print the resultant array after replacing its elements.

Example-5

You have the matrix C(4,4), write a VB program to replace the upper half elements with the lower half.

Solution

```
For I = 1 To 2
```

```
For J = 1 To 4
```

```
T = A(I, J): A(I, J) = A(I + 2, J): A(I + 2, J) = T
```

```
Next J : Next I
```


Example-6

Write a VB program to input and print the matrix A(4, 4) then find and print the following:

11. Maximum value and its position.
12. Sum of rows elements.
13. Sum of columns elements.
14. Sum of all elements.
15. Production of all elements.

Solution

```
Private Sub Form_Click()
Dim A(1 To 4,1 To 4) As Single, I%, J%
Dim Max!, Sum1!, Sum2!, Sum3!, L1%, L2%, Prod!
For I = 1 To 4
For J = 1 To 4
A(I, J) = Val(InputBox("Enter A(4,4) elements", I))
Print A(I, J);
Next J: Print : Next
Next I

Max = A(1, 1)
For I = 1 To 4
For J = 1 To 4
If A(I, J) > Max Then
Max = A(I, J)
```

```
L1=I:L2=J
End If
Next J
Next I
Print "Maximum="; Max, "Location="; L1; L2

For I = 1 To 4
Sum1 = 0
For J = 1 To 4
Sum1 = Sum1 + A(I, J)
Next J
Print "Sum of rows="; Sum1(I)
Next I
For J = 1 To 4
Sum2 = 0
For I = 1 To 4
Sum2 = Sum2 + A(I, J)
Next I
Print "Sum of columns="; Sum2(J);
Next J
Sum3 = 0
For I = 1 To 4
For J = 1 To 4
Sum3 = Sum3 + A(I, J)
Next J
Next I
Print "Sum of array elements="; Sum3
End Sub
```

```
Prod = 1
For I = 1 To 4
For J = 1 To 4
Prod = Prod * A(I, J)
Next J
Next I
Print "Prod of array elements="; Prod
End Sub
```

Example-7

Write a VB program to generate a random elements for the matrix A(8, 8) then do the following:

8. Replace the first quarter with the second.
9. Replace the left half with the right.
10. Replace the second row with the last.
11. Replace the third column with the fifth.

Solution:

```
Private Sub Form_Click()
Dim A(1 To 8, 1 To 8) As Integer, I%, J%

For I = 1 To 8
For J = 1 To 8
A(I, J) = 10*Rnd
```

Next

Next

For I = 1 To 4

For J = 1 To 4

T = A(I, J): A(I, J) = A(I, J+4): A(I, J+4) = T

Next J

Next I

For I = 1 To 8

For J = 1 To 4

T = A(I, J): A(I, J) = A(I, J+4): A(I, J+4) = T

Next J

Next I

For J = 1 To 8

T = A(2, J): A(2, J) = A(8, J): A(8, J) = T

Next J

For I = 1 To 8

T = A(I, 3): A(I, 3) = A(I, 5): A(I, 5) = T

Next J

7.5 Problems

- Write a VB program to read two matrices (A and B) below then find the matrix (C and D) from the following relations:

12. $C=A+B$

13. $D=A-B$

$$B = \begin{bmatrix} 6 & 0 \\ -2 & -4 \end{bmatrix}, \quad A = \begin{bmatrix} 3 & 4 \\ -1 & 5 \end{bmatrix}$$

- ☐ Write a VB program to enter the matrix $X(4, 5)$ then print the matrix $Y(5,4)$ which is the transpose of matrix X .
- ☐ Write a VB program to enter the matrix $C(4, 3)$ then find and print the maximum and minimum value for this matrix.
- 4.** Write a VB program to enter matrix $M(2, 2)$ then find and print the determinant of this matrix.
- 5.** Write a VB program to convert the matrix $A(3, 2)$ to the array $X(3)$. Each element in the array (X) is the sum of the opposite columns elements in matrix (A). Enter the elements of matrix (A) then print the resulted array (X).
- ☐ Write a VB program to enter the matrix $A(4, 4)$ then find and print the following:
 - ☐ Sum of the main diagonal elements.
 - ☐ Sum of the second row elements.
 - ☐ Multiplicand of the first column elements.

- ☐ Sum of the upper triangle elements.
- ☐ Sum of the lower triangle elements
- ☐ Sum of all elements.

7. Four workers are works 5 days per week, the working hours are listed in the table below. Write a VB program to store the information in this table matrix $X(4, 5)$ then find and print the following:

- ☐ Number of overall working hours.
- ☐ Number of working hours for each worker.
- ☐ The worker whose get maximum number of working hours.
- ☐ The day which accomplished a maximum number of working hours.

Worker	Sun	Mon	Tue	Wed	Thu
3	8.1	8	1.1	8.1	8
1	1	8	31	8.1	8
1	8.1	1	1	1	8
1	1.1	8	8	1.1	-

8. Write a VB program to enter the array $E(4, 4)$ then find and print the following:

- ☐ Maximum and minimum element in each row and their places
- ☐ Maximum and minimum element in each column and their places
- ☐ Maximum and minimum element in the array and their place

9. Write a VB program to enter the matrix $Q(8, 8)$ then do and print the following:

- Convert it to the array $R(64)$ then sort it in ascending manner.
- Replace the second quarter elements with the fourth quarter.
- Replace the first quarter elements with the third quarter.
- Replace the fourth column with the second row.

4) Write a VB program to enter the matrix $Z(6, 6)$ then do and print the following:

4) Add number (4) to the first row.

5) Subtract (2) from the third column.

6) Multiply the fourth row by (9).

7) Divide the sixth column over (8).

4) Design a VB project contains two command buttons named "enter matrix elements" and "change". Asking you to enter the elements of matrix $A(6, 6)$ when you click the first command and when you click the second command the 6th column will be changed by the triple value of the 1st column elements and the 3rd row will be replaced double value of the 4th row.

CHAPTER EIGHT

DRAWING IN VISUAL BASIC

Introduction

Visual basic has an advanced methods for drawing shapes like rectangles, circles, squares,... etc or drawing a points or functions like sine, cosine, Ln,...etc. In this chapter we will learn how to draw these geometric shapes in addition to learn how to draw free curves and shapes. Before we learn how to draw we must know some basics.

8-1: Drawing Basics

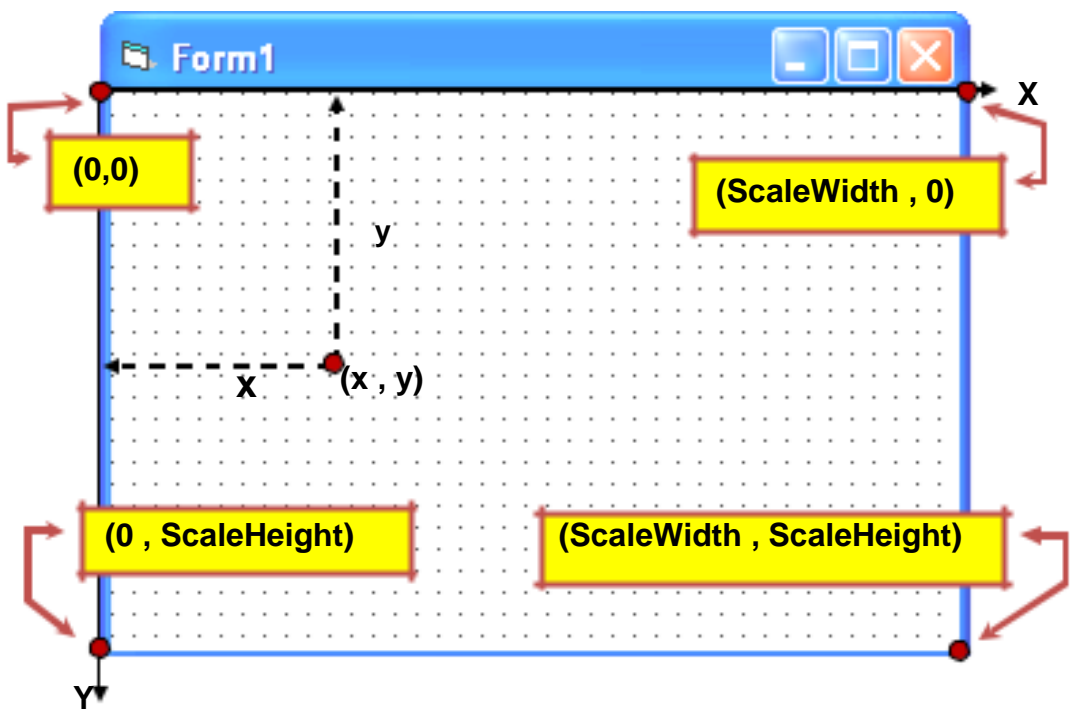
The first important thing is the Scale that we learn it in chapter-1 by using the ScaleMode property that gives us eight standard units. We can use our lonely scale by the use of **Scale**-statement. The format of this statement is:

Scale (X1,Y1)-(X2,Y2)

Where X1, Y1 are the upper left point coordinates and X2, Y2 are the lower right point coordinates. So this statement will divide the

form to number of squares its number equal to $X2-X1$ in X-Axis and $Y2-Y1$ in the Y-Axis.

Unlike the mathematic axis visual basic axis are as shown in Figure (8-1) the upper left corner is the original $(0,0)$ point and the lower right point is the end point and it is equal to $(ScaledWidth, SaledHeight)$.



Figure(8-1): Visual Basic form coordinates

In visual basic we can also control the width of the drawing lines using **DrawWidth** property as shown in the following format:

DrawWidth = Integer Number

8-2: Shapes Drawing

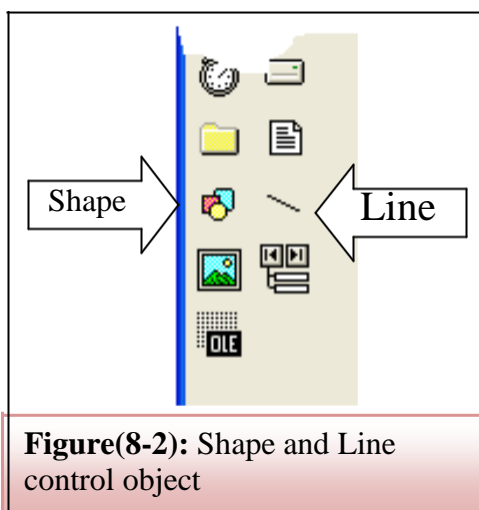
We can use visual basic to draw the standard shapes like lines, squares, rectangles, circles, ... etc. So we will learn how to draw these shapes using visual basic programming language and visual basic toolbox.

8-2-1: Using the shape box

The simplest way to draw prepared shapes is to use the shape box available in the toolbox shown in Figure (8-2). Select this object then place it on the form according to the wanted size then vary the properties (Shape and Fill style) to get the shape and its filling type from Table-1 and Table-2 respectively:

Index	Shape
0	Rectangle
1	Square
2	Oval
3	Circle
4	Rounded Rectangle
5	Rounded Square

Table-1: shape property selection domain



Index	Fill Style
0	Solid
1	Transparent
2	Horizontal Line
3	Vertical Line
4	Upward Diagonal
5	Downward Diagonal
6	Cross
7	Diagonal Cross

Table-2: Fill Style property selection domain

Example-1:

Design a project contain shape box and command button named "Draw shape". When we click this command the following shape will appear with its property: A rounded square shape of width and height=1000 twip with crossed lines filling with red colored.

Solution:

```
Private Sub cmdDraw_Click()
```

```
With Shape1
```

```
    .Shape = 5
```

```
    .Width = 1000
```

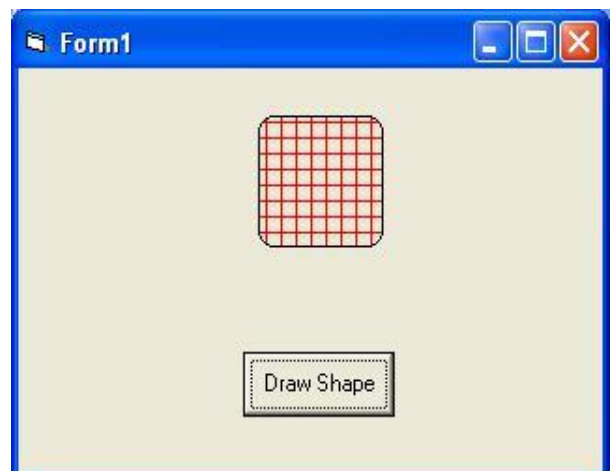
```
    .Height = 1000
```

```
    .FillStyle = 6
```

```
    .FillColor = vbRed
```

```
End With
```

```
End Sub
```



8-2-2: Lines Drawing

Lines can be drawn using either the line box available in the toolbox (back to Figure (8-2)) or using the code window. If the drawing lines are not changed during the program execution you can use the line box to draw these fixed lines. When the drawing lines are varied during the execution we must use the following procedure:

Line (x1 , y1)-(x2 , y2)

Where x_1 , y_1 are the start points and x_2 , y_2 are the end point for this line.

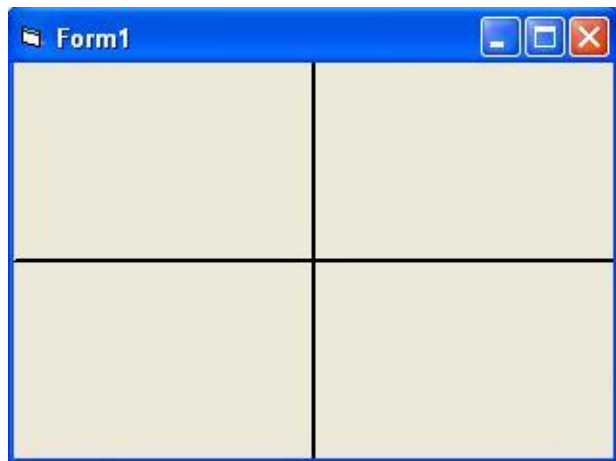
Example-2:

Write a VB program to draw two lines represents the X and Y axis on the form. The two lines must divide the form into two equal parts horizontally and vertically and also must meted in the center of this form.

Solution:

```
Dim Xst As Single 'X start  
Dim Yst As Single 'Y start  
Dim FW As Single 'Form Width  
Dim FH As Single 'Form Height
```

```
Private Sub Form_Click()  
DrawWidth = 2  
FH = Form1.ScaleHeight  
FW = Form1.ScaleWidth  
Xst = FW / 2  
Yst = FH / 2  
Line (0, Yst)-(FW, Yst)  
Line (Xst, 0)-(Xst, FH)  
End Sub
```



8-2-3: Circles, Ellipses, Sectors, and Arcs Drawing

As we learn previously, we can draw circles using the shape box by changing the shape property index to 3 (according to table-1), but these circles are fixed. In this section we will learn how to draw circles and their derivatives (ellipses, Sectors, and Arcs) using the code window. The following procedures will help use to draw these shapes:

1. To draw a circle centered at x , y and radius R. C represent the shape color as a vb constant:

Circle (x , y), R ,C

2. To draw an ellipse its x-axis radius is R1 and y-axis radius R2.

Circle (x , y) , R1 , C , , , R2

3. To draw a sector started at point P1 and end at point P2. Note that the negative sign (-) is not an operation but it is used to distinguish between the sectors and arcs listed below.:

Circle (x , y) , R, C , - P1 , - P2

4. To draw an arc started at point P1 and its end at point P2:

Circle (x , y) , R , C , P1 , P2

Example-3:

Write a VB program to draw a circle, ellipse, sector, and arc with different colors and different places. Use a form scale of 10.

Solution:

```
Private Sub Form_Load()
```

```
Scale (0, 0)-(10, 10)
```

```
DrawWidth = 2 End Sub
```

```
Private Sub Form_Click()
```

```
' Draw a circle
```

```
Circle (2, 2), 1, vbHighlight
```

```
' Draw an ellipse
```

```
Circle (4, 4), 1, vbBlue , , , 2
```

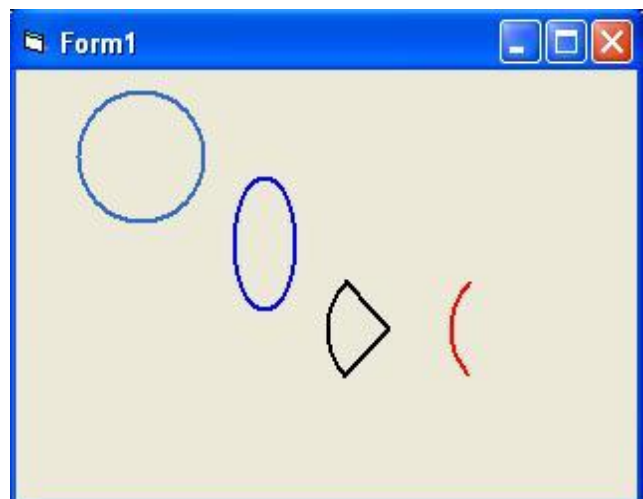
```
' Draw a sector
```

```
Circle (6, 6), 1, vbmagenta , -3 * Atn(1), -5 * Atn(1)
```

```
' Draw an arc
```

```
Circle (8, 6), 1, vbRed , 3 * Atn(1), 5 * Atn(1)
```

```
End Sub
```



8-3: Texts Drawing

The classic method to display texts is to use the Print-statement but this method is force us to print the texts starting at the beginning of the form. If we want to display any text at any point in the form we will use the two properties: **Currentx** and **Currenty** which are used to define the xy starting point for printing the text.

Ex: Print the string "College Of Engineering" starting at (1500,2500)Twip.

Ans:

CurrentX = 1500

CurrentY = 2500

Print "College of Engineering"

8-4: Points Drawing

The other important drawing method is the point drawing which can help use to draw different shapes and plots by drawing a successive and convergent points. The function **PSet** is used for this purpose with the use of the following format:

PSet (x,y), Color

Where x and y is the x-y drawing location for this point, and color is the color of it.

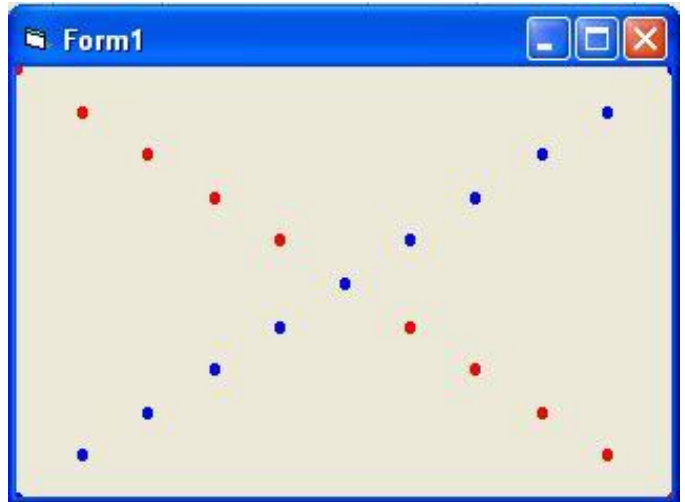
Example-4:

Write a VB program to draw ten points in main diagonal of the form and other ten points in the secondary diagonal of the same form.

Solution:

```
Private Sub Form_Load()  
Scale (0, 0)-(10, 10)  
DrawWidth = 5 End Sub
```

```
Private Sub Form_Click()  
For i = 0 To 10  
PSet (i, i), vbRed  
Next i  
For i = 10 To 0 Step -1  
PSet (10 - i, i), vbBlue  
Next i  
End Sub
```



8-5: Functions Drawing

It is a very important to draw the different mathematical functions like sine, cosine, tan, Log, Ln, ... etc. To plot any of these functions we must first locate the X and Y axis then by repeating PSet function many times we can plot a continuous shape. Repeating PSet function can be done by the use of loops.

Example-5:

Write a VB program to plot the trigonometric functions (Sin, Cos, Tan) when you click the Form.

Solution:

```
Const pi As Single = 3.1415926
```

```
Dim AmpScale, I As Single
```

```
Dim f1, f2, f3 As Single
```

```
Private Sub Form_Click()
```

```
Scale (0, 0)-(10, 10)
```

```
DrawWidth = 2
```

```
Line (0, 5)-(10, 5)
```

```
Line (5, 0)-(5, 10)
```

```
AmpScale= 10 / 8
```

```
For I = - 180 / pi To 180 / pi Step 0.01
```

```
f1 = -Sin (I - 5) * AmpScale + 5
```

```
f2 = -Cos (I - 5) * AmpScale + 5
```

```
f3 = -Tan (I - 5) * AmpScale + 5
```

' The first 5 for the phase shift and the second for the Amplitude shift

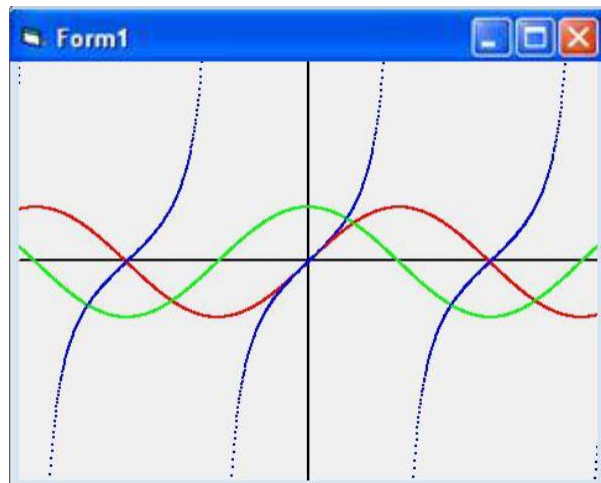
```
PSet (I, f1), vbRed
```

```
PSet (I, f2), vbGreen
```

```
PSet (I, f3), vbBlue
```

```
Next I
```

```
End Sub
```



8-6: Free Drawing

This type of drawing is not limited by any rules. Pset function, explained previously, is used for such case. Free drawing usually used for plotting unlimited features like curves, hand plotting, zigzag drawing, ... etc. The very wide used application uses this kind of drawing is the Paint program that is available in any windows copy when we select the Accessories in the Start menu.

Example-6:

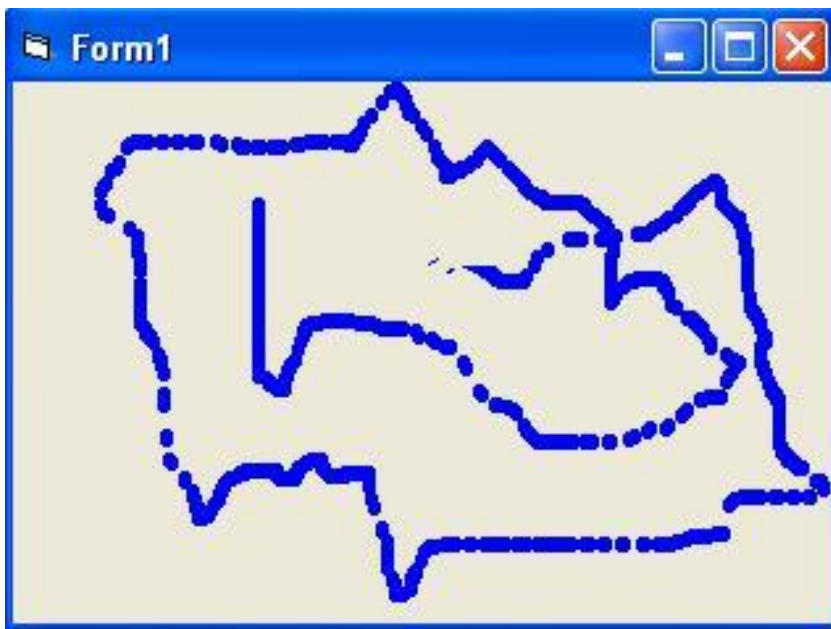
Write a VB program to design a simple paint program. The program begins plotting when we click the mouse left button and start erasing the plots when we click the mouse right button.

Solution:

```
Dim painting As Boolean
Private Sub Form_Load()
    DrawWidth = 5
End Sub
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single,
    Y As Single)
    painting = True
End Sub
```

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As
Single, Y As Single)
If painting And Button = 1 Then
PSet (X, Y), vbBlue
ElseIf painting And Button = 2 Then
PSet (X, Y), BackColor
End If
End Sub

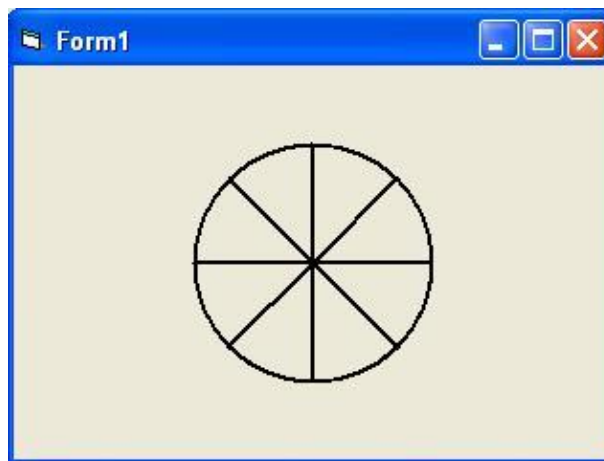
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As
Single, Y As Single)
painting = False
End Sub
```



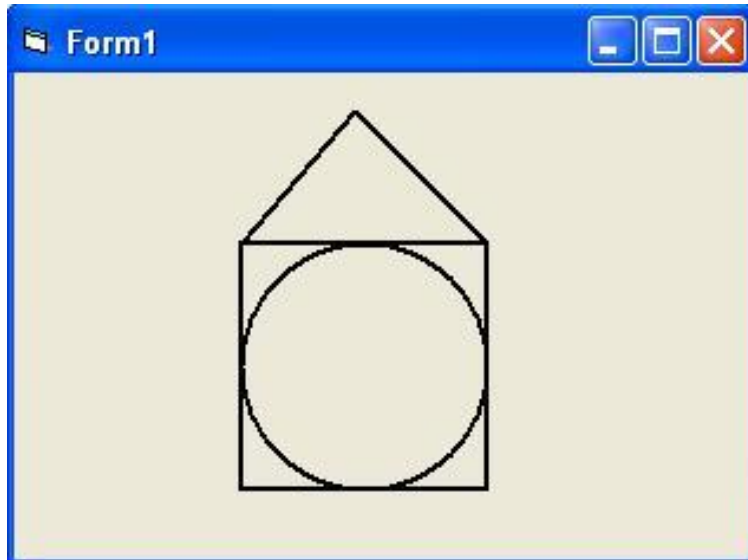
8.7 Problems

- 5.** Design a project contains two command buttons named "Line" and "Circle" When we clicked the command "Line" a straight line with starting point (240 , 480) and end point (1560 , 1680) will appear. The command "Circle" will draw a circle with center (3480 , y=1080) and radius (800). the Used unit is Twip.
- 6.** Write a VB program to draw two circles centered at the middle of the form with radiuses: 500 twip and 2000 twip. The project also contains two commands ("Max" and "Min") the first command used to maximize the first circle to double its size and the second used to minimize the second to 1/8 of its size.
- 7.** Design a VB project to plot a rectangular shape with length 800 Twip and width 400 Twip. Each line in this shape will be plotted according to click four command buttons.
- 8.** Design a VB project contains a single command button with name "Draw Three Shapes" and three shape boxes. When we click this command a three different shapes with three different fill styles will appear on this form.
- 9.** Design a VB project to draw a triangle then find and print the area and circumference for this triangle and view the results on two text boxes.

4. Design a VB project to draw a square then find and print the area and circumference for this square and view the results on two message boxes.
5. Design a VB project contains single command, named "Draw Sector and Arc" and two text boxes. When we click this command a sector and arc with radius 2 unit will be drawn centered in the middle of the form and their starting and ending points are entered from the two text boxes. Use a scale of (10).
6. Write a VB program to draw four points with width (2) at a rectangle corners. The length of this rectangle is (4) and width (3). Use a scale of (12).
7. Write a VB program to draw ten points with width (3) in each coordinates (X and Y) located at the middle of the form.
8. Write a VB program to view the following figure:



8. Write a VB program to view the following figure:



16. Write a VB program to draw the functions: $\sin^{-1}(x)$, $\cos^{-1}(x)$, and $\tan^{-1}(x)$ in the same figure and different colors. Use a scale of 10.

17. Write a VB program to draw the functions: $\sinh(x)$, $\cosh(x)$, and $\tanh(x)$ in the same figure and different colors. Use a scale of 20.

18. Write a VB program to draw the functions: $\ln(x)$, e^x in the same figure and different colors. Use a scale of 15.

19. Write a VB program to draw the functions: x^2 , x^3 in the same figure and different colors. Use a scale of 10.

20. Write a VB program to write the English alphabets. Use the mouse left button to draw and the left button to erase. Use the yellow color.

11. Design a project contains combo box to enter the 4 items represent width of pen drawing. Two option buttons ("Draw" and "Erase") are also added to this project if we select the first option the mouse left button will be a drawer button and if we select the second option the same mouse button will be an eraser button.

APPENDIX-A

Visual Basic Literal Constants

1. Align Constants:

Constants for the Align property.

vbAlignBottom
vbAlignLeft
vbAlignNone
vbAlignRight
vbAlignTop

Control at bottom of form (Align).
Control at left of form (Align).
Size and location set at design time or in code.
Control at right of form (Align).
Control at top of form (Align).

2. Alignment Constants:

Constants for the Alignment property.

vbCenter
vbLeftJustify
vbRightJustify

Center (Alignment).
Left justify (Alignment).
Right justify (Alignment).

3. Border-Style Constants:

Constants for the Border Style property of controls (not Forms).

vbBSDash
vbBSDashDot
vbBSDashDotDot
vbBSDot
vbBSInsideSolid
vbBSSolid
vbTransparent

Dash (shape and line BorderStyle).
Dash-dot (shape and line BorderStyle).
Dash-dot-dot (shape and line BorderStyle).
Dot (shape and line BorderStyle).
Inside solid (shape and line BorderStyle).
Solid (shape and line BorderStyle).
Transparent (shape and line BorderStyle).

4. Button Constants :

Button Constants (for CommandButton, CheckBox, and OptionButton).

vbButtonGraphical
vbButtonStandard

Graphical appearance (picture, text, and/or non-standard Backcolor).
Standard Windows appearance.

5. Check Box Constants : Checkbox Value property constants.

vbChecked	Checked check value.
vbGrayed	Grayed check value.
vbUnchecked	Unchecked check value.

6. Color Constants :Color constants (see also System Color Constants).

vbBlack	Black Color
vbBlue	Blue Color
vbCyan	Cyan Color
vbGreen	Green Color
vbMagenta	Magenta Color
vbRed	Red Color
vbWhite	White Color
vbYellow	Yellow Color

7. Combo Box Constants Combines the features of a TextBox control and a ListBox control.

vbComboDropdown	ComboBox control style that allows typing in a text box or selection from a drop-down list.
vbComboDropdownList	Only allows selection from the drop-down list in a ComboBox control.
vbComboSimple	ComboBox control style that allows typing in a text box or selection from a list, which doesn't drop down.

8. Constants : Predefined constants

vbBack	Constant for backspace character; equivalent to Chr\$(8)
vbCr	Constant for carriage return (without linefeed); equivalent to Chr\$(13)
vbCrLf	Constant for Carriage-return/Linefeed combination; equivalent of Chr\$(13)+Chr\$(10)
vbFormFeed	Constant for form feed (ASCII 12); equivalent to Chr\$(12)
vbLf	Constant for linefeed (without carriage return);

	equivalent to Chr\$(10)
vbNewLine	Constant for NewLine; platform specific
vbNullChar	Basic constant for a single Null character (ASCII value 0); equivalent to Chr\$(0)
vbNullString	Constant for use when calling external procedures requiring a string whose value is zero
vbObjectError	Constant indicating error is being returned from a Visual Basic object
vbTab	Constant for Tab character (ASCII 9); equivalent to Chr\$(9)
vbVerticalTab	Constant for vertical Tab (ASCII 11) character; equivalent to Chr\$(11)

9. Draw Style Constants : **Constants for the Drag method.**

vbDash	Dash (Draw Style).
vbDashDot	Dash-dot (Draw Style).
vbDashDotDot	Dash-dot-dot (Draw Style).
vbDot	Dot (Draw Style).
vbInsideSolid	Inside solid (Draw Style).
vbInvisible	Invisible (Draw Style).
vbSolid	Solid (Draw Style).

10. Fill Style Constants : **Locates and lists files in the directory specified by the Path property at run time.**

vbCross	Cross (FillStyle).
vbDiagonalCross	Diagonal cross (FillStyle).
vbDownwardDiagonal	Downward diagonal (FillStyle).
vbFSSolid	Solid (FillStyle).
vbFSTransparent	Transparent (FillStyle).
vbHorizontalLine	Horizontal line (FillStyle).
vbUpwardDiagonal	Upward diagonal (FillStyle).
vbVerticalLine	Vertical line (FillStyle).

11. Form Arrange Constants : **Constants for the Arrange method for MDI forms.**

vbArrangeIcons	Arrange icons for minimized MDI child forms.
vbCascade	Cascade all non-minimized MDI child forms.
vbTileHorizontal	Horizontally tile all non-minimized MDI child forms.
vbTileVertical	Vertically tile all non-minimized MDI child forms.

12. Key Code Constants : **Constants for the Key Board code.**

vbKey0	0 key.
vbKey1	1 key.
vbKey2	2 key.
vbKey3	3 key.
vbKey4	4 key.
vbKey5	5 key.
vbKey6	6 key.
vbKey7	7 key.
vbKey8	8 key.
vbKey9	9 key.
vbKeyA	A key.
vbKeyAdd	Plus (+) key on the numeric keypad.
vbKeyB	B key.
vbKeyBack	Backspace key.
vbKeyC	C key
vbKeyCancel	Cancel key.
vbKeyCapital	Caps key.
vbKeyClear	Clear key.
vbKeyControl	Ctrl key.
vbKeyD	D key.
vbKeyDecimal	Decimal (.) key on the numeric keypad.
vbKeyDelete	Del key.
vbKeyDivide	Divide (/) key on the numeric keypad.
vbKeyDown	Down key.
vbKeyE	E key.
vbKeyEnd	End key.
vbKeyEscape	Esc key.
vbKeyExecute	Execute key.

vbKeyF	F key.
vbKeyF1	F1 key.
vbKeyF2	F2 key.
vbKeyF3	F3 key.
vbKeyF4	F4 key.
vbKeyF5	F5 key.
vbKeyF6	F6 key.
vbKeyF7	F7 key.
vbKeyF8	F8 key.
vbKeyF9	F9 key.
vbKeyF10	F10 key.
vbKeyF11	F11 key.
vbKeyF12	F12 key.
vbKeyF13	F13 key.
vbKeyF14	F14 key.
vbKeyF15	F15 key.
vbKeyF16	F16 key.
vbKeyG	G key.
vbKeyH	H key.
vbKeyHelp	Help key.
vbKeyHome	Home key.
vbKeyI	I key.
vbKeyInsert	Insert key.
vbKeyJ	J key.
vbKeyK	K key.
vbKeyL	L key.
vbKeyLButton	Left mouse button
vbKeyLeft	Left key.
vbKeyM	M key.
vbKeyMButton	Middle mouse button.
vbKeyMenu	Menu key.
vbKeyMultiply	Multiply (*) key on the numeric keypad.
vbKeyN	N key.
vbKeyNumlock	Num Lock key.
vbKeyNumpad0	0 key on the numeric keypad.
vbKeyNumpad1	1key on the numeric keypad.
vbKeyNumpad2	2 key on the numeric keypad.
vbKeyNumpad3	3 key on the numeric keypad.
vbKeyNumpad4	4 key on the numeric keypad.
vbKeyNumpad5	5 key on the numeric keypad.
vbKeyNumpad6	6 key on the numeric keypad.

vbKeyNumpad7	7 key on the numeric keypad.
vbKeyNumpad8	8 key on the numeric keypad.
vbKeyNumpad9	9 key on the numeric keypad.
vbKeyO	O key.
vbKeyP	P key.
vbKeyPageDown	PAGE DOWN key.
vbKeyPageUp	PAGE UP key.
vbKeyPause	Pause key.
vbKeyPrint	PrintScreen key.
vbKeyQ	Q key.
vbKeyR	R key.
vbKeyRButton	Right mouse button.
vbKeyReturn	Return (Enter) key.
vbKeyRight	Right key.
vbKeyS	S key.
vbKeyScrollLock	Scroll Lock key.
vbKeySelect	Select key.
vbKeySeparator	Enter key on the numeric keypad.
vbKeyShift	Shift key.
vbKeySnapshot	Snapshot key.
vbKeySpace	Spacebar key.
vbKeySubtract	Minus (-) key on the numeric keypad.
vbKeyT	T key.
vbKeyTab	Tab key.
vbKeyU	U key.
vbKeyUp	Up key.
vbKeyV	V key.
vbKeyW	W key.
vbKeyX	X key.
vbKeyY	Y key.
vbKeyZ	Z key.

13. Menu Control Constants :

vbPopupMenuCenterAlign
vbPopupMenuLeftAlign
vbPopupMenuLeftButton
vbPopupMenuRightAlign
vbPopupMenuRightButton

Miscellaneous Menu control constants.

Pop-up menu centered.
Pop-up menu aligned left.
Pop-up menu recognizes left mouse button only.
Pop-up menu aligned right. Pop-up menu recognizes right and left mouse button.

14. Mouse Button Constants : Mouse button bit field constants.

vbLeftButton	Left mouse button.
vbMiddleButton	Middle mouse button.
vbRightButton	Right mouse button.

15. Mouse Pointer Constants : Mouse Pointer property constants.

vbArrow	Arrow mouse pointer.
vbArrowHourglass	Arrow and hourglass.
vbArrowQuestion	Arrow and question mark.
vbCrosshair	Cross mouse pointer.
vbCustom	Custom mouse pointer icon specified by the Mouse Icon property.
vbDefault	Default (MousePointer).
vbHourglass	Hourglass mouse pointer.
vbIbeam	I-Beam mouse pointer.
vbIconPointer	Icon mouse pointer.
vbNoDrop	No drop mouse pointer.
vbSizeAll	Size all.
vbSizeNESW	Size NE SW mouse pointer.
vbSizeNS	Size N S mouse pointer.
vbSizeNWSE	Size NW SE mouse pointer.
vbSizePointer	Size mouse pointer.
vbSizeWE	Size W E mouse pointer.
vbUpArrow	Up arrow mouse pointer.

16. Scroll Bar Constants : Scrollbar property constants.

vbBoth	Both horizontal and vertical scroll bars.
vbHorizontal	Horizontal scroll bar.
vbSBNone	No scroll bar.
vbVertical	Vertical scroll bar.

5. Shape Constants : Constants for the Shape property of the Shape control.

vbShapeCircle	Circle shape.
vbShapeOval	Oval shape.
vbShapeRectangle	Rectangle shape.

vbShapeRoundedRectangle	Rounded rectangle shape.
vbShapeRoundedSquare	Rounded square shape.
vbShapeSquare	Square shape.

18. Shift Constants **Shift parameter constants (bit fields for the Shift parameter of various events).**

vbAltMask	Alt key bit mask.
vbCtrlMask	Ctrl key bit mask.
vbShiftMask	Shift key bit mask.

19. Calendar **Constants for the Calendar**

vbCalGreg	Constant for specifying Gregorian Calendar
vbCalHijri	Constant for specifying Hijri Calendar

20. Date and Time Constants

vbGeneralDate
vbLongDate
vbLongTime
vbShortDate
vbShortTime
vbFriday
vbMonday
vbSaturday
vbSunday
vbThursday
vbTuesday
vbUseSystemDayOfWeek
vbWednesday
vbFirstFourDays
vbFirstFullWeek
vbFirstJan1
vbUseSystem

APPENDIX-A

Visual Basic Literal Constants

1. Align Constants:

Constants for the Align property.

vbAlignBottom
vbAlignLeft
vbAlignNone
vbAlignRight
vbAlignTop

Control at bottom of form (Align).
Control at left of form (Align).
Size and location set at design time or in code.
Control at right of form (Align).
Control at top of form (Align).

2. Alignment Constants:

Constants for the Alignment property.

vbCenter
vbLeftJustify
vbRightJustify

Center (Alignment).
Left justify (Alignment).
Right justify (Alignment).

3. Border-Style Constants:

Constants for the Border Style property of controls (not Forms).

vbBSDash
vbBSDashDot
vbBSDashDotDot
vbBSDot
vbBSInsideSolid
vbBSSolid
vbTransparent

Dash (shape and line BorderStyle).
Dash-dot (shape and line BorderStyle).
Dash-dot-dot (shape and line BorderStyle).
Dot (shape and line BorderStyle).
Inside solid (shape and line BorderStyle).
Solid (shape and line BorderStyle).
Transparent (shape and line BorderStyle).

4. Button Constants :

Button Constants (for CommandButton, CheckBox, and OptionButton).

vbButtonGraphical
vbButtonStandard

Graphical appearance (picture, text, and/or non-standard Backcolor).
Standard Windows appearance.

5. Check Box Constants : Checkbox Value property constants.

vbChecked	Checked check value.
vbGrayed	Grayed check value.
vbUnchecked	Unchecked check value.

7. Color Constants :Color constants (see also System Color Constants).

vbBlack	Black Color
vbBlue	Blue Color
vbCyan	Cyan Color
vbGreen	Green Color
vbMagenta	Magenta Color
vbRed	Red Color
vbWhite	White Color
vbYellow	Yellow Color

7. Combo Box Constants Combines the features of a TextBox control and a ListBox control.

vbComboDropdown	ComboBox control style that allows typing in a text box or selection from a drop-down list.
vbComboDropdownList	Only allows selection from the drop-down list in a ComboBox control.
vbComboSimple	ComboBox control style that allows typing in a text box or selection from a list, which doesn't drop down.

8. Constants : Predefined constants

vbBack	Constant for backspace character; equivalent to Chr\$(8)
vbCr	Constant for carriage return (without linefeed); equivalent to Chr\$(13)
vbCrLf	Constant for Carriage-return/Linefeed combination; equivalent of Chr\$(13)+Chr\$(10)
vbFormFeed	Constant for form feed (ASCII 12); equivalent to Chr\$(12)
vbLf	Constant for linefeed (without carriage return);

	equivalent to Chr\$(10)
vbNewLine	Constant for NewLine; platform specific
vbNullChar	Basic constant for a single Null character (ASCII value 0); equivalent to Chr\$(0)
vbNullString	Constant for use when calling external procedures requiring a string whose value is zero
vbObjectError	Constant indicating error is being returned from a Visual Basic object
vbTab	Constant for Tab character (ASCII 9); equivalent to Chr\$(9)
vbVerticalTab	Constant for vertical Tab (ASCII 11) character; equivalent to Chr\$(11)

9. Draw Style Constants : Constants for the Drag method.

vbDash	Dash (Draw Style).
vbDashDot	Dash-dot (Draw Style).
vbDashDotDot	Dash-dot-dot (Draw Style).
vbDot	Dot (Draw Style).
vbInsideSolid	Inside solid (Draw Style).
vbInvisible	Invisible (Draw Style).
vbSolid	Solid (Draw Style).

10. Fill Style Constants : Locates and lists files in the directory specified by the Path property at run time.

vbCross	Cross (FillStyle).
vbDiagonalCross	Diagonal cross (FillStyle).
vbDownwardDiagonal	Downward diagonal (FillStyle).
vbFSSolid	Solid (FillStyle).
vbFSTransparent	Transparent (FillStyle).
vbHorizontalLine	Horizontal line (FillStyle).
vbUpwardDiagonal	Upward diagonal (FillStyle).
vbVerticalLine	Vertical line (FillStyle).

11. Form Arrange Constants : Constants for the Arrange method for MDI forms.

vbArrangeIcons	Arrange icons for minimized MDI child forms.
vbCascade	Cascade all non-minimized MDI child forms.
vbTileHorizontal	Horizontally tile all non-minimized MDI child forms.
vbTileVertical	Vertically tile all non-minimized MDI child forms.

12. Key Code Constants : Constants for the Key Board code.

vbKey0	0 key.
vbKey1	1 key.
vbKey2	2 key.
vbKey3	3 key.
vbKey4	4 key.
vbKey5	5 key.
vbKey6	6 key.
vbKey7	7 key.
vbKey8	8 key.
vbKey9	9 key.
vbKeyA	A key.
vbKeyAdd	Plus (+) key on the numeric keypad.
vbKeyB	B key.
vbKeyBack	Backspace key.
vbKeyC	C key
vbKeyCancel	Cancel key.
vbKeyCapital	Caps key.
vbKeyClear	Clear key.
vbKeyControl	Ctrl key.
vbKeyD	D key.
vbKeyDecimal	Decimal (.) key on the numeric keypad.
vbKeyDelete	Del key.
vbKeyDivide	Divide (/) key on the numeric keypad.
vbKeyDown	Down key.
vbKeyE	E key.
vbKeyEnd	End key.
vbKeyEscape	Esc key.
vbKeyExecute	Execute key.

vbKeyF	F key.
vbKeyF1	F1 key.
vbKeyF2	F2 key.
vbKeyF3	F3 key.
vbKeyF4	F4 key.
vbKeyF5	F5 key.
vbKeyF6	F6 key.
vbKeyF7	F7 key.
vbKeyF8	F8 key.
vbKeyF9	F9 key.
vbKeyF10	F10 key.
vbKeyF11	F11 key.
vbKeyF12	F12 key.
vbKeyF13	F13 key.
vbKeyF14	F14 key.
vbKeyF15	F15 key.
vbKeyF16	F16 key.
vbKeyG	G key.
vbKeyH	H key.
vbKeyHelp	Help key.
vbKeyHome	Home key.
vbKeyI	I key.
vbKeyInsert	Insert key.
vbKeyJ	J key.
vbKeyK	K key.
vbKeyL	L key.
vbKeyLButton	Left mouse button
vbKeyLeft	Left key.
vbKeyM	M key.
vbKeyMButton	Middle mouse button.
vbKeyMenu	Menu key.
vbKeyMultiply	Multiply (*) key on the numeric keypad.
vbKeyN	N key.
vbKeyNumlock	Num Lock key.
vbKeyNumpad0	0 key on the numeric keypad.
vbKeyNumpad1	1key on the numeric keypad.
vbKeyNumpad2	2 key on the numeric keypad.
vbKeyNumpad3	3 key on the numeric keypad.
vbKeyNumpad4	4 key on the numeric keypad.
vbKeyNumpad5	5 key on the numeric keypad.
vbKeyNumpad6	6 key on the numeric keypad.

vbKeyNumpad7	7 key on the numeric keypad.
vbKeyNumpad8	8 key on the numeric keypad.
vbKeyNumpad9	9 key on the numeric keypad.
vbKeyO	O key.
vbKeyP	P key.
vbKeyPageDown	PAGE DOWN key.
vbKeyPageUp	PAGE UP key.
vbKeyPause	Pause key.
vbKeyPrint	PrintScreen key.
vbKeyQ	Q key.
vbKeyR	R key.
vbKeyRButton	Right mouse button.
vbKeyReturn	Return (Enter) key.
vbKeyRight	Right key.
vbKeyS	S key.
vbKeyScrollLock	Scroll Lock key.
vbKeySelect	Select key.
vbKeySeparator	Enter key on the numeric keypad.
vbKeyShift	Shift key.
vbKeySnapshot	Snapshot key.
vbKeySpace	Spacebar key.
vbKeySubtract	Minus (-) key on the numeric keypad.
vbKeyT	T key.
vbKeyTab	Tab key.
vbKeyU	U key.
vbKeyUp	Up key.
vbKeyV	V key.
vbKeyW	W key.
vbKeyX	X key.
vbKeyY	Y key.
vbKeyZ	Z key.

13. Menu Control Constants :

vbPopupMenuCenterAlign
vbPopupMenuLeftAlign
vbPopupMenuLeftButton
vbPopupMenuRightAlign
vbPopupMenuRightButton

Miscellaneous Menu control constants.

Pop-up menu centered.
Pop-up menu aligned left.
Pop-up menu recognizes left mouse button only.
Pop-up menu aligned right. Pop-up menu recognizes right and left mouse button.

14. Mouse Button Constants : Mouse button bit field constants.

vbLeftButton	Left mouse button.
vbMiddleButton	Middle mouse button.
vbRightButton	Right mouse button.

15. Mouse Pointer Constants : Mouse Pointer property constants.

vbArrow	Arrow mouse pointer.
vbArrowHourglass	Arrow and hourglass.
vbArrowQuestion	Arrow and question mark.
vbCrosshair	Cross mouse pointer.
vbCustom	Custom mouse pointer icon specified by the Mouse Icon property.
vbDefault	Default (MousePointer).
vbHourglass	Hourglass mouse pointer.
vbIbeam	I-Beam mouse pointer.
vbIconPointer	Icon mouse pointer.
vbNoDrop	No drop mouse pointer.
vbSizeAll	Size all.
vbSizeNESW	Size NE SW mouse pointer.
vbSizeNS	Size N S mouse pointer.
vbSizeNWSE	Size NW SE mouse pointer.
vbSizePointer	Size mouse pointer.
vbSizeWE	Size W E mouse pointer.
vbUpArrow	Up arrow mouse pointer.

16. Scroll Bar Constants : Scrollbar property constants.

vbBoth	Both horizontal and vertical scroll bars.
vbHorizontal	Horizontal scroll bar.
vbSBNone	No scroll bar.
vbVertical	Vertical scroll bar.

6. Shape Constants : Constants for the Shape property of the Shape control.

vbShapeCircle	Circle shape.
vbShapeOval	Oval shape.
vbShapeRectangle	Rectangle shape.

vbShapeRoundedRectangle	Rounded rectangle shape.
vbShapeRoundedSquare	Rounded square shape.
vbShapeSquare	Square shape.

18. Shift Constants **Shift parameter constants (bit fields for the Shift parameter of various events).**

vbAltMask	Alt key bit mask.
vbCtrlMask	Ctrl key bit mask.
vbShiftMask	Shift key bit mask.

19. Calendar **Constants for the Calendar**

vbCalGreg	Constant for specifying Gregorian Calendar
vbCalHijri	Constant for specifying Hijri Calendar

20. Date and Time Constants

vbGeneralDate
vbLongDate
vbLongTime
vbShortDate
vbShortTime
vbFriday
vbMonday
vbSaturday
vbSunday
vbThursday
vbTuesday
vbUseSystemDayOfWeek
vbWednesday
vbFirstFourDays
vbFirstFullWeek
vbFirstJan1
vbUseSystem

APPENDIX-C

Reserved Words

A	Clear	DefDbl	Exp
Abs	Clipboard	DefInt	Explicit
Access	CLng	DefLng	F
AddItem	Close	DefObj	FALSE
AddNew	Cls	DefSng	FieldSize
Alias	Command	DefStr	FileAttr
Alphanumeric	Command\$	DefVar	FileCopy
And	CommitTrans	Delete	FileDateTime
Any	CompactDatabase	DeleteSetting	FileLen
App	Compare	Dim	Fix
AppActivate	Const	Dir	For
Append	Control	Dir\$	Form
AppendChunk	Controls	Do	Format
Arrange	Cos	DoEvents	Format\$
As	Count	Double	Forms
Asc	CreateDynaset	Drag	FreeFile
Atn	CSng	Dynaset	FreeLocks
B	CStr	E	Function
ase	CurDir\$	Each	G
Beep	Currency	Edit	Get
BeginTrans	CVar	Else	GetAttr
Between	CVDate	Elself	GetChunk
Binary	D	End	GetData
ByVal	Data	EndDoc	DetFormat
C	Date	EndIf	GetText
Call	Date\$	Environ\$	Global
Case	DateSerial	EOF	GoSub
CCur	DateValue	Eqv	GoTo
CDbl	Day	Erase	H
ChDir	Debug	Erl	Hex
ChDrive	Declare	Err	Hex\$
Chr	DefBool	Error	Hide
Chr\$	DefByte	Error\$	Hour
CInt	DefCur	ExecuteSQL	
Circle	DefDate	Exit	

I	Loop	Or	SaveSetting
If	LSet	P	Scale
InputBox	LTrim	Point	Second
InputBox\$	LTrim\$	Preserve	Seek
InStr	M	Print	Select
Int	Max	Printer	SendKeys
Integer	Me	PrintForm	Set
Is	Mid	Private	SetAttr
IsDate	Mid\$	Property	SetData
IsEmpty	Min	PSet	SetDataAccessOption
IsNull	Minute	Public	SetDefaultWorkspace
IsNumeric	MkDir	Q	SetFocus
K	Mod	QBColor	SetText
Kill	Month	R	Sgn
L	Move	Random	Shared
LBound	MoveFirst	Randomize	Shell
LCase	MoveLast	Read	Show
LCase\$	MoveNext	ReDim	Sin
Left	MovePrevious	Refresh	Single
Left\$	MoveRelative	RegisterDataBase	Space
Len	MsgBox	Rem	Space\$
Let	N	RemoveItem	Spc
Level	Name	RepairDatabase	Sqr
Lib	New	Reset	Static
Like	NewPage	Restore	StDev
Line	Next	Resume	StDevP
LinkExecute	NextBlock	Return	Step
LinkPoke	Not	RGB	Stop
LinkRequest	Nothing	Right	Str
LinkSend	Now	Right\$	Str\$
Load	Null	Rmdir	StrComp
LoadPicture	O	Rnd	String
Loc	Oct	Rollback	String\$
Local	Oct\$	RSet	Sub
Lock	On	RTrim	Sum
LOF	Open	RTrim\$	System
Log	OpenDataBase	S	T
Long	Option	SavePicture	Tab

TableID	Width		
Tan	Write		
Text	X		
TextHeight	Xor		
TextWidth	Y		
Then	Year		
Time	Z		
Time\$	ZOrder		
Timer			
TimeSerial			
TimeValue			
To			
Trim			
Trim\$			
TRUE			
Type			
TypeOf			
U			
UBound			
UCase			
UCase\$			
Unload			
Unlock			
Until			
Update			
Using			
V			
Val			
Var			
Variant			
VarP			
VarType			
W			
Weekday			
Wend			
While			