

University of Baghdad

Collage of Education For Pure Science

Dep.of Computer Science

Third Stage

Windows Programming

**Microsoft
Visual Basic
6.0**

Chapter One

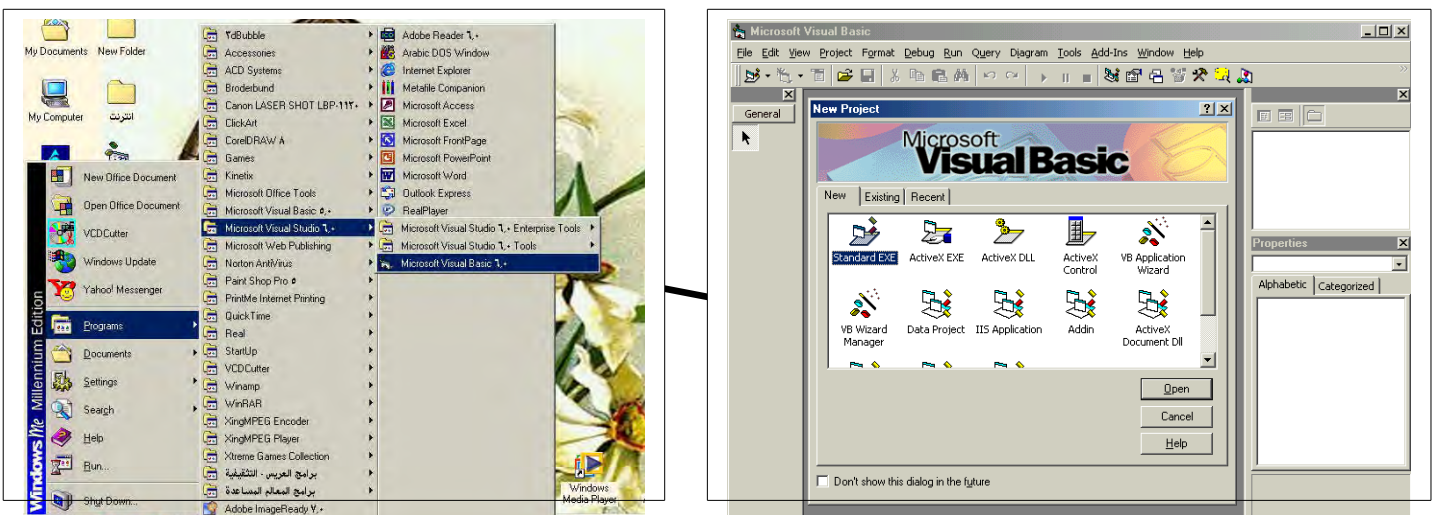
M.Sc. Lubab Ahmed

Introduction

Visual Basic implements graphical user interface that allows the use of graphics for different applications. It provides visual interactive windows with user, like Dialogue box for (color, font ...), Input box, and Output box. Also it is able to create menu to simplify user application.

To run this program on user computer: Start>programs>Microsoft Visual Studio 6.0>Microsoft Visual Basic 6.0.

It will appear on the computer screen as in the following picture.



1.2-The Importance of Visual Basic Program

Languages like Basic and Pascal depend on variables and procedures to build the applications. This is why it is called procedural languages. The new approach is called object programming for visual programs like Visual Basic and Visual C++ and others. In this programming approach everything (form, command buttons, controls) is an object.

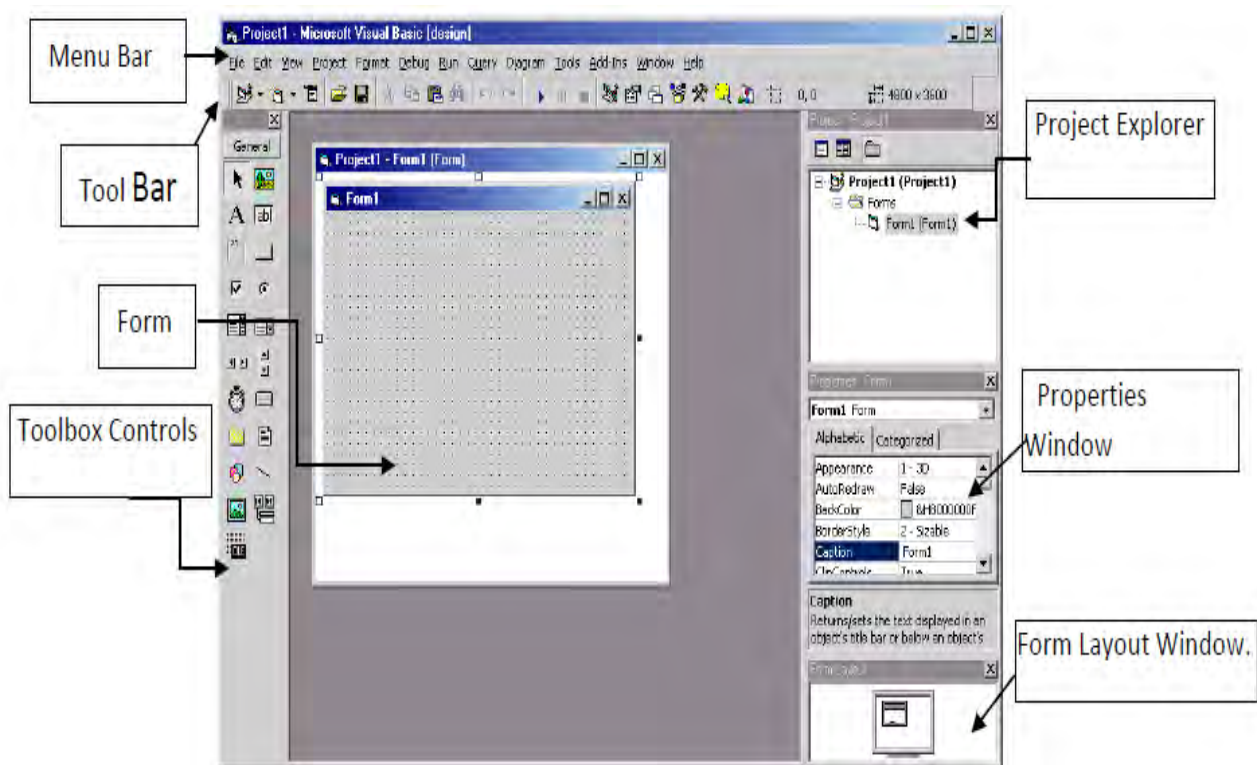
The reasons for of implementing Visual Basic program are listed as follows:

- 1- It uses integrated development environment (IDE) which is easier for the user to minimize code writing.
- 2- All visual programs follow the same concepts, therefore the user will become more familiar with visual approach for other visual languages.
- 3- It provides Input box and Output box as an interactive windows with user.

4- It is able to connect to Internet, and to call Explorer.

1.3- Elements of the Integrated Development Environment (IDE)

Figure below shows The IDE after Standard EXE is selected. The top of the IDE window (the title bar) displays "Project1-Microsoft Visual Basic [design]".The environment consists of various windows when Visual Basic is started (by default):



a- Menu Bar: It contains a standard command like: File, Edit, View, Window, Help menus, and specific command such as: Project, Format, or Debug menus.

b- Toolbar: It contains the most commonly used commands (button), if clicked an action represented by that button is carried out.

c- Toolbox: It contains a collection of tools that are needed for project design.

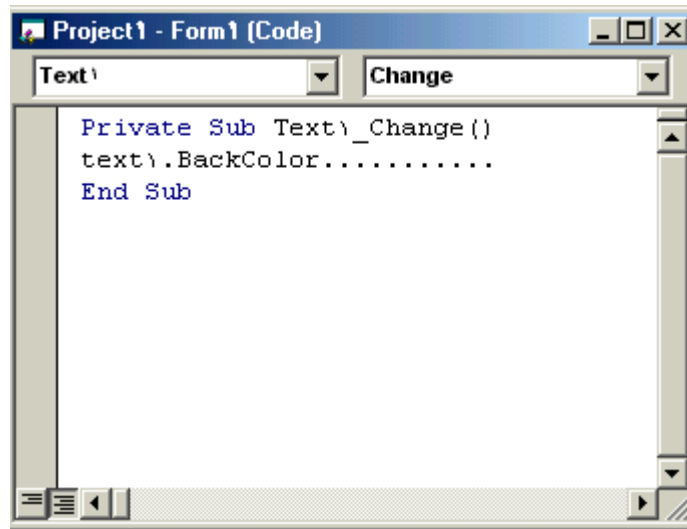
d- Form Designer: It is a window for each form to customize the designed interface of the application. Using the form designer, the user can add controls, graphics, and text to create the desired form appearance.

e- Properties Window: It is a list of properties settings for a selected form or a control. These properties are characteristics (such as size, visible, or color) of the selected object it provides an easy way to set properties.

f- Project Explorer Window: it is a list of the forms and modules for the current projects. It is a hierarchical tree- branch structure, where the project at top of tree and other parts like forms ,modules) descend from this tree.

g- Form Layout Window: The Form Layout window is a small screen. Which is used to reposition the form of the application so that it appears in proper place when project is run.

h- Code Editor Window: Code Editor Window is used to write a VB code for an application. For each form there is a separate code editor window. It is displayed when user clicks on form or object in form.



To Create an Application

The title of program includes the name of project, and when the user first starts the program it takes a defaulted value (project).It also include resize icons. The following steps are required to create an application in Visual Basic 6.0:

- 1- Select type of project New or Existing. A form automatically appears in the form design .The basis for any application's interface is the form that user should create. User can add other forms to the project (to add another form select project menu>add form).
2. To add objects (controls) to the form use the ToolBox.
3. Set the properties for the objects through properties window.
4. Write code. The Visual Basic Code consists of statements, and declarations. The code for an application can be written on the Code Editor window. In this window user can view and edit quickly any of the code.
5. Run the Application. To run the application, click the Start button on the toolbar, or press F5.

6. Stop. To stop running the application and return to visual basic program on stop button in tool bar.
7. Check if there is an error, return to step 3 ,otherwise continue.
8. Save project.
9. Exit.

Project

Project is a program designed to user application that may be simple (like calculator program) or complex (like word program). Visual basic program can create many types of projects. The most important or usual project is the standard project (for window applications) and the DHTML project (for internet).

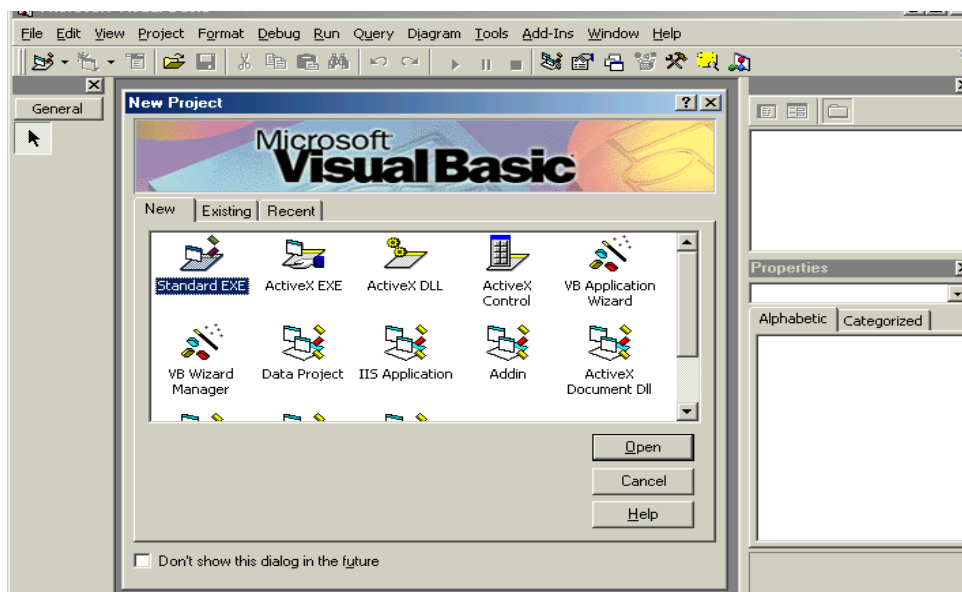
Working with Standard Projects:

The following working steps (create, save, add, open and delete) could be done:

a) To create project:

When program starts, project box appears-select Standard EXE > Project window appears.

OR: File> New project> Box (select Standard EXE)> Project window appears



b) To add project: Any number can be added.

Project icon> Select Standard EXE> Project window appears.

Note: Usually first project runs first, but user can change that by:
Selecting project from project window > mouse list > Set as startup.

c) To open an existing project:

It is previously designed and saved on disc in a folder.

File> Open project> Box (select existing and look for the project) > Project window

d) To delete a project:

Select project in Project window > Mouse list > Remove project.

e) To save project:

The visual basic can save the project on disc in two ways, as an executable type or a non-executable type.

A- for project in non execution stage:

There are many types of files summarized as follows:

1- Project file: it consists of all files which are related to specific project, also some other information with it. This could be saved with extension

(.VBP)

2- The form Files: this contains form description and any Object or program related to it .This is saved with extension (.frm).

To save project for first time:

File>Save project (group) as>Box (project name)> forms saved then projects group saved.

To resave project: to save previously saved project in same place

File>save project (group)

Note: If a form is modified it should be saved. To save a form:

Select a form from Project window>File>Save project form1 as > Save box (select form name). OR: File>Save project form1.

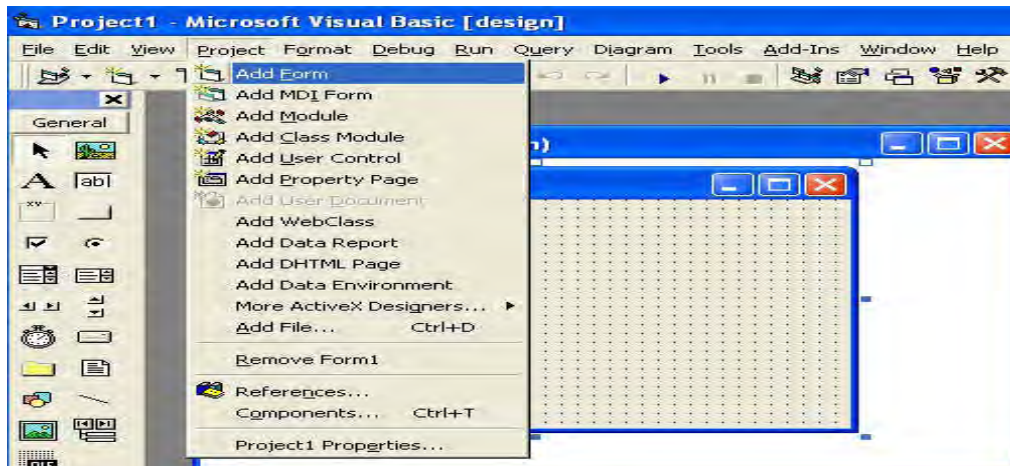
B- project for execution: This is the final stage so that it could be opened and run by Windows and no need for Visual Basic program. File> Make project.exe.

Item		Action steps	Remarks
Create project	New	File>New project	The user can open any number of projects.
	Exist	File>Open project	Project was already designed and saved.
	Recent	File>Open project	Project was recently designed and saved.
Save project		File>Save project group as	Visual Basic can deal with it (open and modify).
		File>Make projectl.exe	For execution by window.
Delete project		File> Remove project	Select project before remove.

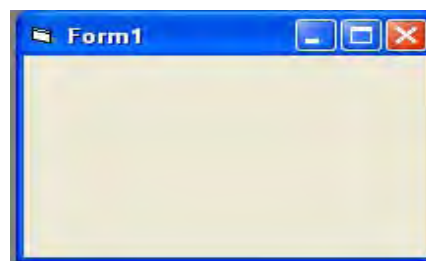
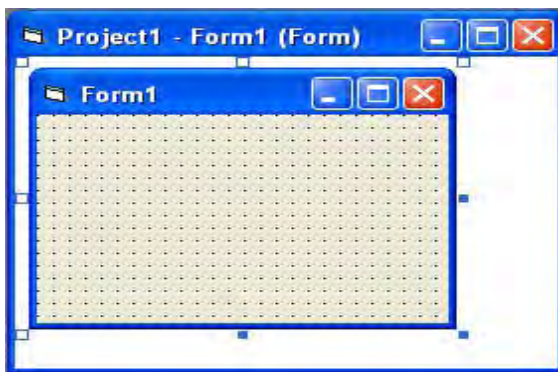
Forms

1) Introduction to form

The form is the most important visible object, without it no control can be displayed. It is a window that can be designed and modified to fit user applications. In the standard project the form Designer creates and modifies visual forms .When user starts visual Basic program a form is automatically displayed in Designer window. The designer can add any number of forms to the project of his application by pressing: add form from project menu.



There are two modes: design mode and running mode. User can interchange between them, by pressing on start icon or stop icon on tool bar.



The forms also have properties and events.

2) Form properties

Properties list has a predefined value (numeric or string) and could be changed, some properties could be rewritten like caption, and some could be selected from option list by pressing on down arrow on the side.

Others could be rewritten or by browsing the computer files when the user clicks on the dotted button on the right side a dialogue box appears. The browsing button appears when the user clicks inside the box.

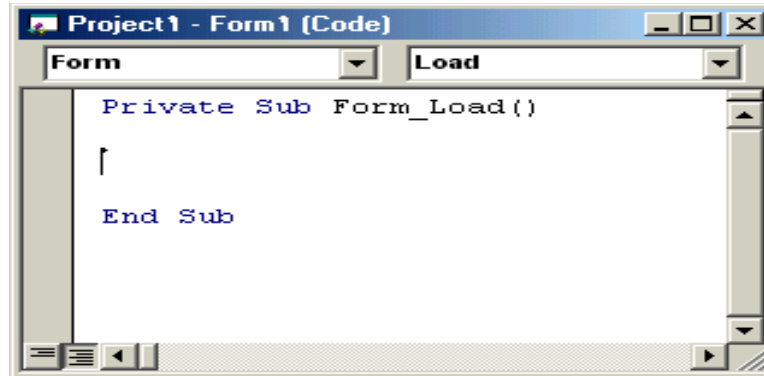
The most important properties of the form are listed in the following table:

Property name	Objective	code	Stage of Changing
Name	Used to represent name of form in code		Design
Caption	String appear in title of form	Form _{no.} .caption= "any name"	Design and run
BackColor	Background color for form.	Form _{no.} .BackColor=Qbcolor(no.)	Design and run
ForeColor	Color of text written on form.	Form _{no.} .forecolor=Qbcolor(no.)	Design and run
Font	Font style, type and size.	Size: Form _{no.} .fontsize= no. Style: $font \begin{cases} italic \\ bold \\ underline \end{cases}$ Type: Form1.FontName = "arial"	Design and run
borderstyle	0-None borderless and captionless 1-Fixed Single a nonresizable form 2-Sizable (default), creates a resizable window. 3-Fixed Dialog: nonresizable form without Minimize and Maximize buttons 4-Fixed Tool window for a floating toolbox like form. 5-Sizable Tool window		Design
Enabled	The tools enable or disable.	Form _{no.} . Enabled =true or false	Design and run
Min button Max button	=true. The Minimize and Maximize buttons are enabled. =false. The Minimize and Maximize button are disabled.	Form1.MaxButton = True or = false Form1.MinButton = True or = false	Run
Start up position	0- Manual ,use form layer window to position Form 1- Center owner 2- at Center Screen 3- Windowdefault.		Design
movable	True or false to make form movable or unmovable		Design
Hide	To hide the form	Form _{no.} .hide	Run
show	To show the form	Form _{no.} .Show	Run
icon	Change the icon on title bar of form (the icon must have the extension ico or cur)		

3) Code form

The code is written in code Form and it will be edited quickly by code editor . The codes are of two categories:

- 1- Declaration is written before any procedure in the code
- 2- Statements. The user selects the required event then code statements are written inside these event procedures.



4) Events:

Events are like electrical switches. The electrical switches are of many types, so are the events.

The forms and controls support events (generation, interaction with mouse and keyboard). The most important events for the form are described in the following table.

Event	Action taken when
Click	Single click on object.
DbClick	Double click on object.
load	Loading the object

Examples:

- 1- Design a form such that: in event load, when project runs, the form bgcolor property changed (chose any color).

sol:

code:

```
Private Sub Form_Load()  
Form1.BackColor = QBColor(12)  
End Sub
```



Toolbox Controls: Contains a collection of tools that are needed for project design as shown in Fig. below. To show the toolbox press View> toolbox icon. The user can place the tool on form, and then work with the tool. To place the tool on form: click on tool>draw tool to form > the tool appears on form or double click on tool then the tool appears on form. Table (1) summarizes the toolbox controls.

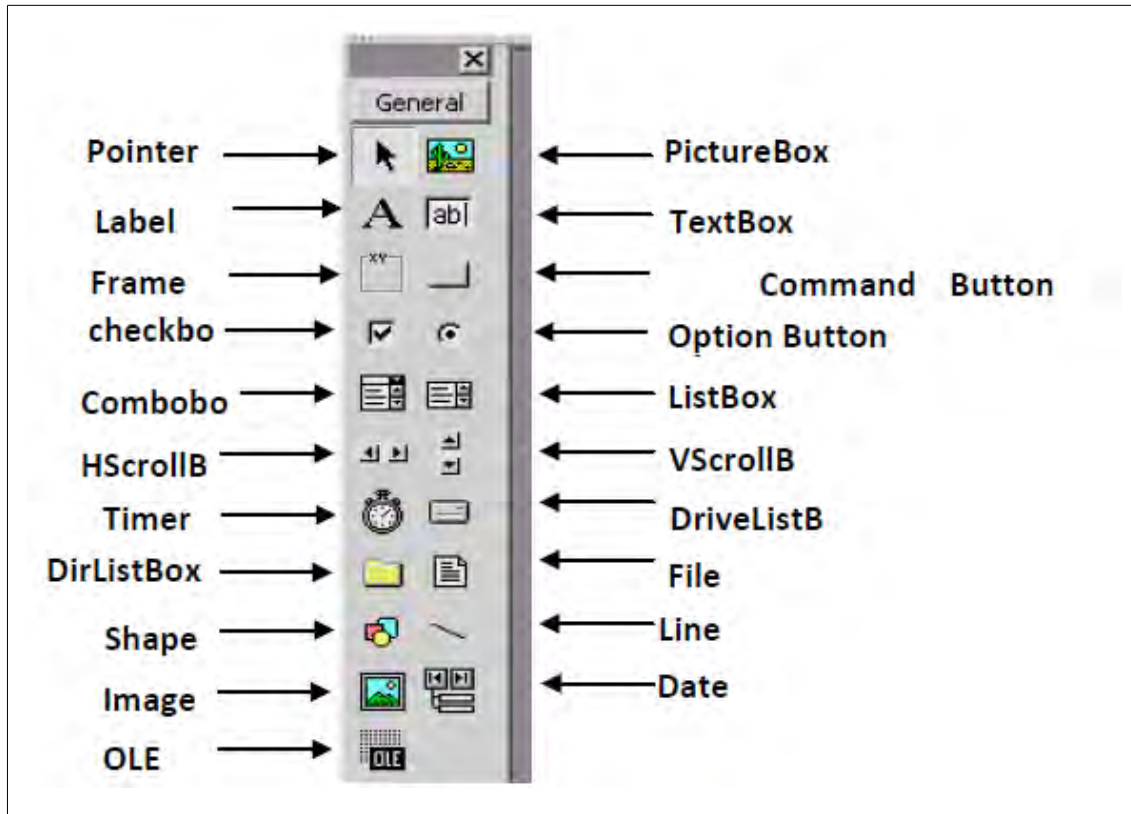


Table (1): Toolbox controls summary.

Control	Description
Pointer	Used to interact with controls on the form(resize them, move them, etc.). The pointer is not a control
PictureBox	A control that display images or print the result.
Label	A control that displays uneditable text to the user.
TextBox	A control for accepting user input. Textbox can also display text.
Frame	A control for grouping other controls.
CommandButton	A control that represents a button. The user presses or clicks to initiate an action.
CheckBox	A control that provides the user with a toggle choice (checked or unchecked)
OptionButton	Option buttons are used in groups where only one at a time can be true.
ListBox	A control that provides a list of items.
ComboBox	A control that provides a short list of items.
HscrollBar	A horizontal scrollbar.
VscrollBar	A vertical scrollbar.
Shape	A control for drawing circles, rectangles, squares or ellipse
Line	A control for drawing line.
DrivelistBox	A control accessing the system disk drivers.
DirlistBox	A control accessing directories on a system
Filelistbox	A control accessing file in a directory
Image	A control for displaying images. The images control does not provide as many capabilities as a picturebox.
OLE	A control for interacting with other window applications.
Timer	A control that performs a task at programmer specified intervals. A timer is not visible to the user.

Tool Box and Form

The user can place the tool on form and then work with the tool. To place the tool on form:

Click on tool > Draw tool to Form > the tool appears on Form.

Or: double click on it.

Notes:

a) Each tool has a property window .To see this window: Click on tool on form > Property window appears.

b) Property can be changed manually or by code and the effect of code appears in the run time (when user runs project).

c) To put code for tool action:

Double click on tool > code sheet of the Form appears (with code of corresponding tool is written) > User write the desired code inside tool event, or outside in Form event.

Label:

It is used to display fixed text on form



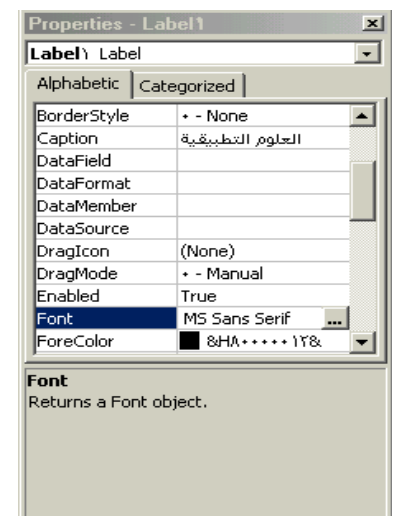
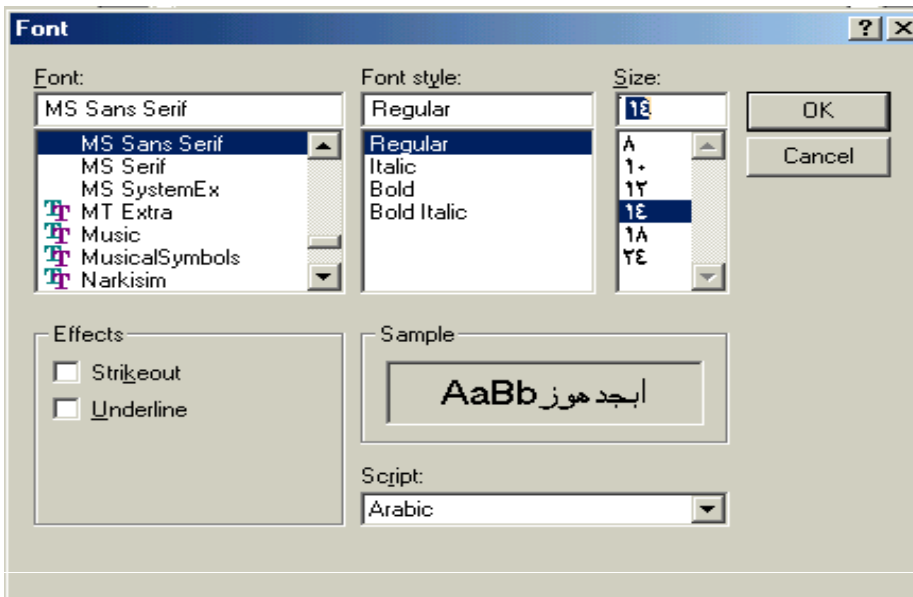
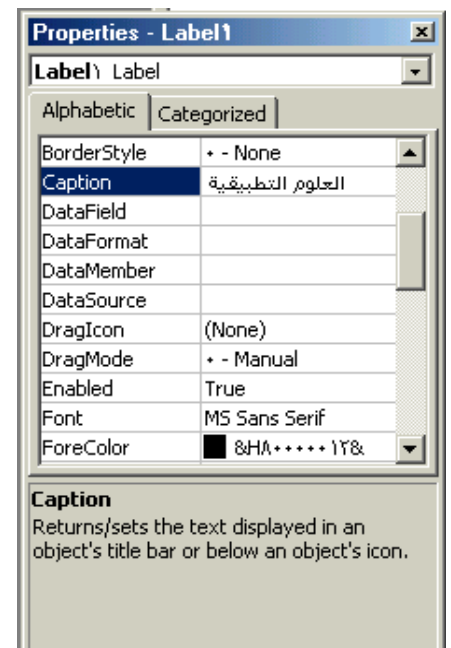
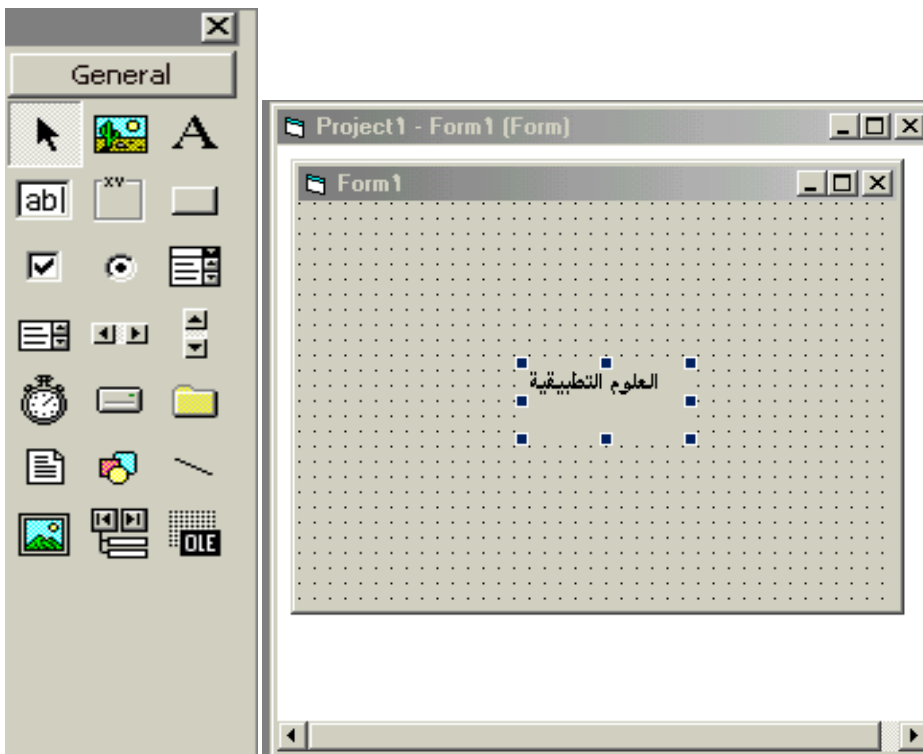
Property name	Objective	Code	Stage of Changing
Caption	String appear on label	label _{no.} .caption= "any name"	Design and run
Autosize	To resize tool to fit text	label _{no.} .autosize= true or false	Design and run
BackColor	Background color for label	label _{no.} .BackColor=Qbcolor(no.)	Design and run
ForeColor	Color of text written on label	label _{no.} .forecolor=Qbcolor(no.)	Design and run
Font	Font style, type and size	Size: label _{no.} .fontsize= no. Style: $font \begin{cases} italic \\ bold \\ underline \end{cases}$ Type: label.FontName = "arial"	Design and run
visible	The label appear or disappear	Label _{no.} .visible= true or false	Design and run
Enabled	The label enable or disable.	label _{no.} . Enabled =true or false	Design and run

Note: The available color numbers that used with QBcolor is the integers 0 to 15 only

Example: Design a form contains label "العلوم التطبيقية" in size 14.

Sol: the properties are:

Label1	
caption	Applied science
fontsize	14



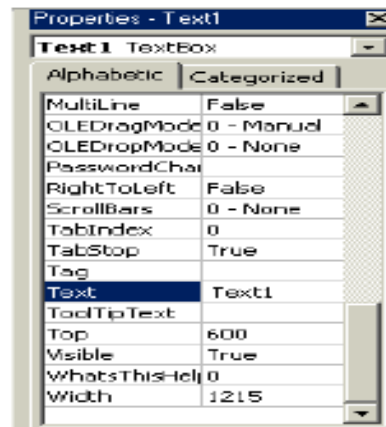
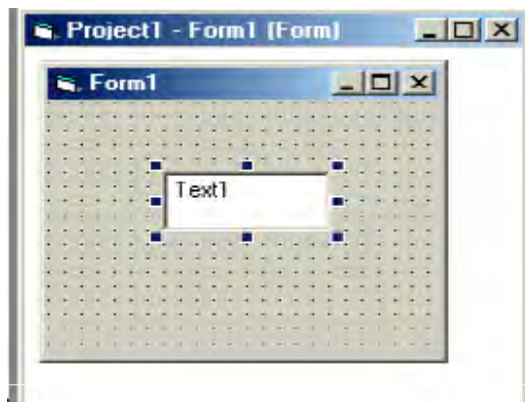


Textbox

The textbox is a box for entering and displaying text (characters or values) in user project. This tool is used frequently in most of the application. The textbox has property window, with no caption, but with space for text. The most important property of this tool is the text content which is described in the following:

Property name	Objective	Code	Stage of Changing
Text	String appear on textbox	text _{no.} .text = "any name"	Design and run
multiline	To enter more than one line	true or false	Design
BackColor	Background color for textbox.	text _{no.} .BackColor=Qbcolor(no.)	Design and run
ForeColor	Color of text written on textbox.	text _{no.} .forecolor=Qbcolor(no.)	Design and run
Font	Font style, type and size.	Size: text _{no.} .fontsize= no. Style: font { <i>italic</i> bold <u>underline</u> Type: label.FontName = "arial"	Design and run
visible	The textbox appear or disappear	text _{no.} .visible= true or false	Design and run
Enabled	The textbox enable or disable.	text _{no.} . Enabled =true or false	Design and run
passwordchar	A row of symbols appear instead of letters	Text _{no.} .passwordchar=(symbol)	Design and run
Setfocus	Put the focus on the specified textbox	Text _{no.} .setfocus	Run

Change text manually: change text property from property window, click inside textbox and add text.

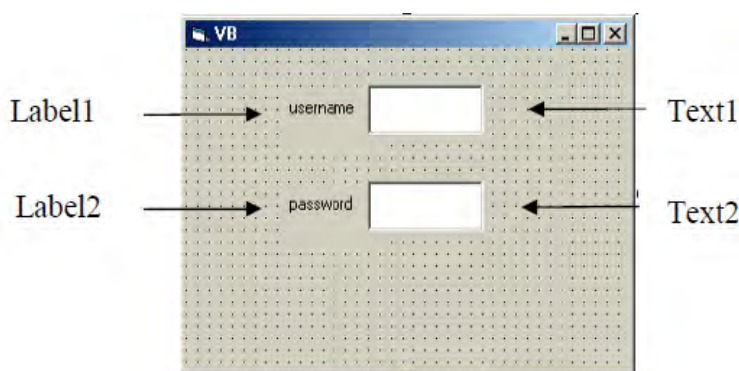


Change text by code:

- 1- Text1.text=""
- 2- Text1.text=" نص"
- 3-Text1.text=text2.text
- 4- Text1.text=label1.caption
- 5- Text1.text = inputbox (" نص")

Example: Design a form to enter username and password such that the title of the form is VB.

Sol: design stage



Form1	
caption	V.B
Text1	
text	
Text2	
Text	
Label1	
caption	username
Label2	
caption	password

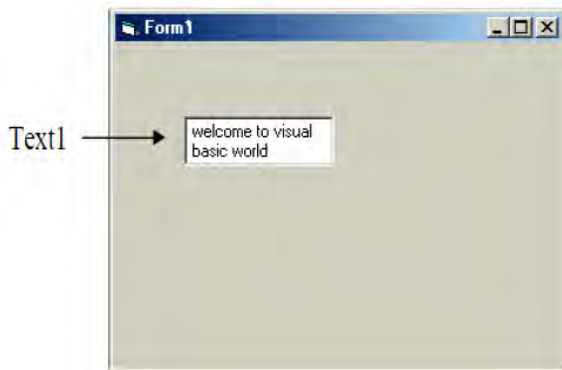
Example: Design a form with one textbox, set the text properties so that this message when project runs (welcome to visual basic world).

Sol: There are two methods:

First method: changing property by code:

```
Private Sub Form_Load()  
Text1.Text = "welcome to visual basic world"  
End Sub
```

Second method: by properties window

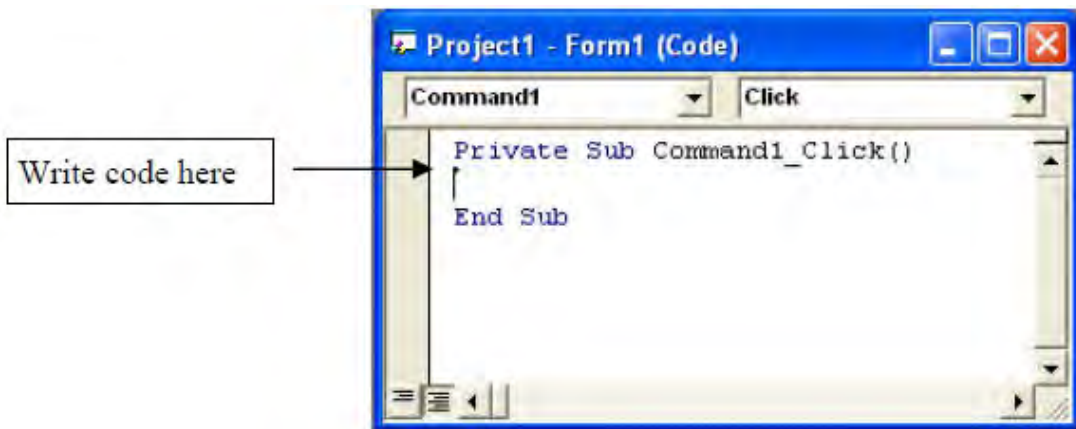


Text1	
text	Welcome to visual basic world

Command button

It acts as a switch. To deal with tool property> click on command button> property window appear> change setting of any desired property. Usually change set its caption property to a suitable string.

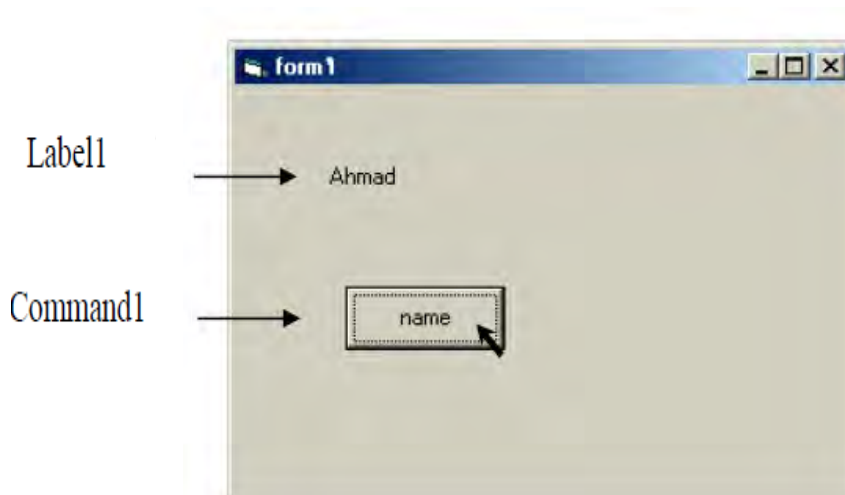
To make the button functional, the user should add some code. To do this: click on command tool> code form appears with click event procedure. Write code in this event or other events like press key event.



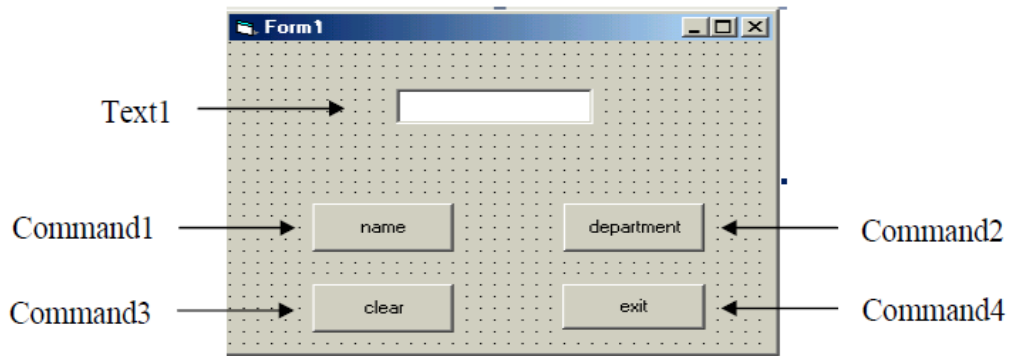
The most familiar properties that are needed for the command button are stated in the table below.

Property name	Objective	Code	Stage of Changing
Caption	String appear on command	<code>command_{no.}.caption="any name"</code>	Design and run
style	Determine the style of command	1-graphical 0-standard	Design
BackColor	Background color for command	<code>command_{no.}.BackColor=Qbcolor(no.)</code>	Design and run
ForeColor	Color of text written on command	<code>command_{no.}.forecolor=Qbcolor(no.)</code>	Design and run
Font	Font style, type and size	Size: <code>command_{no.}.fontsize= no.</code> Style: $font \begin{cases} italic \\ bold \\ underline \end{cases}$ Type: <code>command_{no.}.FontName = "arial"</code>	Design and run
visible	The command appear or disappear	<code>command_{no.}.visible= true or false</code>	Design and run
Enabled	The command enable or disable.	<code>command_{no.}. Enabled =true or false</code>	Design and run

```
Private Sub Command1_Click()
Label1.Caption = "Ahmad"
End Sub
```



Example: Design a form to appear your name and department in textbox, when click on command button "name" and "department" respectively so that you can clear these information's when click on command "clear" and stop project when click on command "exit".



Text1	
text	
Command1	
caption	name
Command2	
caption	department
Command3	
caption	Clear
Command4	
caption	exit

```

Private Sub Command1_Click()
Text1.text="Muna"
End Sub
Private Sub Command2_Click()
Text1.text="Science"
End Sub
Private Sub Command3_Click()
Text1.text=" "
End Sub
Private Sub Command4_Click()
end
End Sub

```

Example: Design a form contains two textbox so that when click on command button "copy" the text copied from first textbox to the second textbox but in size (28).

Sol:

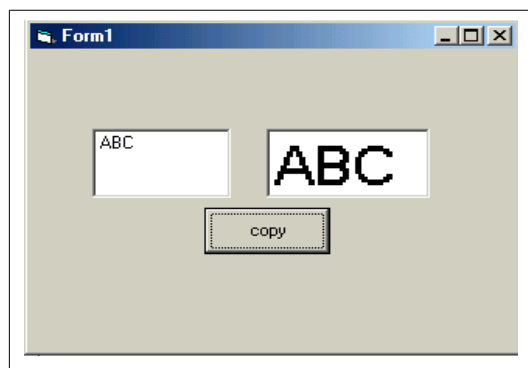
```
Private Sub Command1_Click()  
Text2.Text = Text1.Text  
Text2.FontSize = 28  
End Sub
```

Text1	
text	
Text2	
Text	
Command1	
caption	copy

At run stage this window appear



If the user enter by example the text (ABC) in first textbox and click on command (copy) the same text appear on the second textbox but in size 28.

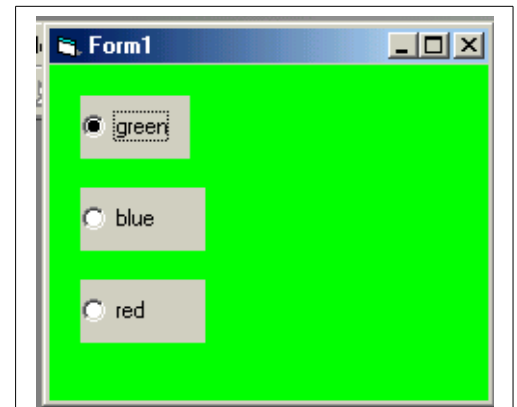


Option button: Used only as a group of buttons. When the user selects one of them the others are deselected automatically.

All other properties of this control are similar to those in form and command button where they are fully discussed which are caption, font, enabled, bgcolor and visible beside an important property which is value that takes true or false and it used with if statement. The option button usually takes click event.

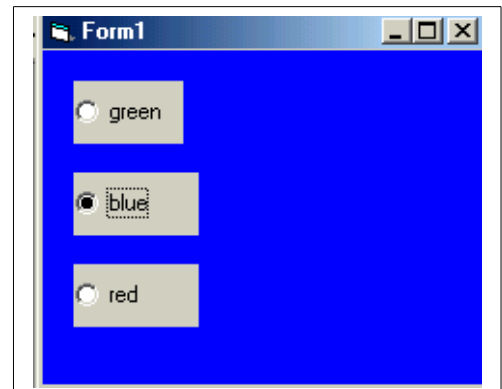
Example: Design a form with three option buttons " red ", " green " and " blue " such that when we click on options the color of the form colored by red, green and blue respectively.

```
Private Sub Option1_Click()  
Form1.BackColor = vbGreen  
End Sub
```

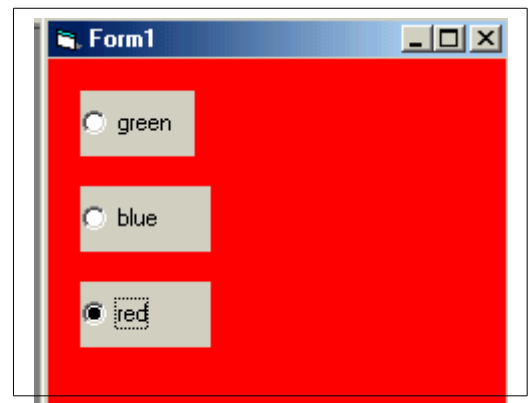


option1:caption	green
option2: caption	blue
option3: caption	red

```
Private Sub Option2_Click()  
Form1.BackColor = vbBlue  
End Sub
```

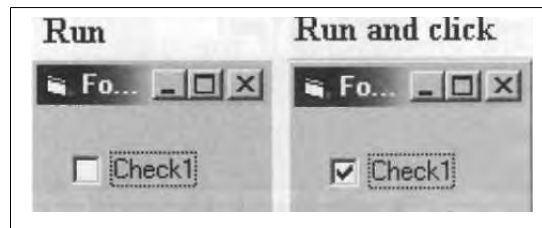


```
Private Sub Option3_Click()  
Form1.BackColor = vbRed  
End Sub
```

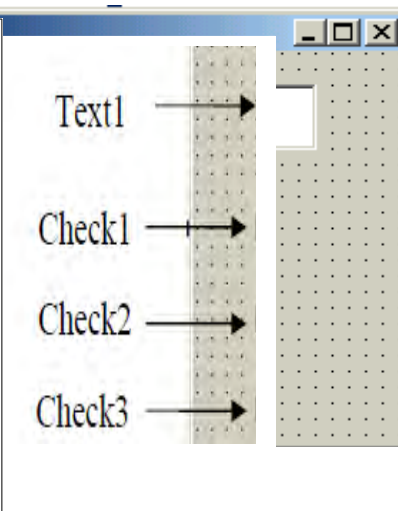


Check box:

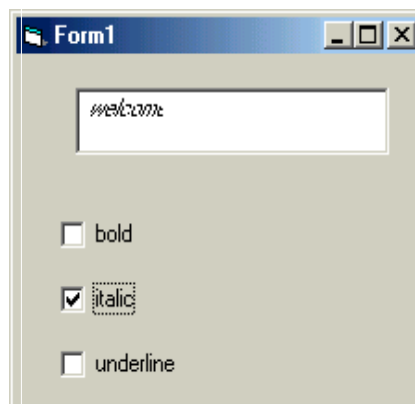
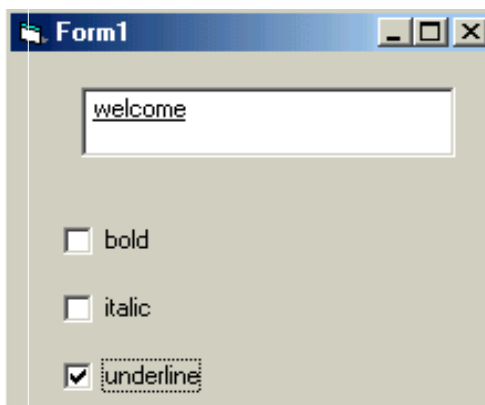
Any number of check boxes can be used on a form. They work independently. Its Property value could be changed in design stage manually, or in running stage by code.



Sol:	
Text1	
Text	•••
Check1	
caption	Bold
Check2	
caption	Italic
Check3	
caption	underline



```
Private Sub Check1_Click()  
Text1.FontBold = Check1.Value  
End Sub  
Private Sub Check2_Click()  
Text1.FontItalic = Check2.Value  
End Sub  
Private Sub Check3_Click()  
Text1.FontUnderline = Check3.Value  
End Sub  
Run stage:
```



Timer

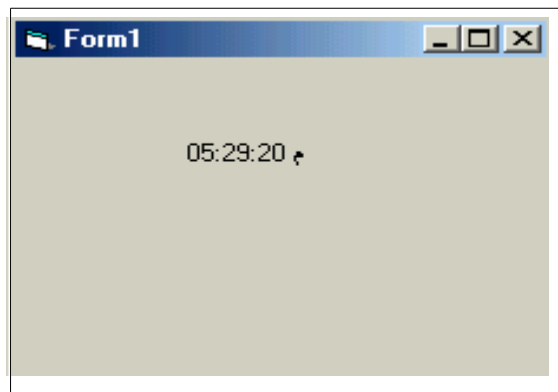
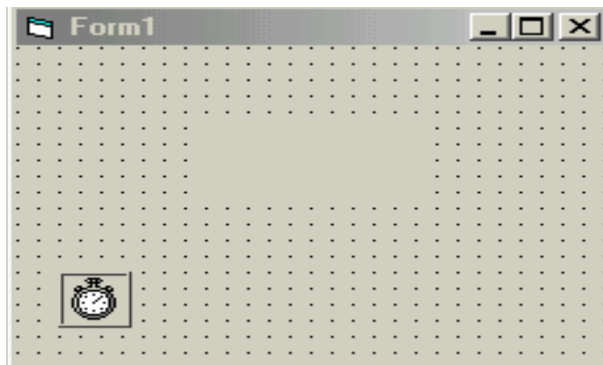
Timer returns the time in millisecond. It may be used to measure execution time of code (program efficiency).

Property name	Objective and code
interval	To repeat the code according to event. It takes an integer values (0-65535) and measured in millisecond
enabled	timer _{no.} . Enabled =true or false

Ex: design electronic clock to display the time in seconds.

sol:

Timer1	
interval	1000
Label1	
Caption	



Example: Design a form to display "applied science" such that when click on command button "start" the color of "applied science" changed randomly every

Sol:

Timer1	
interval	1000
enabled	false
Label1	
Caption	العلوم التطبيقية
Command1	
caption	ابدأ



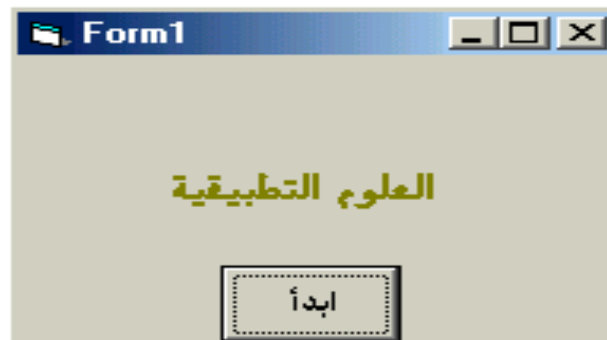
```

Private Sub Command1_Click()
Timer1.Enabled = True
End Sub
Private Sub Timer1_Timer()
t = Rnd * 15
Label1.ForeColor = QBColor(CInt(t))
End Sub

```

Run stage:

When click on command button ابدأ the color of the font will be change every second randomly in integer no. (0-15).



Note: the function (Cint) used to convert to integer no.
And (Rnd) used to generate a random no. in a range (0-1)

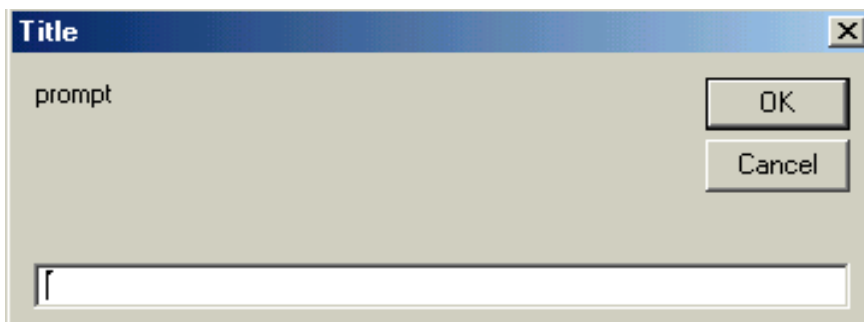
Input - output boxes

There are two types of dialog boxes which are inputbox and messagebox. The first is used to input variable and the second to output variable or message. Both needs code and appear at run time.

a) Inputbox

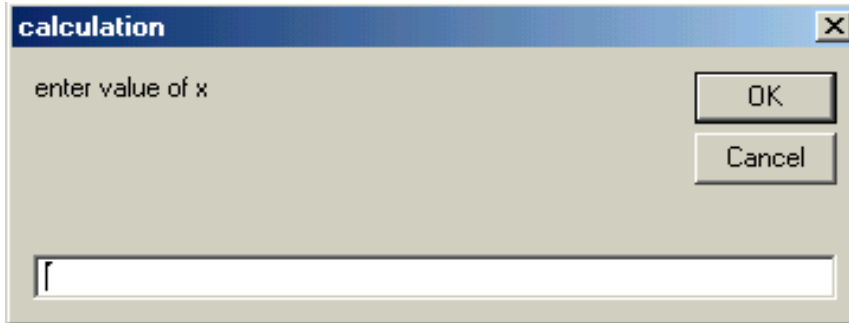
Inputbox used to input value or characters for one variable from keyboard at running stage.

This box needs a code in code sheet and could be written in any event or command
X=inputbox(" prompt or remark", "title")



Example: enter value of x using inputbox

Sol:







```
Private Sub Form_Load()
X=Inputbox("enter value of x", "calculation")
End Sub
```

Message box

It is used to output a message to the user (at running stage) the code needed could be written in code sheet and in any event or command.

The available icons for message box

structure	value	icon
vbcritical	16	
vbquestion	32	
vbexclamation	48	
vbinformation	64	

The available commands for message box

structure	value	Commands
Vbokonly	0	Ok
Vbokcancel	1	Ok, Cancel
vbAbortRetryIgnor	2	Abort, Retry, Ignore
vbYesNoCancel	3	Yes, No, Cancel
vbYesNo	4	Yes, No
vbRetryCancel	5	Retry, Cancel

For example if we write the following statement then a message box will be appear as shown below

```
MsgBox "please close your program", 16, "Error"
```

or

```
MsgBox "please close your program", vbcritical, "Error"
```



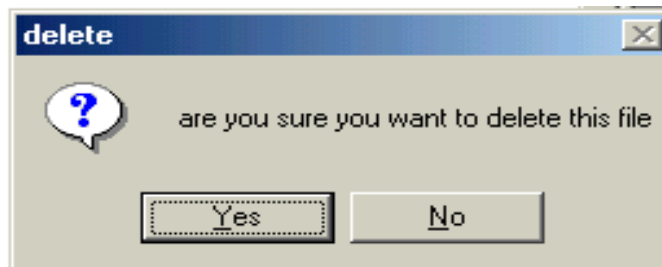
Example: show what appear after running the following statement

```
MsgBox "are you sure you want to delete this file", 32 + 4, "delete"
```

or

```
MsgBox "are you sure you want to delete this file", vbQuestion+vbYesNo,"delete"
```

Sol:



Example: write a program to move the text (excellent) from textbox to message box and change the color of the text after click on command button (display).

Sol:

```
Text1: text="excellent"
```

```
Command1: caption=" عرض "
```

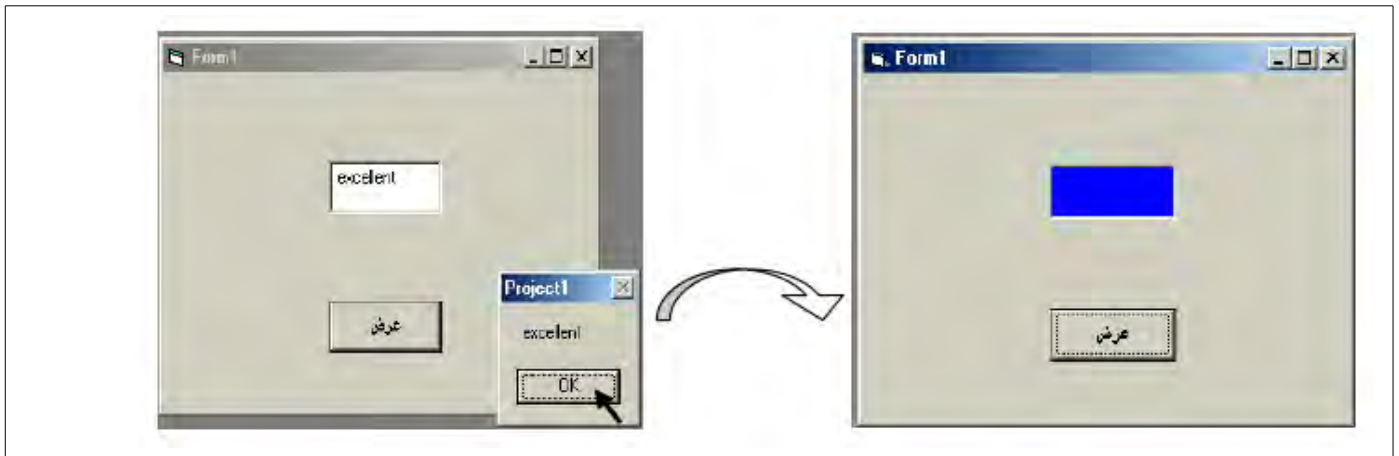
```
Private Sub Command1_Click()
```

```
MsgBox (Text1.Text)
```

```
Text1.BackColor = QBColor(9)
```

```
Text1.Text = " "
```


End Sub



Or we can write the following code:

```
Private Sub Command1_Click()
```

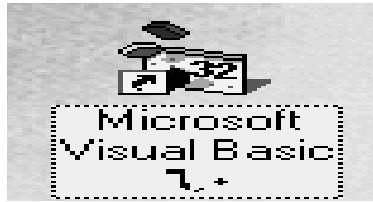
```
X= Text1.Text
```

```
MsgBox (X)
```

```
Text1.BackColor = QBColor(9)
```

```
Text1.Text = " "
```

```
End Sub
```



University of Baghdad
Collage of Education For Pure Science
Ibn Alhythim
Dep.of Computer Science

Windows Programming

Microsoft Visual Basic 6.0

Chapter Two

M.Sc. Lubab Ahmed

Visual basic statements

In visual basic program (code) there are four basic parts, i.e. it contains the following statements:

- 1- Declaration of variables and constants
- 2- Inputting variables
- 3- Operators for variables
- 4- Outputting variables

1- Declaration of a variable and constants

The declaration means defining the data type (variable or constant).

· Variables

A variable is a space in memory filled with data (value, character, time or date).

Notes:

- Variable name must start with character (not number or function) and maximum length 256 character, and does not contain point or symbol.
- Variable name must not repeat for other values.

The variable has to be declared. Variable type is defined by its content. The content may be data as numeric or character or string or Boolean or date, or any type of data (called variant), these types declared as:

Dim variable name **as** type

Or

Global variable name **as** type

Note: The **Dim** declaration written in general part of the form or in any place in form or sub procedure which used for one form. While **Global** declaration used for all forms

The types of variables that are allowed in visual basic are stated in the table below:

Types of variables

Type	Value range	Declaration
1-Integer	-32768<x<32768	Dim x as integer
2-Long	-2.1 e+009<x<2.1 e+009	Dim x as long
3-Single	1.4e-045 < x <3.4e+038	Dim x as single
4-Double	4.9e-324<x<1.79e+308	Dim x as double
5-String	65535 characters	Dim x as string
6-Boolean	True or false	Dim x as Boolean
7-Date	Computer time and date Jan 100<x< 31 Dec 9999	Dim x as date

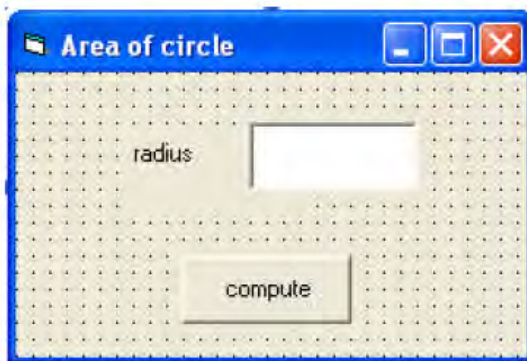
- **Constants**

It is a space in memory filled with fixed value that will not be changed. Constant may be declared as:

Const constant name = value

Example: Declare x as a constant (P), then compute the area of a circle. Put suitable design.

Sol:



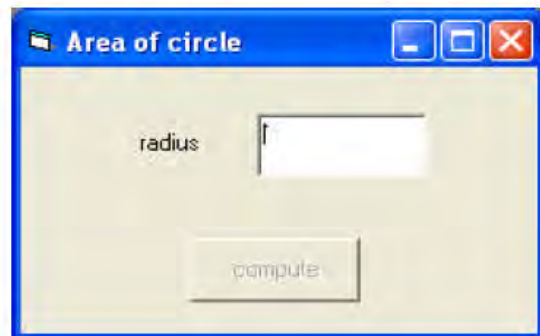
Form1	
caption	Area of a circle
label1	
Caption	radius
Text1	
text	فارغ
Command1	
caption	compute
Enabled	false

code stage:

```
Const p = 3.14159
Dim a, r As Single
```

```
Private Sub Text1_Change()
Command1.Enabled = True
End Sub
```

```
Private Sub Command1_Click()
r = Val (Text1.Text)
a = r ^ 2 * p
MsgBox ("area=" & a)
Text1.Text = " "
Text1.SetFocus
End Sub
```



2- Inputting variables

There are methods to input variable x as stated in the following:

Method of input	For all type of variable
In text tool	X=text _{no} .text
In input box	X=inputbox("prompt","title")

Note: To enter many variables we usually use the second method with loop.

3- Operators for variables

The operators that are used for variable are described in the following table

Arithmetic operators	+	addition
	-	subtraction
	*	multiplication
	/	division
	mod	Modulus –rest of division
	^	exponent
Relational operators	=	equal
	<	Less than
	<=	Less or equal
	>	Greater than
	>=	Greater or equal
	<>	Not equal

Note: The order of operations when executing arithmetic operation is:

Exponentiation - multiplication division and mod - finally addition and subtraction.

Assignment statement

There are many statements ways to fill a variable as follows:

Variable = expression

Expression may include variables, operations and functions as follows:

- 1- Numerical variable. For example: i=3
- 2- Mathematical relation. For example: x=a/b
- 3- Characters variable (string). For example: t="abc"
- 4- Boolean variable (logical). For example: p=true

Functions for variables

The numeric and string variables are the most common used variables in programming, therefore V.B provides the user with many functions to be used with a variable to perform certain operations or type convention. The most common functions for numerical variable x

Function	Description
Abs(x)	Absolute of x
Sqr(x)	Square root of x
Int(x)	Integer of x
Len(x)	Number of character of variable x
Lcase(x)	Change the text x to small letters
Ucase(x)	Change the text x to capital letters
Cstr(x)	Convert variable x to string
Val(x)	Convert string x to numerical variable

4- Outputting variables

There are methods to output variable x as stated in the following:

Method of output	For all type of variable
On form	Print x Note: in load event we must use the statement: (form1.show)
to text tool	text _{no} .text =X
to label tool	Label _{no} .caption=x
By message box	msgbox (x) Or msgbox ("remark"& x)

The instruction print could be very helpful to display data and used as follows:

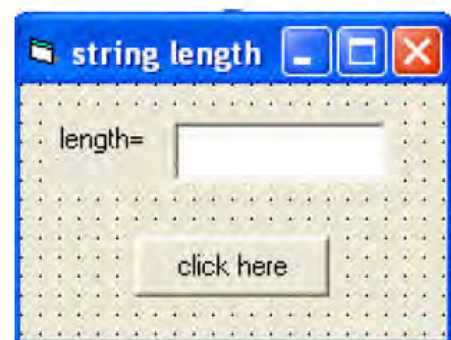
Code	Description	example
print	To leave one line and print on next	
Print "a", "b", "c"	Use (,) to print a distance between outputs	a b c
Print "a"; "b"; "c"	Use (;) to print the outputs adjacent	abc
Print "a","b"; Print "c"	Print a, b then print c on the same line	abc

Example1: write a program to enter any text and compute its length. Put suitable design.

Sol:

Design stage:

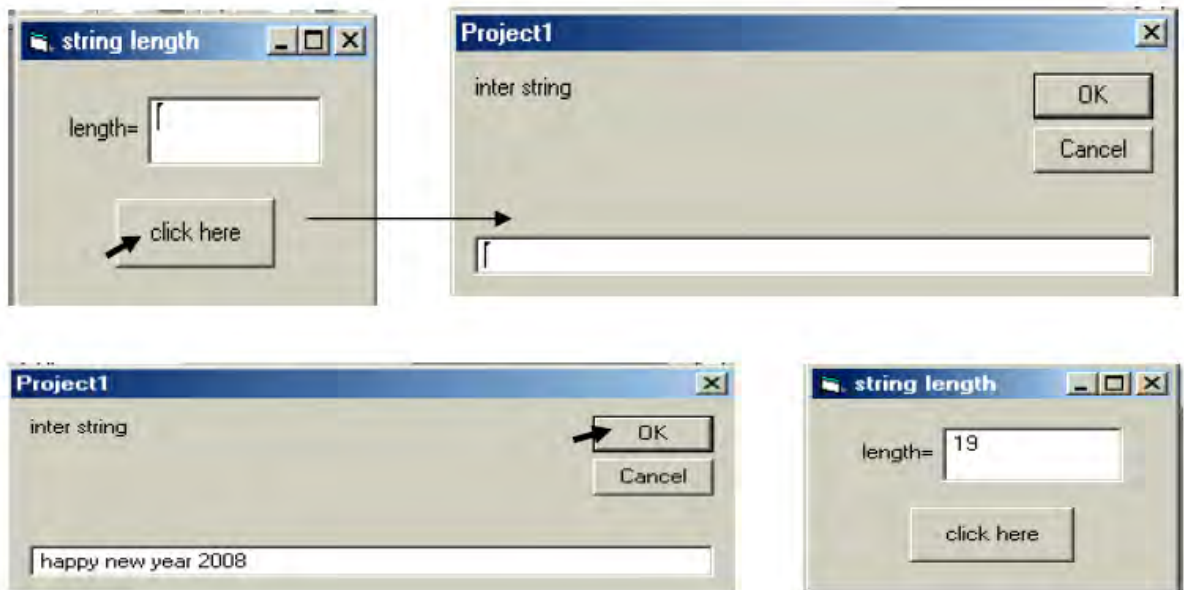
Form1	
Caption	String length
Command1	
caption	Click here
Label1	
Caption	Length=
Text1	
Text	



Code stage:

```
Dim s As String  
Private Sub Command1_Click()  
s = InputBox("inter string")  
L = Len(s)  
Text1.Text = CStr(L)  
End Sub
```

Running stage:



Conditional statements

There are two types of conditional statements:

- 1- If statement
- 2- Select case

1- **If** statement: The comparison operations are used with conditional statements.

The comparison operations are: (<, <=, >, >=, =, <>, and, or)

There are four structures for if statement.

a) Simple structure **If.. then**:

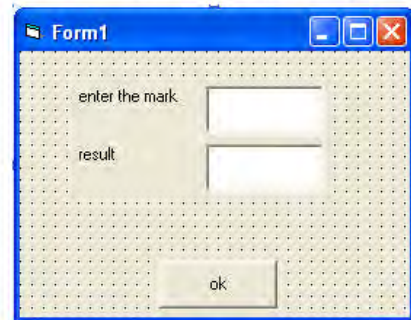
Used for running one programming statement only if the required condition satisfied.

The general form is: **If** condition **then** statement

Example 1: write a program to enter a mark of a student then print (pass) if he successful.

Sol:

```
Dim x as integer
Private sub command1_click()
X= cint(text1.text)
If x>= 50 then text2.text= "pass"
End sub
```



b) **If block** structure: Used for running many programming statements if the required condition satisfied.

The general form is:

If condition **then**

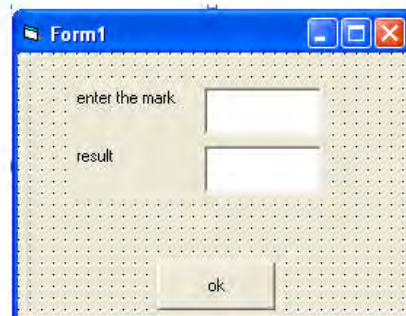
Statements

End if

Example 2: write a program to enter a mark of a student then print (pass) in size 18 if he successful.

Sol:

```
Dim x as integer
Private sub command1_click()
X= cint(text1.text)
If x>= 50 then
text2.text= "pass"
text2.fontsize=18
end if
End sub
```



c) **If.. Then.. Else** structure: Used for running many programming statements if the required condition satisfied. And running another programming statements (after else) if the required condition not satisfied.

The general form is:

If condition **then**

Statements

Else

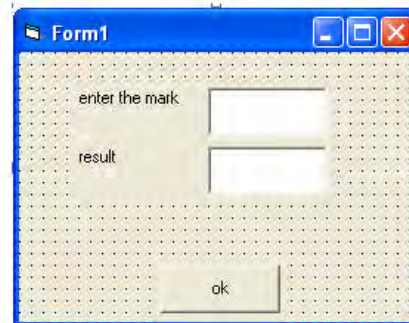
Statements

End if

Example3: write a program to enter a mark of a student then print (pass) if he successful and print (fail) otherwise.

Sol:

```
Dim x As Integer
Private Sub command1_click()
x = CInt(Text1.Text)
If x >= 50 Then
Text2.Text = "pass"
Else
Text2.Text = "fail"
End If
End Sub
```



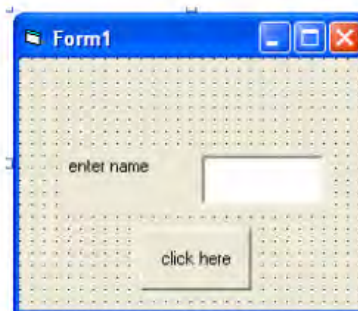
d) **If.. Then.. Elseif.. Else** structure:

Used if we have many conditions to be satisfied

Example 4: write a program to enter a user name and display the message (hello) three times. The first one for (Muna), the second one for (Maha) and the third for any user as a guest.

Sol:

```
Dim x As String
Private Sub command1_click()
x = Text1.Text
If x = "Muna" Then
MsgBox "hello, Muna"
ElseIf x = "Maha" Then
MsgBox "hello,Maha"
Else
MsgBox "hello, guest"
End If
End Sub
```



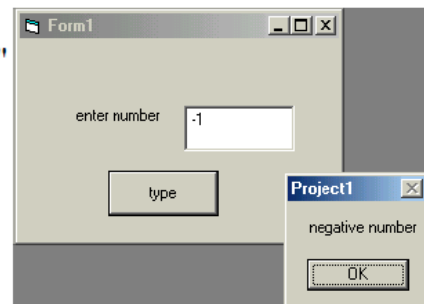
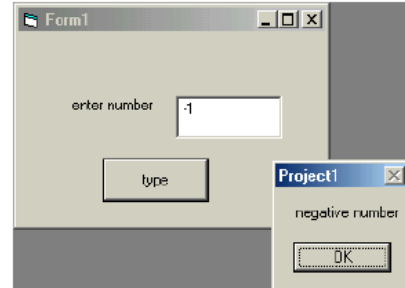
Example 5: Write a program to classify any entered number according to its sign and display the phrase (negative number) when the number is negative and the phrase (positive number) when the number is positive, otherwise display the phrase (neither positive nor negative).

Sol:

```

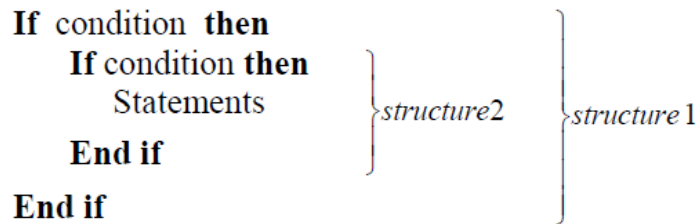
Dim x As Single
Private Sub command1_click ()
x = Val(Text1.Text)
If x > 0 Then
MsgBox "positive number"
ElseIf x < 0 Then
MsgBox "negative number"
Else
MsgBox "neither positive nor negative"
End If
End Sub

```



Nested If statement:

It can be takes the following structure:



Note: Any structure of if structures can be used instead of structure 1 and 2 above.

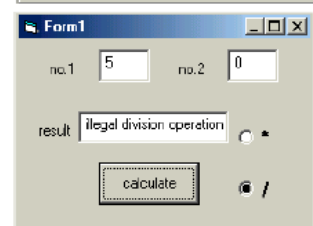
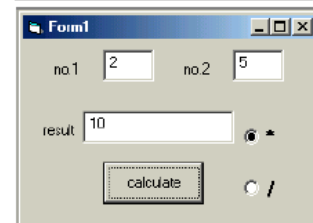
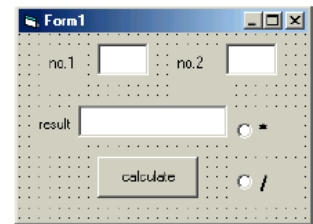
Example 6: Write a program to enter two numbers and compute multiplication and division operations using option button with display the phrase (illegal division operation) when the denominator is zero.

Sol:

```

Dim a, b, c As Single
Private Sub command1_click()
a = Val(Text1.Text)
b = Val(Text2.Text)
If Option1.Value Then
c = a * b
Text3.Text = CStr(c)
Else

```



```

If b <> 0 Then
    c = a / b
    Text3.Text = CStr(c)
Else
    Text3.Text = "illegal division operation"
End If
End If
End Sub

```

Select statement

Used for applying many statements depending on one variable. The general form is:

```

Select case variable
Case value1
statements
Case value2
Statements
.
.
.
Case value n
Statements
Case else
Statements
End select

```

Example 7: write a program to print the days of the week when we enter its number

Sol:

```

Dim x As Integer
Private Sub Command1_Click()
x = CInt(Text1.Text)
Select Case x
Case 1
MsgBox ("Sunday")
Case 2
MsgBox ("Monday")
Case 3
MsgBox ("Tuesday")
Case 4
MsgBox ("Thursday")
Case 5
MsgBox ("Wednesday")

```



```

Case 6
MsgBox ("Friday")
Case 7
MsgBox ("Saturday")
End Select
End Sub

```

Example 8: write a program to give the evaluation for different marks as follows:

mark	evaluation
90-100	Excellent
80-89	Very good
70-79	Good
60-69	Medium
50-59	Pass
0-49	Fail



```

Dim x As Integer
Private Sub Command1_Click()
x = CInt(Text1.Text)
Select Case x
Case 90 To 100
MsgBox ("excellent")
Case 80 To 89
MsgBox ("very good")
Case 70 To 79
MsgBox ("good")
Case 60 To 69
MsgBox ("medium")
Case 50 To 59
MsgBox ("pass")
Case 0 To 49
MsgBox ("fail")
Case Else
MsgBox "the range is 0-100", vbCritical, "error"
End Select
End Sub

```



Loop statement:

Visual basic supports statement to perform loops. The loops statements could have different structures as follows:

- 1- Counter loop.
- 2- Conditional loop.

1- Counter loop:

Loops apply programming statements for fixed number of times using counter (for... next) statement.

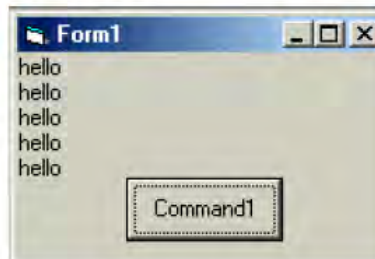
The general form is:

```
For variable = start value to end value step step value
Statements
Next variable
```

Example1: Write a program to print (hello) five times.

Sol:

```
Dim i as integer
Private Sub Command1_Click ()
For i = 1 To 5
Print "hello"
Next i
End Sub
```



Example2: Write a program to print even numbers from 1 to 10.

Sol:

```
Dim i as integer
Private Sub Command1_Click ()
For i = 2 To 10 step 2
Print i
Next i
End Sub
```



Notes:

- 1-The variable's value that we use as counter must be integer value (integer, long).
- 2- If we don't determined the step value then the assumed value is 1.
- 3- If the final value smallest than the initial value, then the step value must be negative.

2- Conditional loop

Loops repeat programming statements according to specific condition. There are two types of conditional loop:

- 1- Do while
- 2- Do until

1-Do while loop: In this loop the statements will be implemented and repeated when ever the condition satisfied. The general form is:

Do while condition
Statements
Loop

Example3: Write a program to print (hello) five times with its numbering using do while loop.

Sol:

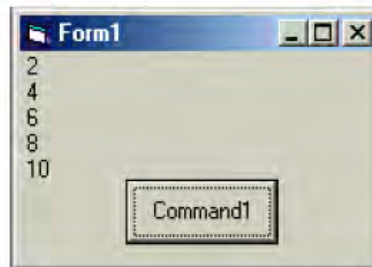
```
Dim i as integer
Private Sub Command1_Click ()
i = 1
Do while i <= 5
Print "hello"; i
i = i + 1
Loop
End Sub
```



Example4: Write a program to print even numbers from 1 to 10.

Sol:

```
Dim i as integer
Private Sub Command1_Click ()
i = 2
Do while i <= 10
Print i
i = i + 2
Loop
End Sub
```



2-Do until loop: In this loop the statements will be implemented and repeated when ever the condition not satisfied, (i.e) the loop will be stopped when the condition satisfied. The general form is:

Do until condition
Statements
Loop

Example5: Write a program to print (hello) five times with its numbering using do until loop.

Sol:

```
Dim i as integer
Private Sub Command1_Click ()
i = 1
Do until i > 5
Print "hello"; i
i = i + 1
Loop
End Sub
```


Example6: Write a program to find the summation of undetermined number of positive numbers such that the program will be stopped when we enter negative number.

Sol:

```

Dim x, sum As Single
Private Sub command1_click()
sum = 0
x = Val(InputBox("enter x", "summation"))
Do While x >= 0
sum = sum + x
x = Val(InputBox("enter x", "summation "))
Loop
MsgBox (CStr(sum))
End Sub

```



Example7: Write a program to find the summation of the numbers from 5 to 15.

Sol:

```

Dim I, sum as integer
Private Sub command1_click ()
sum = 0
For i = 5 to 15
Sum = sum + i
Next i
Label1.caption = "sum ="&cstr(sum)
End Sub

```

Example8: Write a program to find the summation of 10 numbers.

Sol:

```

Dim i as integer
Dim x, sum as double
Private Sub command1_click ()
sum = 0
For i = 1 to 10
x = val(inputbox ("enter number"))
Sum = sum + x
Next i
Label1.caption = "sum="& cstr(sum)
End Sub

```



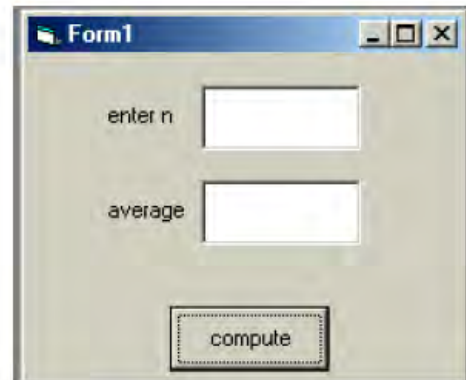
Running stage:

For example if we entered the numbers: 1, 5, -1, 3, 2, 0, -1, 3, 0, -4 then sum=8

Example9: Write a program to find the average of n numbers.

Sol:

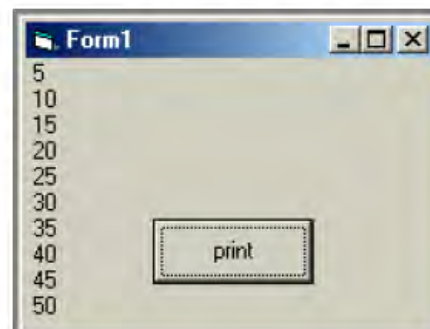
```
Dim i as integer
Dim x, sum, av as Double
Private Sub command1_click ()
i = 1: sum = 0
n = cint (text1.text)
Do while i <= n
x = val(inputbox ("enter number"))
Sum = sum + x
i = i + 1
Loop
Av = sum/n
Text2.text = cstr(av)
End Sub
```



Example10: Write a program to print multipliers of 5 (from 5 to 50)

Sol:

```
Dim i as integer
Private Sub Command1_Click ()
i = 5
Do until i > 50
Print i
i = i + 5
Loop
End Sub
```



Example11: write a program to find the average of numbers that dividable by 3 (with out remainder) from 3 to 99.

Sol:

```
Dim I, n, sum as integer
Dim av as Double
Private Sub command1_click ()
i = 3 : n = 0
sum = 0
Do while i <= 99
Sum = sum + i
i = i + 3
n = n + 1
Loop
Av = sum/n
Print "av ="; av
End Sub
```

Nested loop:

The nested loops are the loops that are placed inside each other. The most inner loop will be executed first, then the outer ones. These loops should neither intersect, nor have the same index. As follows:

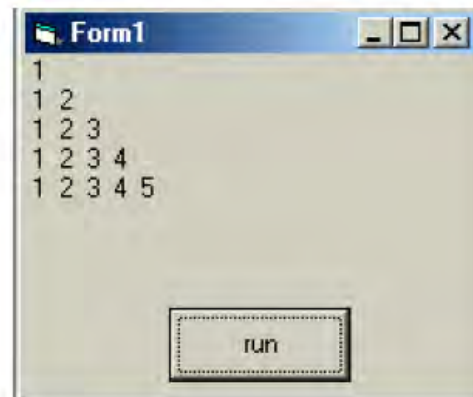
```
For i = 1 To n
  For j = 1 To m
    Statements
  Next j
Next i
```

Example15: write a program to generate the numbers in the following form.

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Sol:

```
Dim I, j As Integer
Private Sub command1_click()
  For I = 1 To 5
    For j = 1 To i
      Print j;
    Next j
    Print
  Next I
End Sub
```




Menu

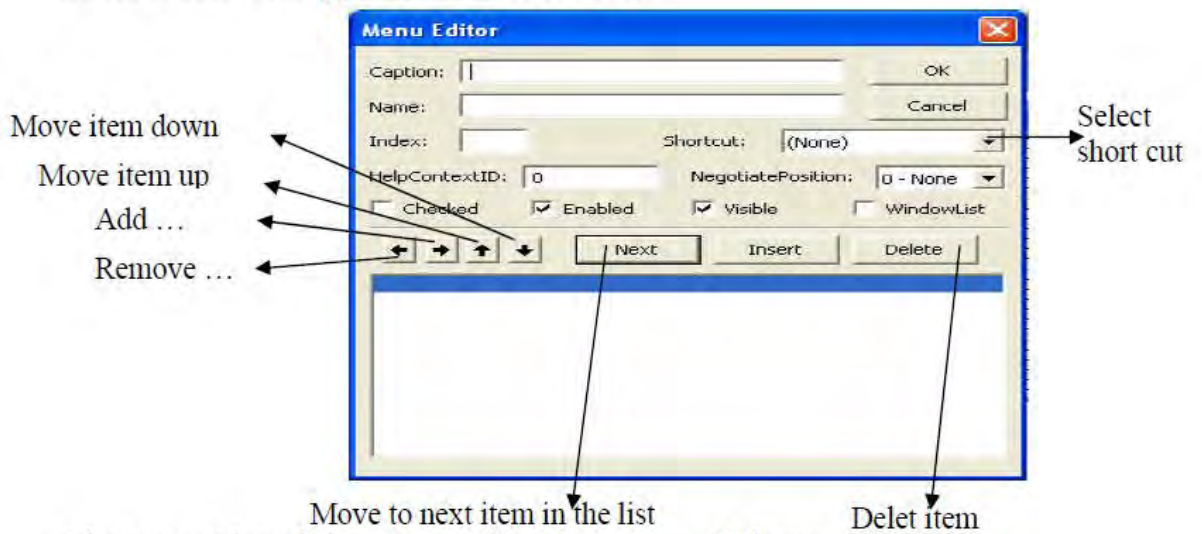
The menu is a bar at the top of the form. The standard form is display without menu, but the user can add it. This menu could be included in form using menu editor. In next section the menu editor and the required code will be discussed.

Menu Editor



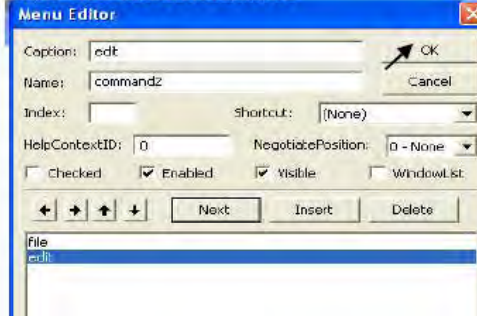

To use menu there are three ways:


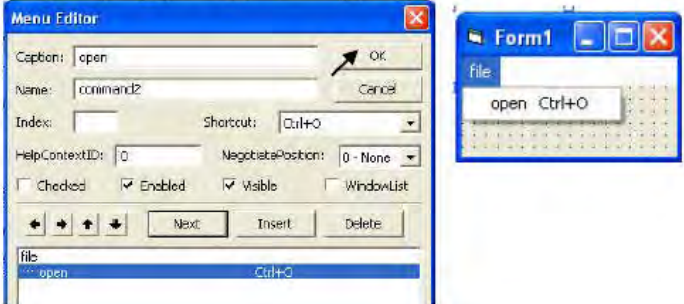
- 1- Press menu icon from toolbar .
- 2- press (ctrl+E) from key board.
- 3- click on: tools>Menu Editor.

Menu editor box appears as shown below.



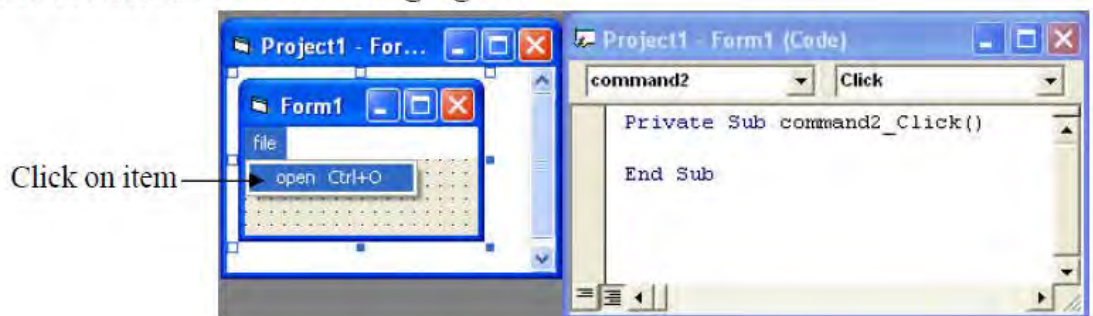
The steps of applying menu editor box are as follows:

<p>To add item</p>	<p>Use menu editor as follows</p>  
<p>To add many items</p>	<p>Use the menu Editor</p>  

Create sub Menu list	<p>Use menu Editor</p> 
Create Shortcut	<p>Use & with caption and select shortcut</p> 

Code for menu items

Each item in menu or sub menu is considered as a command which takes the event click only. The user can add code for each item: click on item>code for that item appears on code sheet. Also code can be added to form: click on item >code for that item appears on code sheet. This is described in the following figure:



Example: Design a form with menu and a label with a specific title. The menu contains one item color with sub menu items: red, green, blue and exit, to color the label in red, green, blue then exit from the program.
Sol: put label1 with any caption for example (hello)

Caption: color Name: command1	Create standard menu (color) from menu editor>next
Caption: red Name:command2	Add sub menu items by pressing → then enter the caption and name>next.

Caption: green Name:command3	Do the same thinks for other items>ok
Caption: blue Name:command4	
Caption: exit Name:command5 Shortcut: ctrl+E	



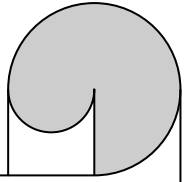
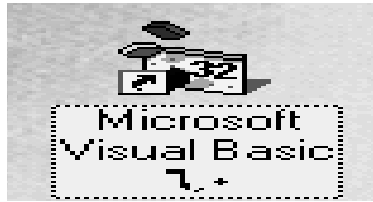
To programming these commands click on each one to open its code window and write the following code:

```
Private sub command2_click ()
Label1.backcolor=vbred
End Sub
```

```
Private sub command3_click ()
Label1.backcolor=vbgreen
End Sub
```

```
Private sub command4_click ()
Label1.backcolor=vbblue
End Sub
```

```
Private sub command5_click ()
End
End Sub
```



University of Baghdad

Collage of Education For Pure Science

Ibn Alhythim

Dep.of Computer Science

Windows Programming

**Microsoft
Visual Basic
6.0**

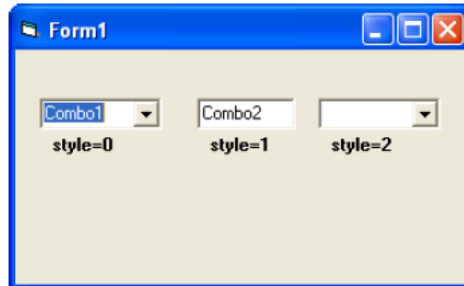
Chapter Three

M.Sc. Lubab Ahmed

5-5 Using ListBox and ComboBox Controls In Visual Basic 6:

The ListBox will display a single column of strings, referred to as **items**. The items to appear initially can either be specified at design time with the List property or set with code in the Form_Load event procedure. Then code is used to access, add, or delete items from the list. If the number of items exceed the value that be displayed, scroll bars will automatically appear on the control. These scroll bars can be scrolled up and down or left to right through the list.

A ComboBox is best through of as a text box with a help list attached. With an ordinary textbox, the user must type information into the box. With a combobox, the user has the option of either typing in information or just selecting the appropriate piece of information from a list. The two most useful types of combobox are denoted as style property combobox as shown in Figure below.



With a style 1 combo box, the list is always visible. With style 0 or 2 combobox, the list drops down when the user clicks on the arrow. In either case, when an item from the list is highlighted, the item automatically appears in the text box at the top and its value is assigned to the text property of the combo box. The items to appear initially can either be specified at design time with the combo property or set with `combo_change()` event procedure directly.

ComboBoxes have essentially the same properties, event and methods as ListBoxes.

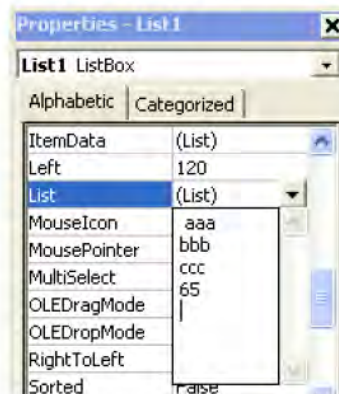
The following Figure lists some of the common **ListBox & ComboBox** properties and methods.

Property	Description
Properties	
Enabled	By setting this property to True or False user can decide whether user can interact with this control or not
List	String array. Contains the strings displayed in the drop-down list. Starting array index is 0. Use CTRL+Enter to insert values.
Sorted	Boolean. Specifies whether the ListBox & ComboBox items are sorted or not.
Style	Integer. Specifies the style of the ListBox & ComboBox appearance
Text	String. Specifies the selected item in the ComboBox.
Visible	Boolean. Specifies whether ListBox & ComboBox is visible or not at run time
Event Procedures	
Change	Called when text in ComboBox is changed
Click	Called when the ListBox & ComboBox is clicked

Methods	Description	Example
AddItem	Add an item to the ListBox &ComboBox	List1.additem str (x):Combo1.additem str (x)
ListCount	Integer. Contains the number of drop-down list items	X=List1.listcount: Y=Combo1.listcount
ListIndex	Integer. Contains the index of the selected ListBox &ComboBox item. If an item is not selected, ListIndex is -1	X=List1.ListIndex : Y=Combo1.ListIndex
List	String array. Contains the strings displayed in the drop-down list. Starting array index is 0.	X=List1.List(1): Y=Combo1.List(4) X=List1.List(List1.ListIndex)
Text	String. Specifies the selected item in the ListBox &ComboBox.	X=List1.Text: Y=Combo1.Text
Clear	Removes all items from the ListBox &ComboBox	List1.Clear: Combo1.Clear
RemoveItem	Removes the specified item from the ListBox &ComboBox	List1.RemoveItem 1: Combo1.RemoveItem 5 List1.RemoveItem List1.ListIndex
NewIndex	Integer. Index of the last item added to the ListBox &ComboBox. If the ComboBox does not contain any items , NewIndex is -1	X=list1.NewIndex: Y= Combo1.NewIndex

5-5.1 Adding items to a ListBox &ComboBox : It is possible to populate the list at design time or run time

- **Design Time :** To add items to a list at design time, click on List property in the property box and then add the items. Press CTRL+ENTER after adding each item as shown below.

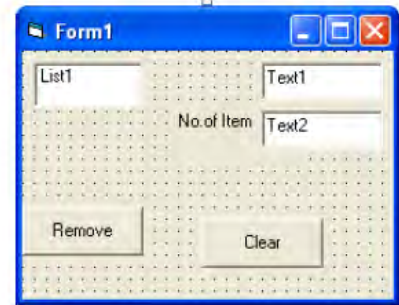


Example 5-9: Design a form with one list box, two textboxes and two command buttons. Write code for the following-events.

- 1- Form_Load event, to add items.(5)
- 2- In click event of listbox,to add item to text1 from list box if item is selected
- 3- In click event of command1 (Remove), to remove item from list box if item is selected and display the number of items in the listbox into text2.
- 4- In click event of command2 (Clear), to clear items from list box.

Solution:

```
Private Sub Form_Load()  
Dim I  
For I=0 To 4  
List1.AddItem InputBox("")  
Next  
Text2.Text=List1.ListCount  
EndSub
```



```
Private Sub List1_Click()  
Text1.text=List1.Text           or   Text1.Text=List1List(List1.ListIndex).  
EndSub
```

```
Private Sub Command1_Click()  
List1.RemoveItem List1.ListIndex  
EndSub
```

```
Private Sub Command2_Click()  
List1.Clear  
EndSub
```

Example 5-10: Design a form with two list boxes, one label, and one command buttons. Write code for the following-events.

- 1- Form_Load event, to add items (n=100) to list1.
- 2- In click event of command1 (Sum), to Sum items from list1 and add to list2 at each step. Exit loop if (Sum=120).
- 3- In click event of list2, to display number of items to label1

Solution:

```
Private Sub Form_Load()  
Dim I  
For I=1 To 100  
List1.AddItem Str(I)  
Next I
```

```
Private Sub Command1_click()  
Dim I, Sum  
For I=1 to 100  
Sum=Sum+I           or   Sum=Sum + list1.list (I-1)  
List2.AddItem Str(Sum)  
If Sum=120 Then Exit For  
Next  
End sub
```

```

Private Sub List2_Click()
Label1.caption=List2.listcount
End Sub

```

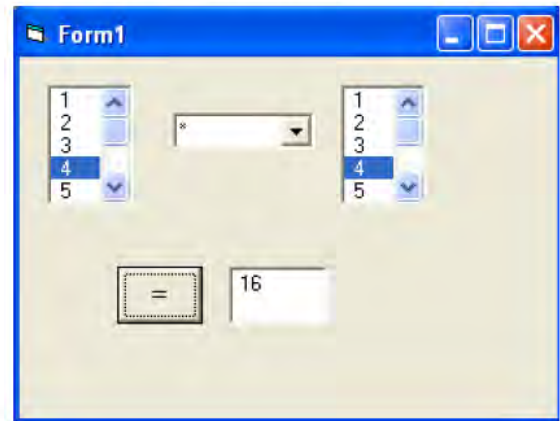
Example 5-11: Write a code program to design a simple calculator. This program uses a combo box which includes four operators, addition, subtraction, multiplication and division and two list boxes included the integer numbers. It can perform the above four basic calculations by changing the operators.

Solution:

```

Private Sub Form_Load()
Dim I
For I=1 To 10
List1.AddItem Str(I)
List1.AddItem Str(I)
Next I
Combo1.AddItem "+"
Combo1.AddItem "-"
Combo1.AddItem "*"
Combo1.AddItem "/"
Private Sub Command1_Click()
Dim z
Select Case Combo1.Text
Case "+"
z = Val(List1.Text) + Val(List2.Text)
Case "-"
z = Val(List1.Text) - Val(List2.Text)
Case "*"
z = Val(List1.Text) * Val(List2.Text)
Case "/"
z = Val(List1.Text) / Val(List2.Text)
Case Else
Exit Sub
End Select
Text1.Text = Str(z)
End Sub

```



6- Arrays in Visual Basic 6

An array is a collection of simple variables of the same type to which the computer can efficiently assign a list of values. Array variables have the same kinds of names as simple variables. An array is a consecutive group of memory locations that all have the same name and the same type. To refer to a particular location or element in the array, we specify the array name and the array element position number. The individual elements of an array are identified using an index. Arrays have upper and lower bounds and the elements have to lie within those bounds. Each index number in an array is allocated individual memory space and therefore users must evade declaring arrays of larger size than required. We can declare an array of any of the basic data types including variant, user-defined types and object variables. The individual elements of an array are all of the same data type.

6-1 Declaring arrays: Arrays may be declared as Public (in a code module), module or local. Module arrays are declared in the general declarations using keyword Dim or Private. Local arrays are declared in a procedure using Dim. Array must be declared explicitly with keyword "As".

There are two types of arrays in Visual Basic namely:

- **6-1-1 Fixed-Size Array:** The size of array always remains the same-size doesn't change during the program execution. When an upper bound is specified in the declaration, a Fixed-array is created. The upper limit should always be within the range of long data type.

One Dimension Array:

Declaring a fixed-array, if array-Name is the name of an array variable and N is a whole number, then the statement

Dim ArrayName (N) As Var Type

Where, the Dim statement is said to dimension the array and (N) is the range of the array. The array holds either all string values or all numeric values, depending on whether *Var Type* is string or one of the numeric type names.

For example:

Dim Num (5) As Integer

In the above illustration, num is the name of the array, and the number 6 included in the parentheses is the upper limit of the array. The above declaration creates an array with 6 elements, with index numbers running from 0 to 5.

The numbers inside the parentheses of the individual variables are called **subscripts**, and each individual variable is called a **subscripted variable** or **element**. The elements of an array are assigned successive memory locations. The following figure shows the memory location for the array Num(5)

Num (5)	Num (0)	Num (1)	Num (2)	Num (3)	Num (4)	Num (5)
	1	3	-10	5	3	2

If we want to specify the lower limit, then the parentheses should include both the lower and upper limit along with the To keyword. An example for this is given below.

Dim Num (6) As Integer

Num (6)	Num (0)	Num (1)	Num (2)	Num (3)	Num (4)	Num (5)	Num(6)
	-	1	3	-10	5	3	2

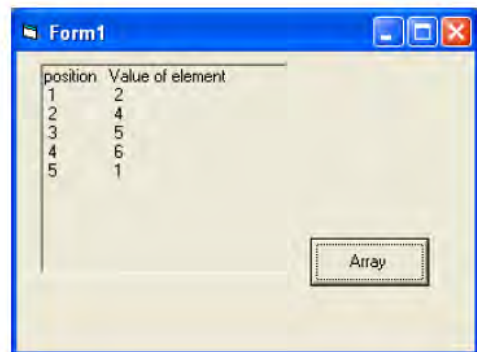
In the above statement an array of 10 elements (Num(10)) is declared but with indexes running from 1 to 6.

Example 6-1: Write a code program to read of one dimensional array A(5). Print the value and position of each element.

$$A = \begin{bmatrix} 2 \\ 4 \\ 5 \\ 6 \\ 1 \end{bmatrix}$$

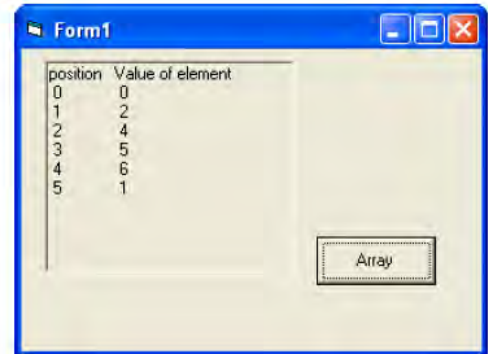
Solution 1:

```
Dim A(5) as single
Picture1.cls
Picture1.Print "position"; Space (3); "Value of element"
For I=1 To 5
A(I)= Val(InputBox(""))
Next I
For I= 1 to 5
Picture1.Print I; Space (11); A(I)
Next
```

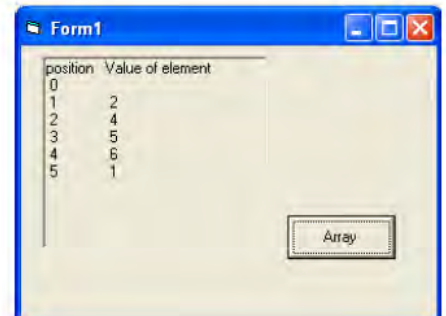


Solution 2:

```
Dim A(5) as single
Picture1.cls
Picture1.Print "position"; Space (3); "Value of element"
For I= 1 To 5
A(I)= Val(InputBox(""))
Next I
For I= 0 to 5
Picture1.Print I; Space (11); A(I)
Next
```



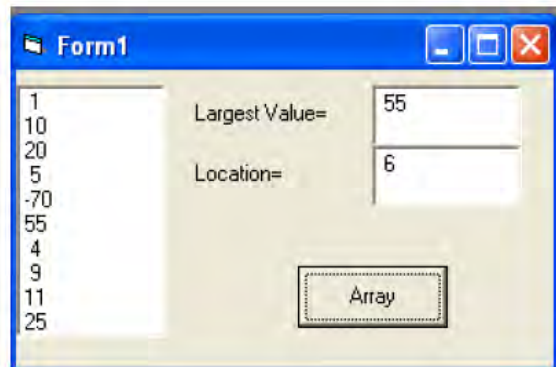
Note: In solution 2, The type value of array (A) as single, then the default value of A(0)=0.If Type value of array(A) as Variant, then empty value in position A(0).



Example 6-2: Suppose A is a one dimension array with 10 elements is entered into listbox. Write a program segment to find the location J such that A (J) contains the largest value in A. Display the Largest value and the location into textboxes.

Solution:

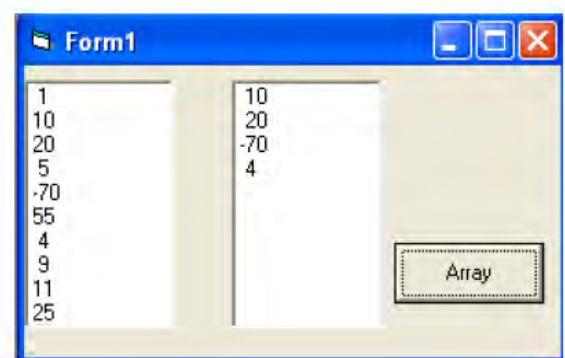
```
Dim A(10) as single
For I=1 To 10
A(I)=Val(list1.list(I-1))
Next
Max=A(1) : P=1
For J=1 to 10
If A(J)> Max Then
Max=A(J) : P= J
EndIf
Next
Text1.text= Str(Max)
Text2.text= Str(P)
End Sub
```



Example 6-3: Suppose A is a one dimension array with 10 elements is entered into listbox. Write a program segment to create the one dimension array (B) contains the even value in array (A). Display the new array (B) into list box2.

Solution:

```
List2.Clear
Dim A(10) As Single, B(10) As Single
For I = 1 To 10
A(I) = Val(List1.List(I - 1))
Next
For I = 1 To 10
If A(I) Mod 2 = 0 Then
k = k + 1
B(k) = A(I)
End If
Next
For I = 1 To k
List2.AddItem Str(B(I))
Next
End Sub
```



Two Dimensional Arrays:

Arrays can have multiple dimensions. A common use of multidimensional arrays is to represent tables of values consisting of information arranged in rows and columns. To identify a particular table element, we must specify two indexes: The first (by convention) identifies the element's row and the second (by convention) identifies the element's column.

Tables or arrays that require two indexes to identify a particular element are called two dimensional arrays. The following statement declares a two-dimensional array (3 by 3) within a procedure.

Dim Avg (3, 3) as Single

	Avg (0,0)	Avg (0,1)	Avg (0,2)	Avg (0,3)
Avg (Row, Col.)	Avg (1,0)	Avg (1,1)	Avg (1,2)	Avg (1,3)
	Avg (2,0)	Avg (2,1)	Avg (2,2)	Avg (2,3)
	Avg (3,0)	Avg (3,1)	Avg (3,2)	Avg (3,3)

Avg (3, 3)	2	6	1	0
	3	1	6	-3
	7	3	1	5
	5	4	-2.5	9

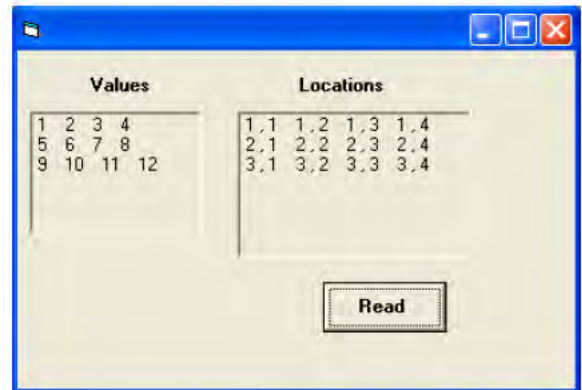
It is also possible to define the lower limits for one or both the dimensions as for fixed size arrays.

Example 6-6: Write a code program to read of two dimensional array A(3,4) on a row by row. Print the value and position of each element.

Solution:

```

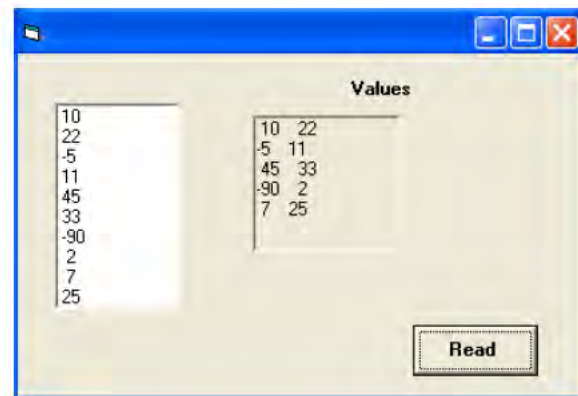
Dim A(3,4) As Single
For I=1 To 3                                (Rows)
For J= 1 To 4                                (Columns)
A(I,J) =Val(TextBox(""))
Next J
Next I
For I=1 To 3
For J= 1 To 4
Picture1.Print A(I, J) ; Space(2) ;
Picture2.Print I ; " , " ; J ; Space(2) ;
Next J
Picture1.Print : Picture2.Print
Next I
    
```



Example 6-8: Write a code program to create a two dimensional array N (5X2) into List Box on row by row. Print the values of array N.

Solution:

```
Dim N(5,2) As Single
K=0
For I= 1 To 5
For J=1 To 2
N(I,J)= Val (List1.List (K))
K=K+1
Next J, I
For I=1 To 5
For J= 1 To 2
Picture1.Print N(I, J) ; Space(2) ;
Next J : Picture1.Print : Next I
```



- **6-1-2 Dynamic Array:** The size of the array can be changed at the run time- size changes during the program execution.

In actual practice, the amount of data that a program will be processing is not known in advance. Programs should be flexible and incorporate a method for handling varying amounts of data. Visual basic makes this possible with the statement

ReDim *ArrayName* (N) As *Var Type*

This can use variables or expression when indicating the subscript range. However, ReDim statements can only be used inside procedures.

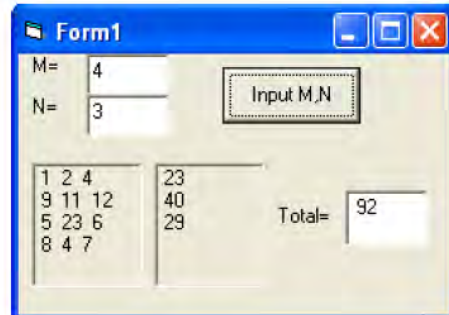
Example 6-15: An MxN matrix array A is entered into input box. Write a visual basic program segment which is calculated the SUM of elements in each Column and Stored in a one dimension Z. Print the arrays A and Z and the sum of all elements of array Z.

Solution:

```

Dim M, N
M=Val (Text1.Text) : N= Val (Text2.Text)
ReDim A (M, N), Z(N)
For I=1 To M
For J=1 To N
A(I,J) = Val (InputBox(""))
Next J , I
For J=1 To N
Sum=0
For I=1 To M
Sum=Sum+ A(I,J)
Next I
Z(J)=Sum
Total=Total +Z(J)
Next J
For I =1 To M
For J=1 To N
Picture1.Print A(I,J);
Next J: picture1.print: Next I
For I=1 To N
Picture2.print Z(I)
Next
Text3.text=Str(Total)

```



9- Sub Procedure and Function Procedure

Most computer programs that solve real-world problems are much larger than those presented in the first few chapters. Experience has shown that the best way to develop and maintain a large program is to construct it from smaller pieces each of which is more manageable than the original program. This technique is called divide and conquer. This chapter describes many key features that facilitate the design, implementation, operation and maintenance of large programs.

Functions and **Subroutines** are programs designed for specific task, and could be called from the main program or from sub-procedures without pre definition or declaration. Users are allowed to call in any number of times which save the main program space, since it avoids repetition of code these subroutines could be designed by user or could be previously built. The concepts and descriptions are summarized in the following table.

Item	Subroutine	Function
Code	Sub Name (arguments) Statements End Sub	Function Name (arguments) Statements End Function
Remark	<ul style="list-style-type: none"> • Need call statement • Return values by arguments • Return many values (arguments) • Used for Input/output, condition treatment • Could be used with out arguments. 	<ul style="list-style-type: none"> • Used in arithmetic statement • Return value by its name • Return one value • Used for arithmetic's or conversion of variable type.
Call Statement	Call Name(value1,value2,,,))	Z=name(value1)
Exit statement	Exit Sub	Exit Function

9.1 Sub Procedures

Sub procedure are created with the add procedure dialog (displayed when add procedure is selected from the tools menu). The add procedure menu item is grayed unless the code window is visible. Figure (9-1) displays the add procedure dialog. The procedure name is entered in TextBox Name and can be any valid identifier Frame Type contains option buttons for selecting the procedure type (Sub or Function). Frame scope contains option buttons for selecting keyword public or keyword private that will procedure, we will use keyword private, which also preceded our event procedures.

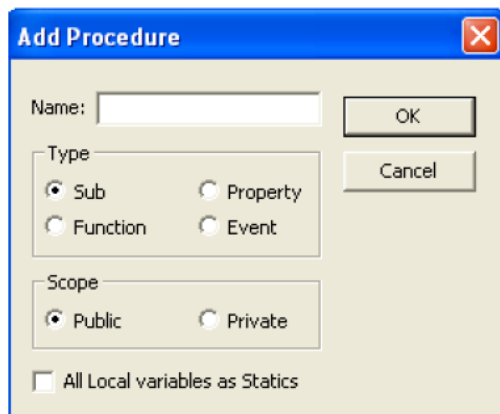


Figure (9-1): add procedure dialog

Once a valid name has been type into textbox name (add) has been passed, the procedure appears in the code window. Figure (9-2) shows procedure (add) which we created with the add procedure dialog. The code representing (add) in figure (9-2) is called the sub procedure definition.

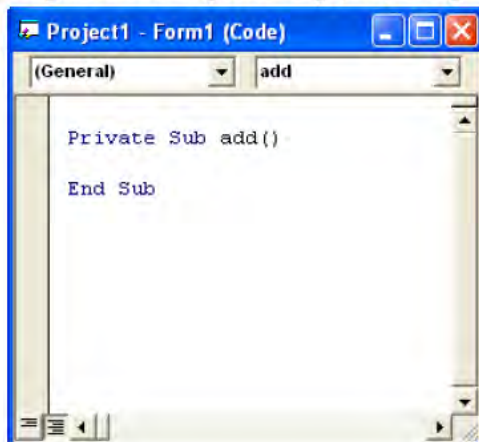


Figure (9-2): A sub procedure created with the add Procedure dialog.

Sub Procedures can also be created by typing the sub procedure directly into the code window. Once a line such as

```
Private Sub add2 ( )
```

Is typed and the enter key pressed, visual basic automatically creates the end sub line. Figure (9-3) shows the results when (add2) is typed directly into the code window.

The line

```
Private Sub add2 ( )
```

is the sub procedure header. The header contains keyword private, keyword sub, the procedure name, and parentheses. Any declarations and statements the programmer places between the header and end sub form the sub procedure body. Every time the sub procedure is called (or invoked) the body is immediately executed.

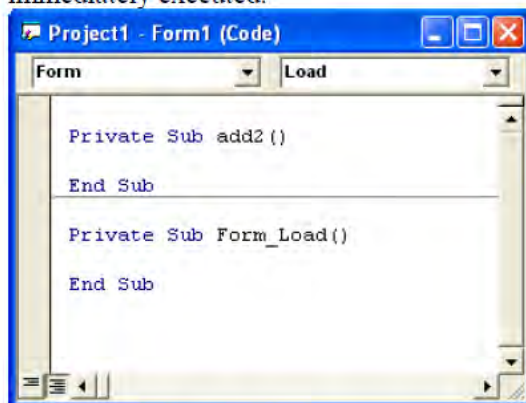


Figure (9-3): A Sub procedure created by typing directly into the code window.

Execution of the sub procedure terminates when end sub is reached. Program execution then continues with the statement immediately following the call to (add2).

All **Sub** procedure definitions contain parentheses which may be empty (e.g., add2). Consider the following sub procedure:

Private Sub Calculate (a as single, b as double)

Picture1.print a*b

End Sub

Which declares two parameter variables, (a, and b), in the parameter list. Parameter variables are declared using the **As** keyword. Parameter variables are not explicitly given a type default to **Variant**. Parameter variables receive their values from the procedure call and are used in the procedure body

The call to **Calculate** could also have be written as

Call Calculate (30,10.0)

Which uses keyword **Call** and encloses the arguments passed in a set of parentheses. The arguments passed can be variable names as well, for example, the call

Call Calculate (a, b)

Would pass a, and b to Calculate.

Example 9-1: Write a code program to read three integer numbers. Using a define sub procedure (Minimum) to determine the smallest of three integers. Display the smallest value in textbox.

Solution:

```
Private Sub Command1_Click()
```

```
Dim Num1 As Single, Num2 As Single, Num3 As Single
```

```
Num1 = Fix(Text1.Text)
```

```
Num2 = Fix(Text2.Text)
```

```
Num3 = Fix(Text3.Text)
```

```
Call Minimum(Num1, Num2, Num3, min)
```

```
Text4.Text = Str(min)
```

```
End Sub
```

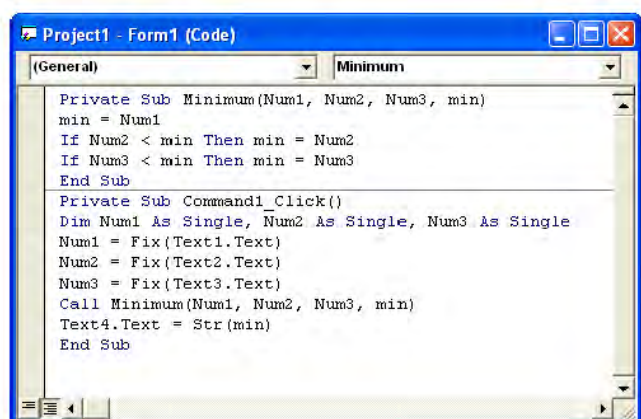
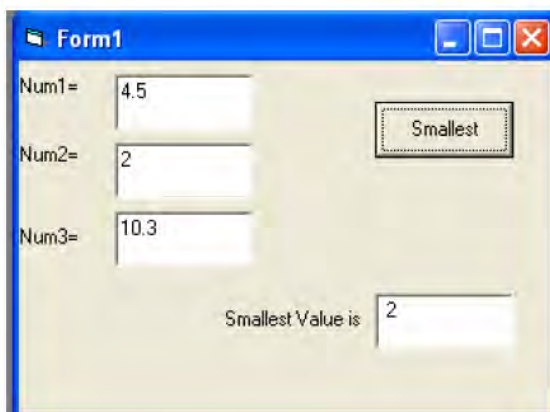
```
Private Sub Minimum(Num1, Num2, Num3, min)
```

```
min = Num1
```

```
If Num2 < min Then min = Num2
```

```
If Num3 < min Then min = Num3
```

```
End Sub
```



```
(General) Minimum
Private Sub Minimum(Num1, Num2, Num3, min)
min = Num1
If Num2 < min Then min = Num2
If Num3 < min Then min = Num3
End Sub
Private Sub Command1_Click()
Dim Num1 As Single, Num2 As Single, Num3 As Single
Num1 = Fix(Text1.Text)
Num2 = Fix(Text2.Text)
Num3 = Fix(Text3.Text)
Call Minimum(Num1, Num2, Num3, min)
Text4.Text = Str(min)
End Sub
```

9.2 Function Procedures: function procedures and sub procedures share the same characteristics, with one important difference- function procedures return a value (i.g., give a value back) to the caller, whereas sub procedures do not.

Function Procedures can be created with the add procedure dialog shown in figure (9-1) by selecting function. Figure (9-4) shows a function procedure. **Fact**, created with the add procedure dialog. **Fact** implicitly returns variant.

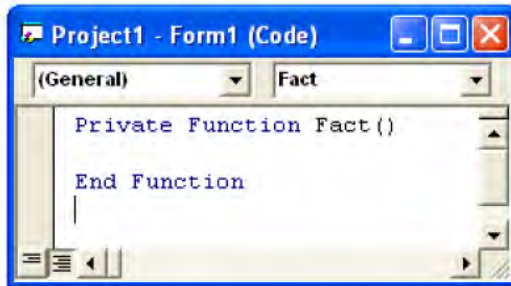


Figure (9-4): Function procedure created with add procedure dialog

Fact could also have been created by typing the function procedure directly into the code window. The line

Private Function Fact()

is the function procedure header. The header contains the keyword function, the function name and parentheses. The declarations and statements that the programmer will insert between the header and **End Function** form the function procedure body, **Fact** is invoked with the line.

Result= Fact()

When a function procedure name (such as **Fact**) is encountered at run time, the function procedure is called, causing its body statements to execute. Consider the complete definition for **Fact**

Private Function Fact(N)

Fact=N^2

End Function

A function procedure return value is specified in the body by assigning a value to the function procedure name, as in

Fact=N^2

Then returns (along with the value returned) to the calling statement

Result=Fact (N)

And the return value is assigned to variable result. Program execution then continues with the next statement after the call to **Fact**.

All function procedure definitions contain parentheses, the parentheses may be empty (e.g. **Fact**) or may contain one parameter variable declarations. Consider the following function procedure:

Private Function Area (s1 as single,s2 as single)

Area=s1*s2

End Function

Which declare two parameter variables s1, and s2. Area's return type is variant. Area is called with the statement

```
Square=area(8.5, 7.34)
```

The value 8.5 is stored in s1 and the value 7.34 is stored in s2.

Example 9-3: Write a code program to read three integer numbers. Using a define sub Function (Min) to determine the smallest of three integers. Display the smallest value in textbox.

Solution:

```
Private Sub Command1_Click()
```

```
Dim Num1 As Single, Num2 As Single, Num3 As Single, Result As Single
```

```
Num1 = Fix(Text1.Text)
```

```
Num2 = Fix(Text2.Text)
```

```
Num3 = Fix(Text3.Text)
```

```
Result = Min(Num1, Num2, Num3)
```

```
Text4.Text = Str(Result)
```

```
End Sub
```

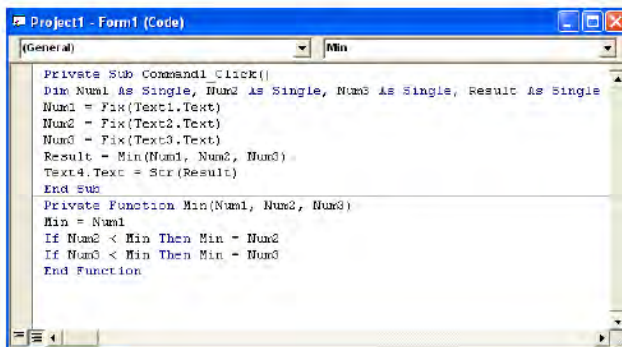
```
Private Function Min(Num1, Num2, Num3)
```

```
Min = Num1
```

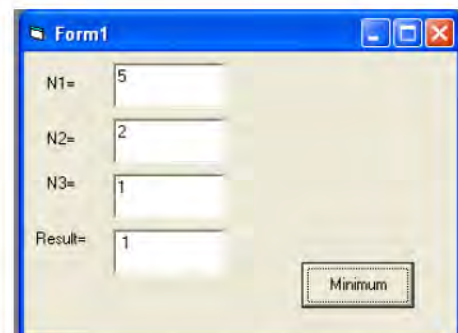
```
If Num2 < Min Then Min = Num2
```

```
If Num3 < Min Then Min = Num3
```

```
End Function
```



```
Project1 - Form1 (Code)
(General)
Min
Private Sub Command1_Click()
Dim Num1 As Single, Num2 As Single, Num3 As Single, Result As Single
Num1 = Fix(Text1.Text)
Num2 = Fix(Text2.Text)
Num3 = Fix(Text3.Text)
Result = Min(Num1, Num2, Num3)
Text4.Text = Str(Result)
End Sub
Private Function Min(Num1, Num2, Num3)
Min = Num1
If Num2 < Min Then Min = Num2
If Num3 < Min Then Min = Num3
End Function
```



Form1

N1=	5
N2=	2
N3=	1
Result=	1

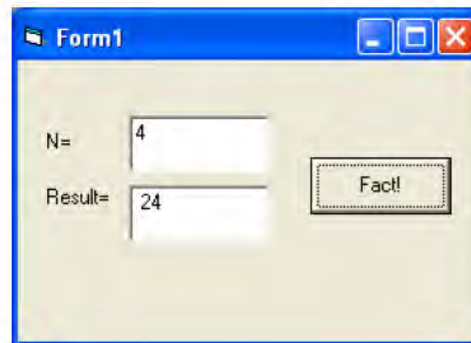
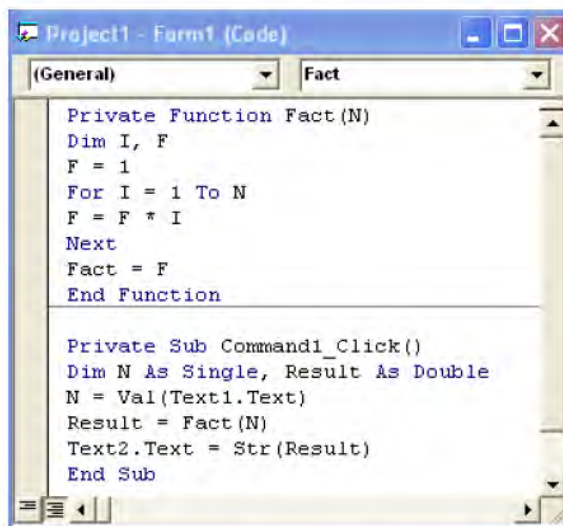
Minimum

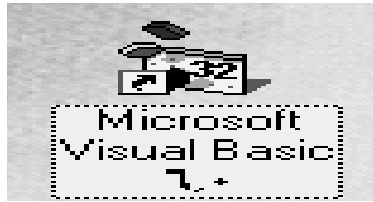
Example 9-4: Write a code program to input the value of N. Using a define sub function fact to determine(N!). Display the result into text box.

Solution:

```
Private Sub Command1_Click()  
Dim N As Single, Result As Double  
N = Val(Text1.Text)  
Result = Fact(N)  
Text2.Text = Str(Result)  
End Sub
```

```
Private Function Fact(N)  
Dim I, F  
F = 1  
For I = 1 To N  
F = F * I  
Next  
Fact = F  
End Function
```





University of Baghdad
Collage of Education For Pure Science
Ibn Alhythim
Dep.of Computer Science

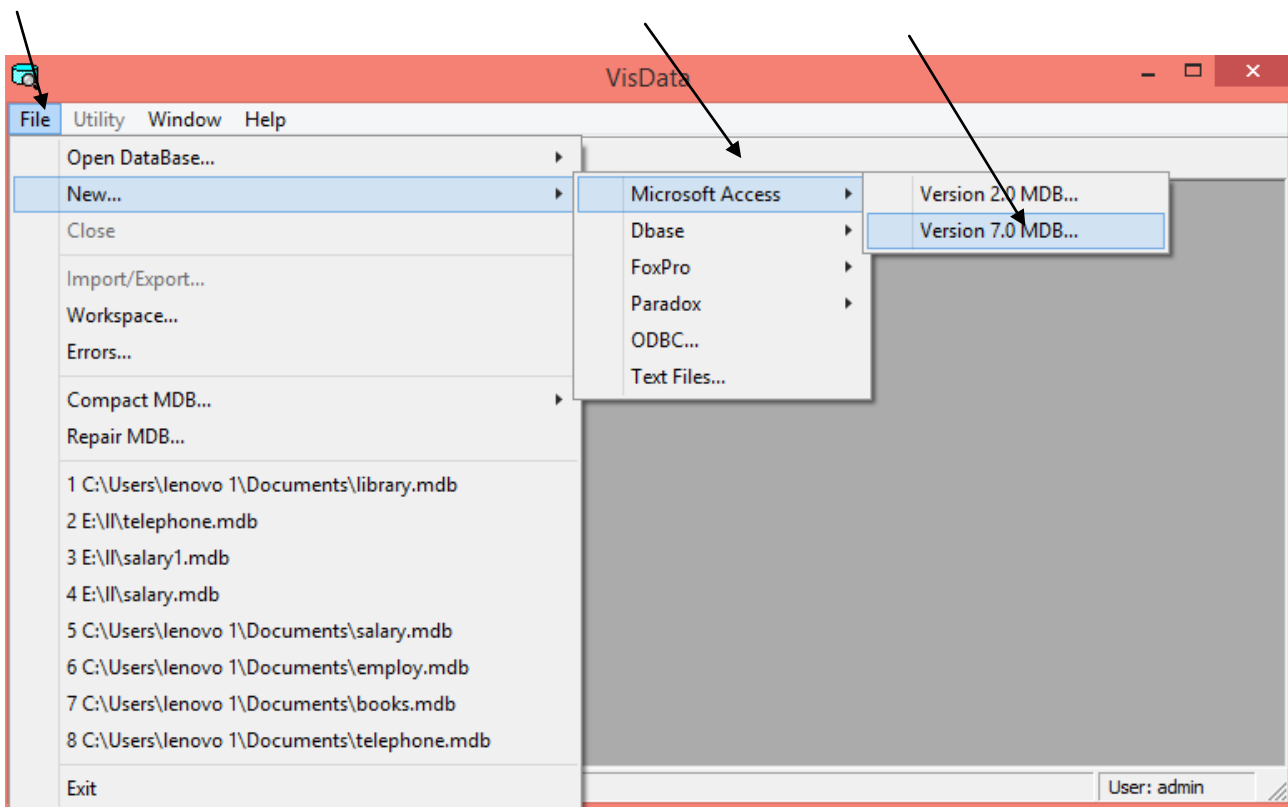
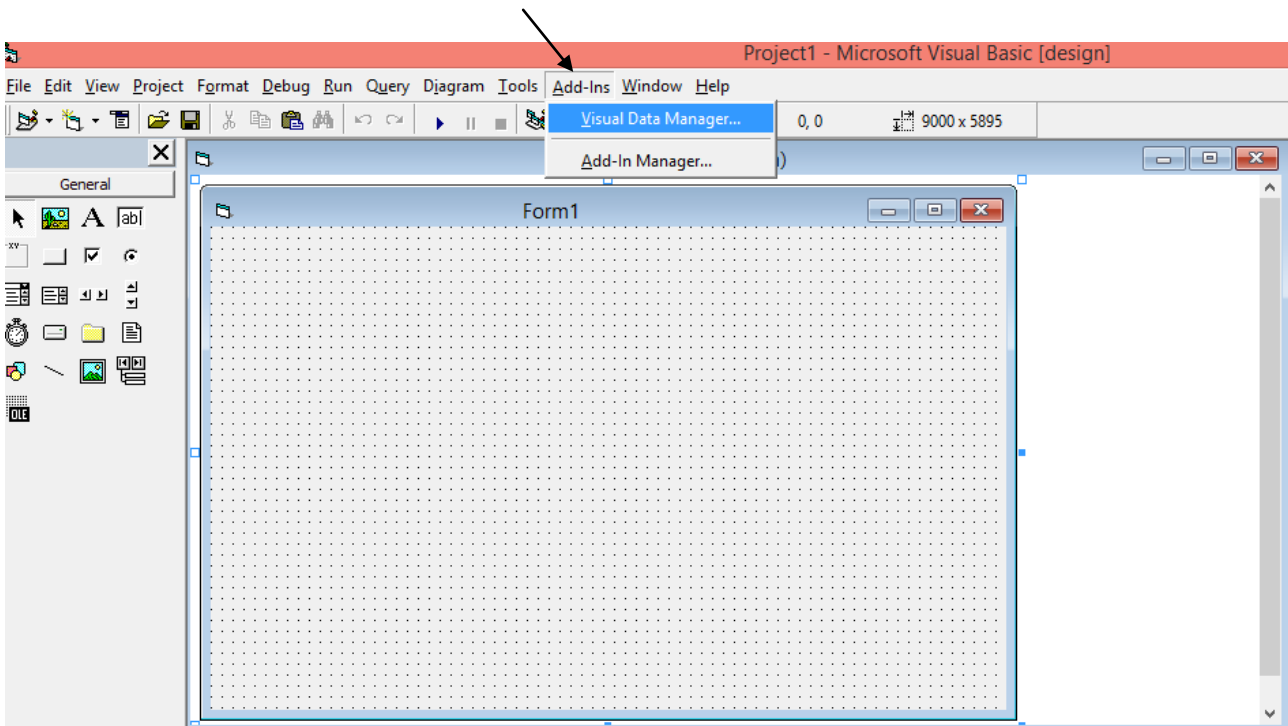
Windows Programming

Microsoft Visual Basic 6.0

Chapter Four-Part 2

M.Sc. Lubab Ahmed

1. Steps create Table using Add-Ins from Menu Bar



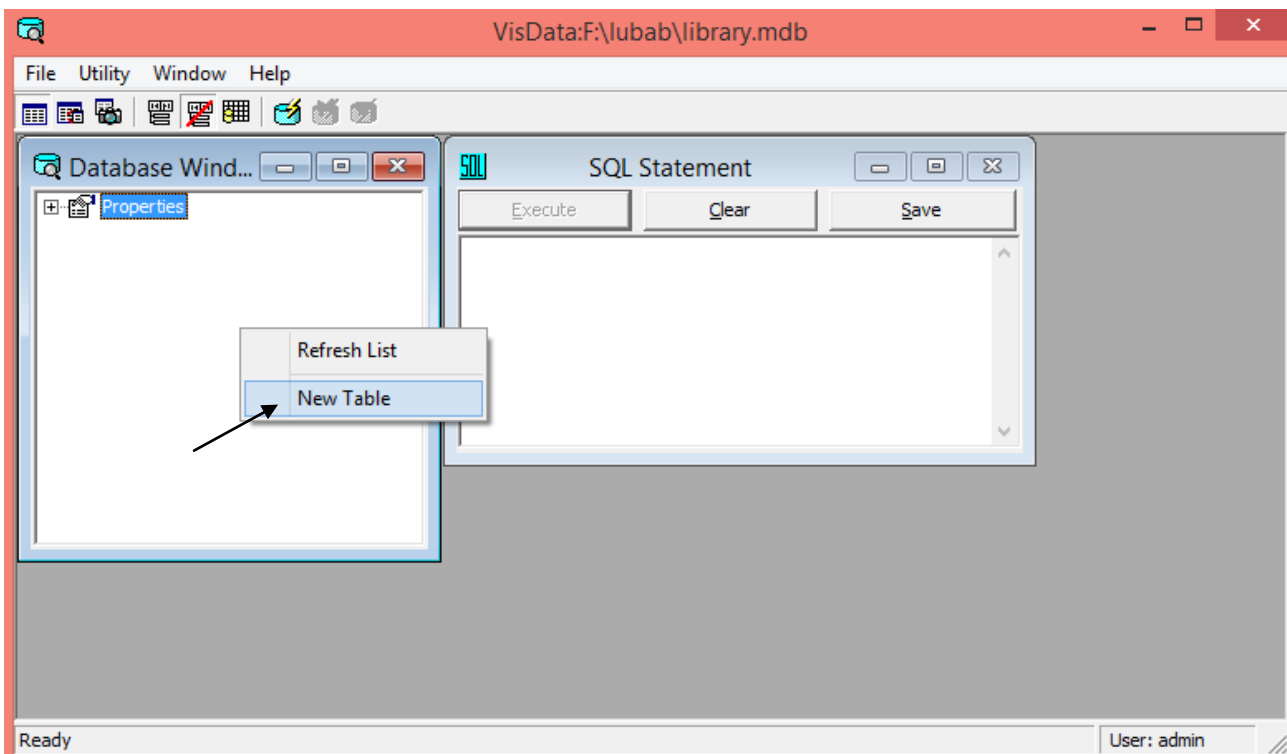
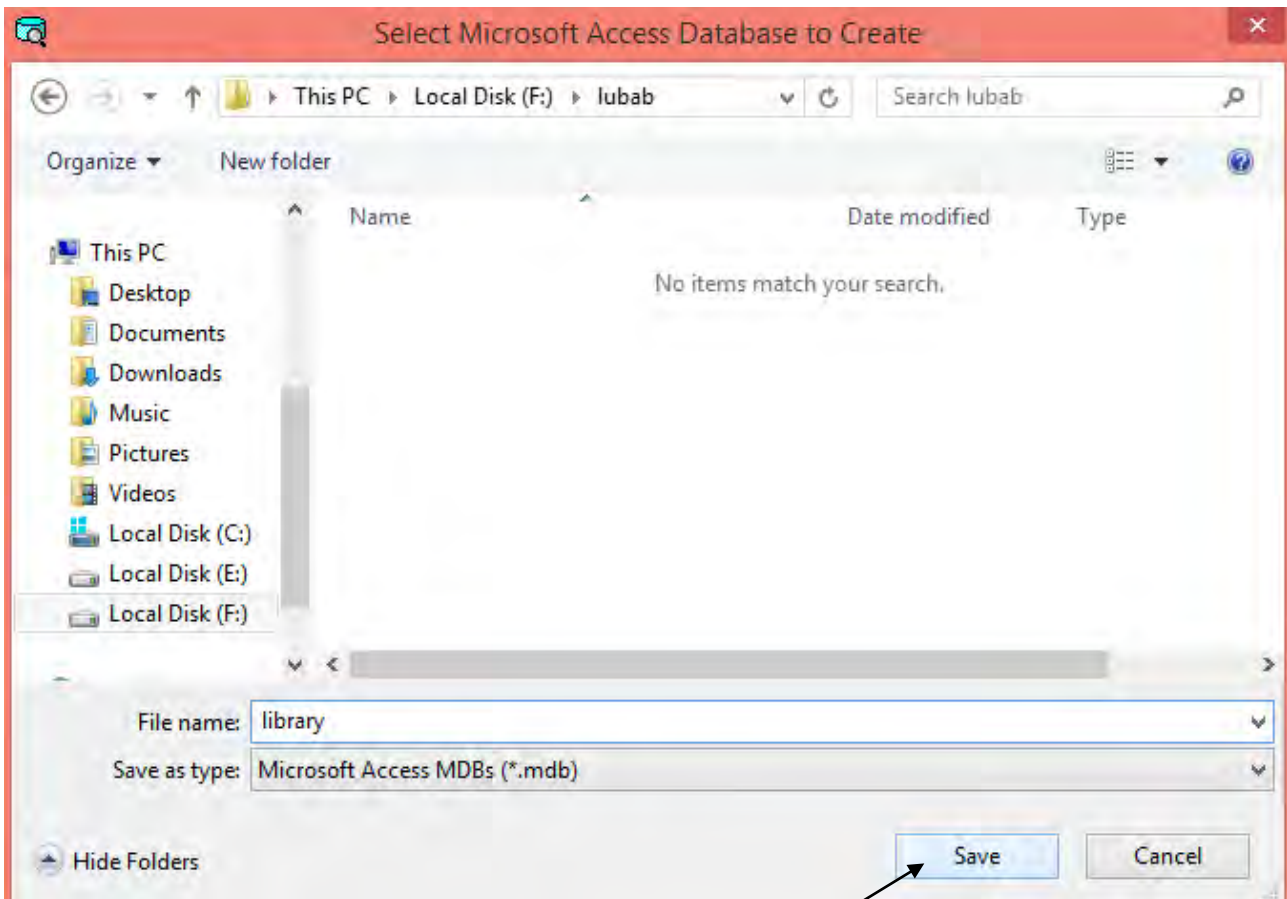


Table Structure

Table Name:

Field List:

Name:	<input type="text"/>	
Type:	<input type="text"/>	<input type="checkbox"/> FixedLength
Size:	<input type="text"/>	<input type="checkbox"/> VariableLength
CollatingOrder:	<input type="text"/>	<input type="checkbox"/> AutoIncrement
		<input type="checkbox"/> AllowZeroLength
OrdinalPosition:	<input type="text"/>	<input type="checkbox"/> Required
ValidationText:	<input type="text"/>	
ValidationRule:	<input type="text"/>	
DefaultValue:	<input type="text"/>	

Index List:

Name:	<input type="text"/>	
<input type="checkbox"/> Primary	<input type="checkbox"/> Unique	<input type="checkbox"/> Foreign
<input type="checkbox"/> Required	<input type="checkbox"/> IgnoreNull	
Fields:	<input type="text"/>	

Add Field

Name:

OrdinalPosition:

Type:

Size:

FixedField
 VariableField
 AutoIncrField
 AllowZeroLength

ValidationText:

ValidationRule:

DefaultValue:

Add Field

Name: **OrdinalPosition:**

Type: **ValidationText:**

Size: **ValidationRule:**

FixedField
 VariableField

AutoIncrField
 AllowZeroLength
 Required

Table Structure

Table Name:

Field List:

Author
Year

Name: **Type:** FixedLength

Size: VariableLength

CollatingOrder: AutoIncrement

OrdinalPosition: AllowZeroLength

Required

ValidationText:

ValidationRule:

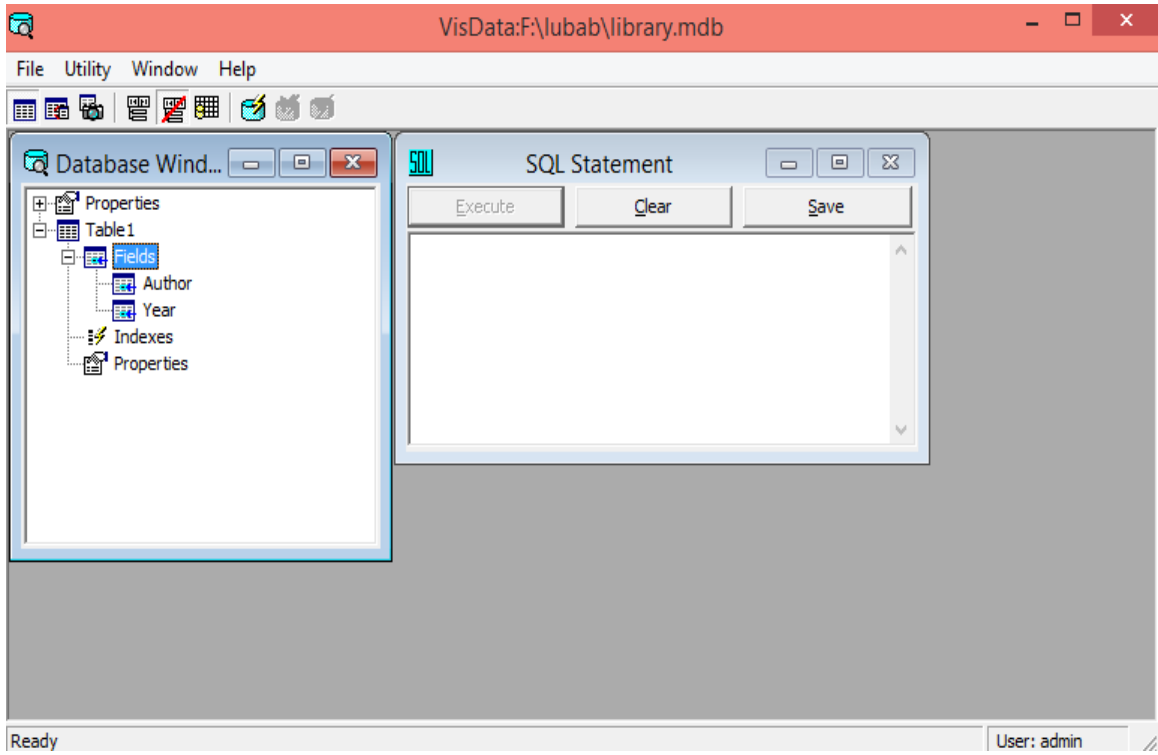
DefaultValue:

Index List:

Name:

Primary Unique Foreign
 Required IgnoreNull

Fields:



Manipulating with Records

Add Record

```
Private Sub Command1_Click()  
  
On Error Resume Next  
  
Adodc1.Recordset.AddNew  
  
End Sub
```

Delete Record

```
On Error Resume Next  
  
a = MsgBox("Are you want to delete this record?", vbYesNo)  
  
If a = vbYes Then  
  
Adodc1.Recordset.Delete  
  
Adodc1.Recordset.MoveNext  
  
End If  
  
End Sub
```

Update Record

```
Private Sub Command3_Click()  
  
On Error Resume Next  
  
Adodc1.Recordset.Update  
  
End Sub
```

Transfer to the first Record

```
Private Sub Command4_Click()  
  
On Error Resume Next  
  
Adodc1.Recordset.MoveFirst
```

End Sub

Transfer to the last Record

```
Private Sub Command5_Click)(
```

```
On Error Resume Next
```

```
Adodc1.Recordset.MoveLast
```

```
End Sub
```

Transfer to the next Record

```
Private Sub Command6_Click)(
```

```
On Error Resume Next
```

```
Adodc1.Recordset.MoveNext
```

```
End Sub
```

Transfer to the Previous Record

```
Private Sub Command7_Click)(
```

```
On Error Resume Next
```

```
Adodc1.Recordset.MovePrevious
```

```
End Sub
```

SQL Statement

Order Ascending

```
Private Sub Command8_Click()
```

```
On Error Resume Next
```

```
Adodc1.RecordSource = "select * from Table_Name order by Fileld_Name asc"
```

```
Adodc1.CommandType = adCmdText
```

Adodc1.Refresh

End Sub

Order Descending

Private Sub Command10_Click()

On Error Resume Next

Adodc1.RecordSource = "select * from **Table_Name** order by **Filed_Name** desc"

Adodc1.CommandType = adCmdText

Adodc1.Refresh

End Sub

Show Recods

Private Sub Command11_Click()

On Error Resume Next

Adodc1.RecordSource = "select * from **Table_Name** "

Adodc1.CommandType = adCmdText

Adodc1.Refresh

End Sub

Number Of records

Private Sub Command12_Click()

MsgBox Adodc1.Recordset.RecordCount

End Sub

Maximum Value

Private Sub Command13_Click()

On Error Resume Next


```
Adodc1.RecordSource = "select * from Table_Name where Filed_Name = (select  
max (Field_Name) from Table_Name )"
```

```
Adodc1.CommandType = adCmdText
```

```
Adodc1.Refresh
```

```
End Sub
```

Minimum Value

```
Private Sub Command13_Click()
```

```
On Error Resume Next
```

```
Adodc1.RecordSource = "select * from Table_Name where Filed_Name = (select  
min (Field) from Table_Name )"
```

```
Adodc1.CommandType = adCmdText
```

```
Adodc1.Refresh
```

```
End Sub
```

Research in Database

```
Private Sub Command15_Click()
```

```
On Error Resume Next
```

```
Adodc1.RecordSource = "select * from Table_Name where Filed_Name = ' Value' "
```

```
Adodc1.CommandType = adCmdText
```

```
Adodc1.Refresh
```

```
End Sub
```

Research by text

```
Private Sub Command18_Click()
```

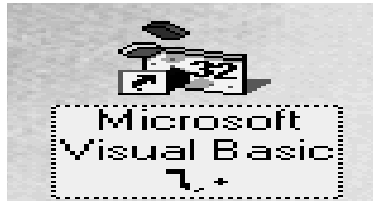
```
On Error Resume Next
```

```
Adodc1.RecordSource = Text no.Text
```

```
Adodc1.CommandType = adCmdText
```

```
Adodc1.Refresh
```

```
End Sub
```



University of Baghdad
Collage of Education For Pure Science
Ibn Alhythim
Dep.of Computer Science

Windows Programming

Microsoft Visual Basic 6.0

Chapter Four-Part 3

M.Sc. Lubab Ahmed

Picture Boxes



- The **picture box** allows you to place graphics information on a form. It is best suited for dynamic environments - for example, when doing animation.
- Picture boxes lie in the top layer of the form display. They behave very much like small forms within a form, possessing most of the same properties as a form.
- Picture Box Properties:

AutoSize	If True, box adjusts its size to fit the displayed graphic.
Font	Sets the font size, style, and size of any printing done in the picture box.
Picture	Establishes the graphics file to display in the picture box.

- Picture Box Events:

Click	Triggered when a picture box is clicked.
DblClick	Triggered when a picture box is double-clicked.

- Picture Box Methods:

Cls	Clears the picture box.
Print	Prints information to the picture box.

- Picture Box LoadPicture Procedure:

An important function when using picture boxes is the **LoadPicture** procedure. It is used to set the **Picture** property of a picture box at run-time.

Example

```
picExample.Picture = LoadPicture("c:\pix\sample.bmp")
```

This command loads the graphics file `c:\pix\sample.bmp` into the **Picture** property of the `picExample` picture box. The argument in the `LoadPicture` function must be a legal, complete path and file name, else your program will stop with an error message.

Image Boxes



- An **image box** is very similar to a picture box in that it allows you to place graphics information on a form. Image boxes are more suited for static situations - that is, cases where no modifications will be done to the displayed graphics.
- Image boxes appear in the middle-layer of form display, hence they could be obscured by picture boxes and other objects. Image box graphics can be resized by using the **Stretch** property.
- Image Box Properties:

Picture	Establishes the graphics file to display in the image box.
Stretch	If False, the image box resizes itself to fit the graphic. If True, the graphic resizes to fit the control area.
- Image Box Events:

Click	Triggered when a image box is clicked.
DbClick	Triggered when a image box is double-clicked.
- The image box does not support any methods, however it does use the **LoadPicture** function. It is used in exactly the same manner as the picture box uses it. And image boxes can load the same file types: bitmap (.bmp), icon (.ico), metafiles (.wmf), GIF files (.gif), and JPEG files (.jpg). With Stretch = True, all three graphic types will expand to fit the image box area.

Image Boxes



- An **image box** is very similar to a picture box in that it allows you to place graphics information on a form. Image boxes are more suited for static situations - that is, cases where no modifications will be done to the displayed graphics.
- Image boxes appear in the middle-layer of form display, hence they could be obscured by picture boxes and other objects. Image box graphics can be resized by using the **Stretch** property.
- Image Box Properties:

Picture	Establishes the graphics file to display in the image box.
Stretch	If False, the image box resizes itself to fit the graphic. If True, the graphic resizes to fit the control area.
- Image Box Events:

Click	Triggered when a image box is clicked.
DbClick	Triggered when a image box is double-clicked.
- The image box does not support any methods, however it does use the **LoadPicture** function. It is used in exactly the same manner as the picture box uses it. And image boxes can load the same file types: bitmap (.bmp), icon (.ico), metafiles (.wmf), GIF files (.gif), and JPEG files (.jpg). With Stretch = True, all three graphic types will expand to fit the image box area.

Drive List Box



- The **drive list box** control allows a user to select a valid disk drive at run-time. It displays the available drives in a drop-down combo box. No code is needed to load a drive list box; Visual Basic does this for us. We use the box to get the current drive identification.

- Drive List Box Properties:

Drive Contains the name of the currently selected drive.

- Drive List Box Events:

Change Triggered whenever the user or program changes the drive selection.

Directory List Box



- The **directory list box** displays an ordered, hierarchical list of the user's disk directories and subdirectories. The directory structure is displayed in a list box. Like, the drive list box, little coding is needed to use the directory list box - Visual Basic does most of the work for us.

- Directory List Box Properties:

Path Contains the current directory path.

- Directory List Box Events:

Change Triggered when the directory selection is changed.

File List Box



- The **file list box** locates and lists files in the directory specified by its Path property at run-time. You may select the types of files you want to display in the file list box.

- File List Box Properties:

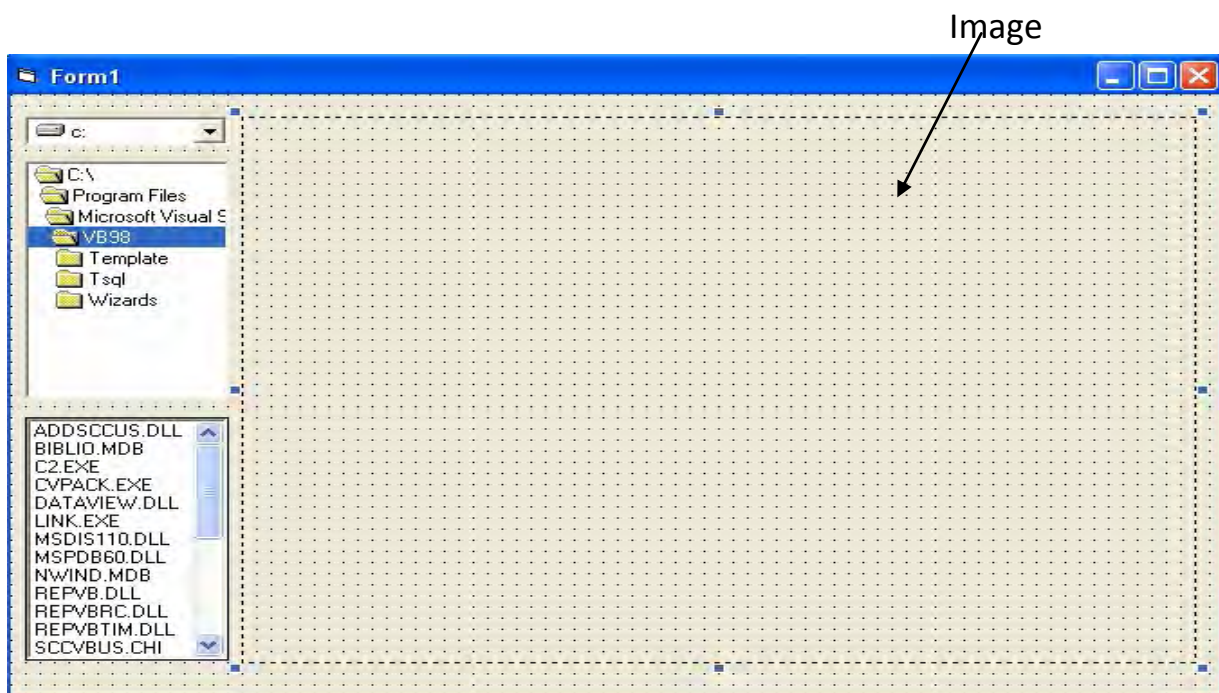
File Name	Contains the currently selected file name.
Path	Contains the current path directory.
Pattern	Contains a string that determines which files will be displayed. It supports the use of * and ? wildcard characters. For example, using *.dat only displays files with the .dat extension.

- File List Box Events:

DbClick	Triggered whenever a file name is double-clicked.
PathChange	Triggered whenever the path changes in a file list box.

- You can also use the **MultiSelect** property of the file list box to allow multiple file selection.

Example : **Photo Browser** 1. Design the form



2. Adjust the properties of the form and its objects according to the following table:

Object	Property Name	Stage of Changing
File1	Pattern	*.bmp; *.jpg; *.gif *.ico;
Image1	Stretch	True

3. Write the following code :

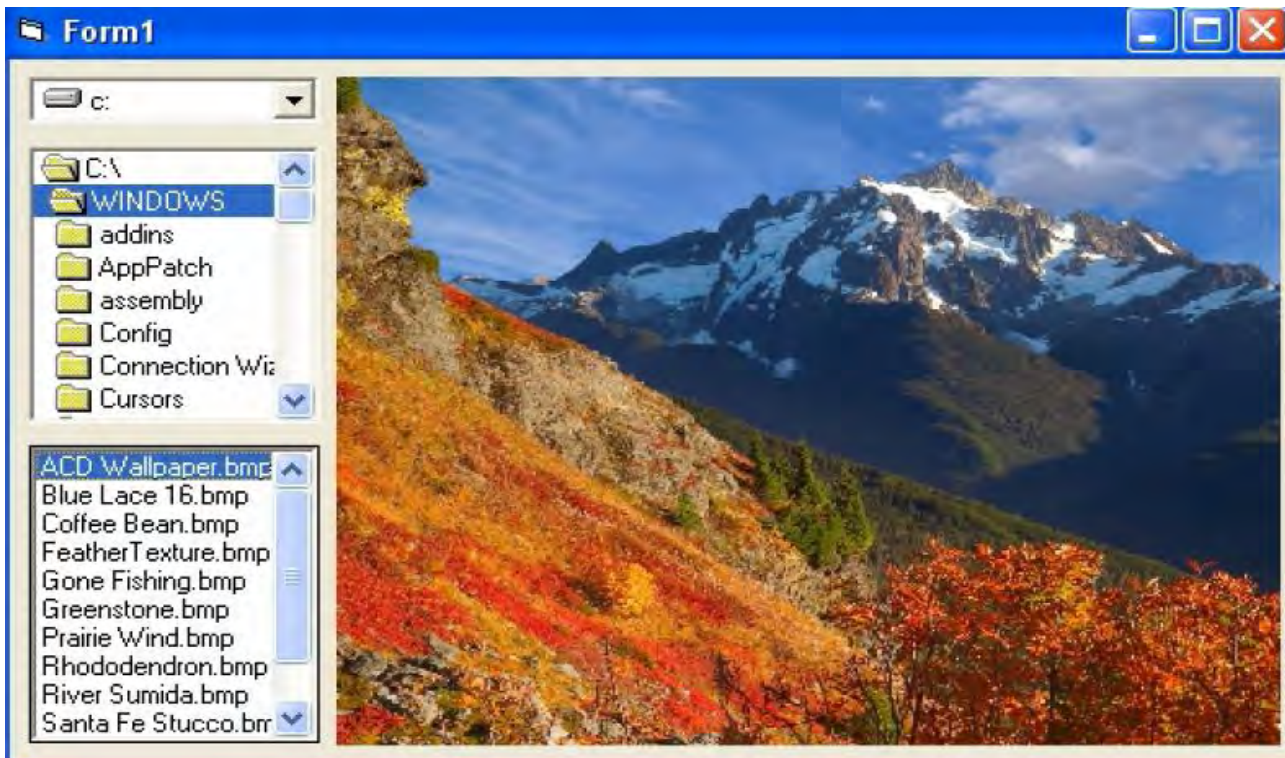
```

Project1 - Form1 (Code)
File1 Click
Private Sub Dir1_Change()
File1.Path = Dir1.Path
End Sub

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

Private Sub File1_Click()
SelectedFile = File1.Path & "\" & File1.FileName
Image1.Picture = LoadPicture(SelectedFile)
End Sub
    
```

4. After Run



Common Dialog Boxes



- The primary use for the drive, directory, and file name list boxes is to develop custom file access routines. Two common file access routines in Windows-based applications are the **Open File** and **Save File** operations. Fortunately, you don't have to build these routines.
- To give the user a standard interface for common operations in Windows-based applications, Visual Basic provides a set of **common dialog boxes**, two of which are the **Open** and **Save As** dialog boxes. Such boxes are familiar to any Windows user and give your application a professional look. And, with Windows 95, some context-sensitive help is available while the box is displayed. Appendix II lists many symbolic constants used with common dialog boxes.
- The Common Dialog control is a '**custom control**' which means we have to make sure some other files are present to use it. In normal setup configurations, Visual Basic does this automatically. If the common dialog box does not appear in the Visual Basic toolbox, you need to add it. This is done by selecting **Components** under the **Project** menu. When the selection box appears, click on **Microsoft Common Dialog Control**, then click **OK**.
- The common dialog tool, although it appears on your form, is invisible at run-time. You cannot control where the common dialog box appears on your screen. The tool is invoked at run-time using one of five '**Show**' methods. These methods are:

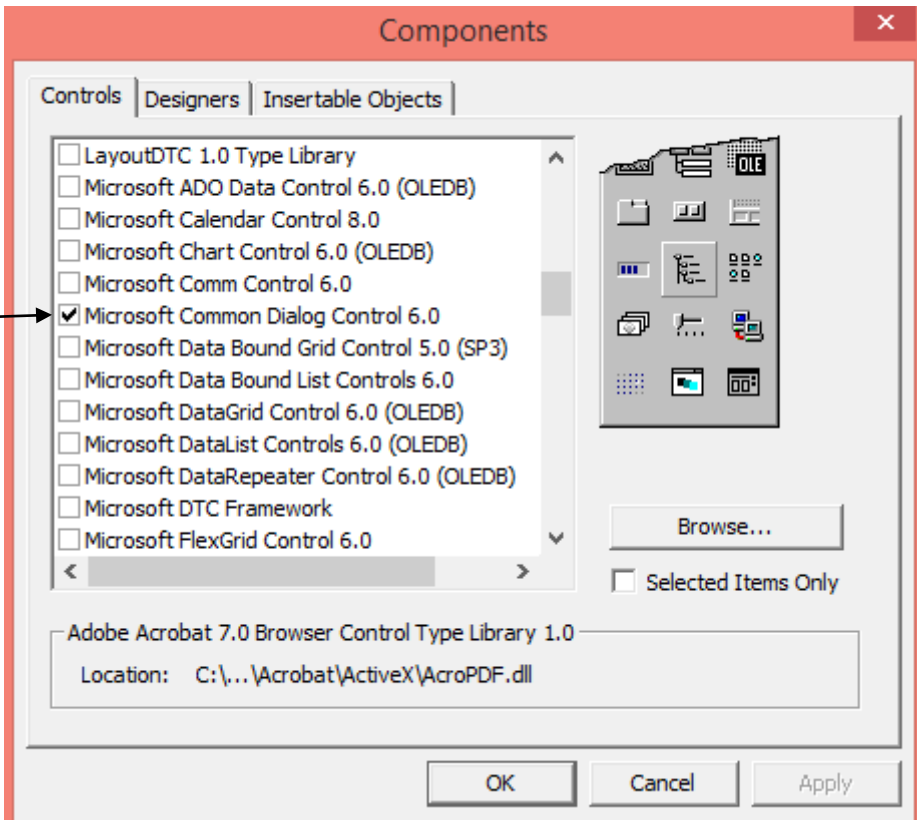
Method	Common Dialog Box
ShowOpen	Open dialog box
ShowSave	Save As dialog box
ShowColor	Color dialog box
ShowFont	Font dialog box
ShowPrinter	Printer dialog box

- The format for establishing a common dialog box named **cdlExample** so that an **Open** box appears is:

```
cdlExample.ShowOpen
```

Control to the program returns to the line immediately following this line, once the dialog box is closed in some manner. Common dialog boxes are system modal.

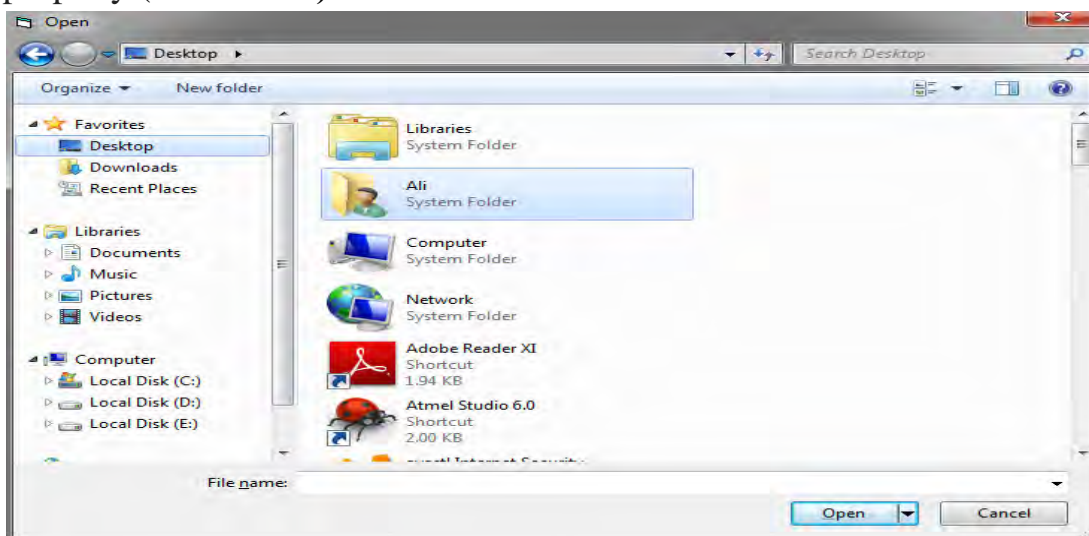
To add Common Dialog click object → Components → Microsoft
 Common Dialog Control 6.0 → ok



1. **Open Dialog box** : A box (Open) is used to open a specific file within the computer. We use the following formula to open a box :

CommonDialog1.ShowOpen

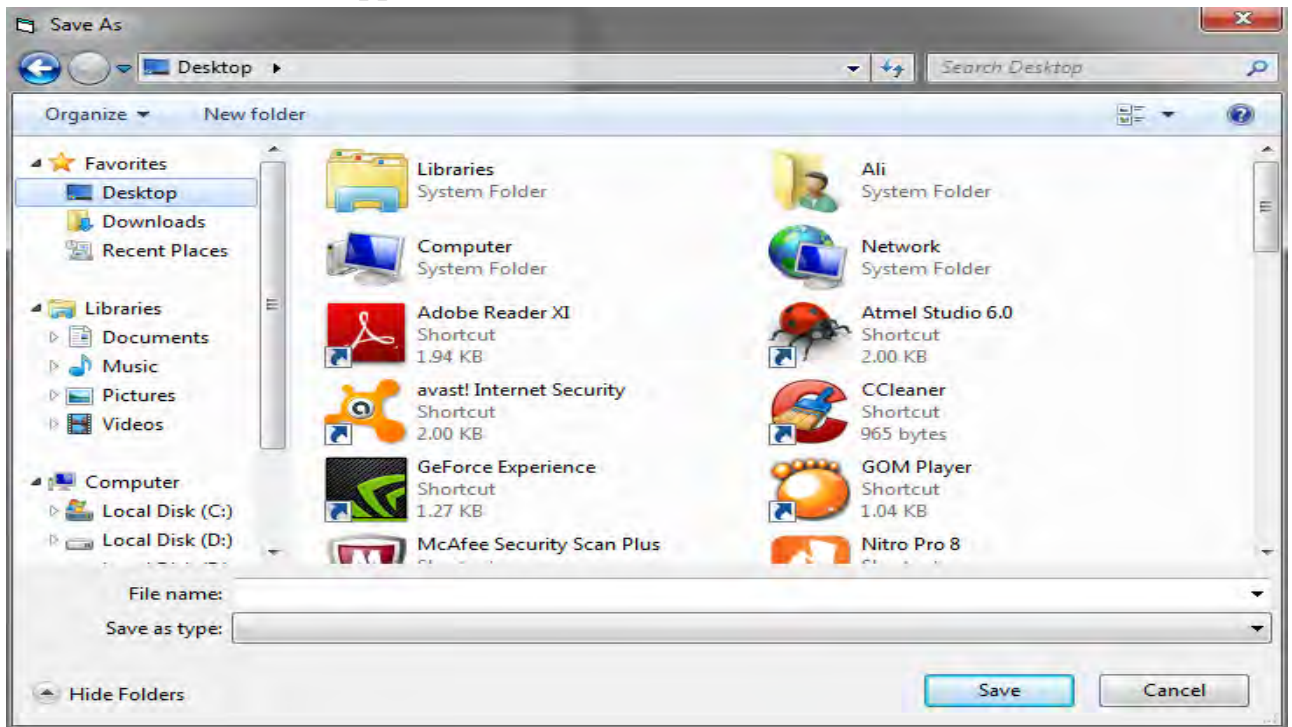
The box shown below appears. The file name chosen can be obtained through a property (FileName)



2. **Save Dialog box** : We use the following formula to save a box :

CommonDialog1.ShowSave

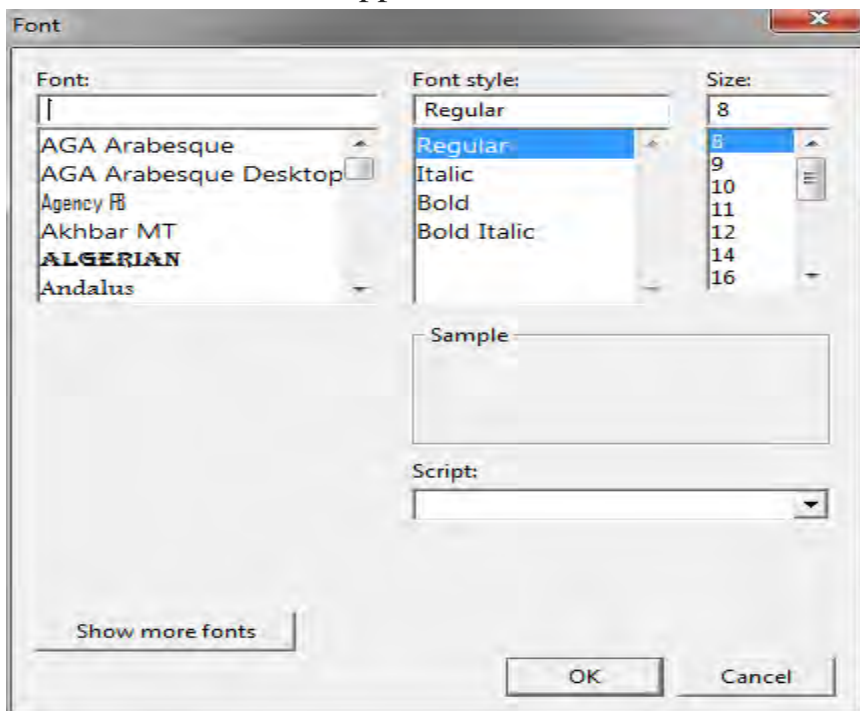
The box shown below appears :



3. **Font Dialog box** : We use the following formula to font a box :

CommonDialog1.ShowFont

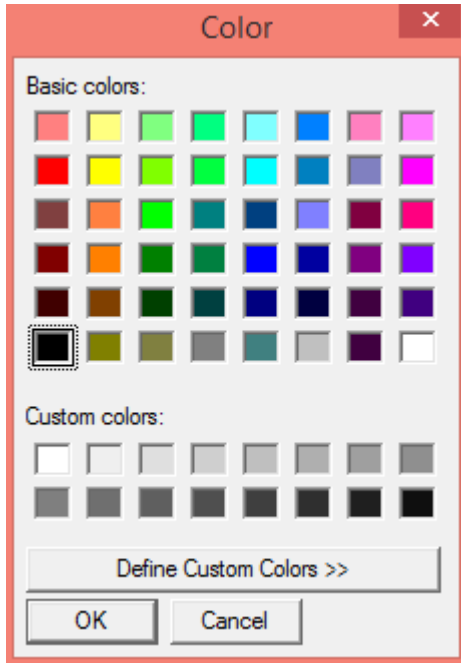
The box shown below appears :



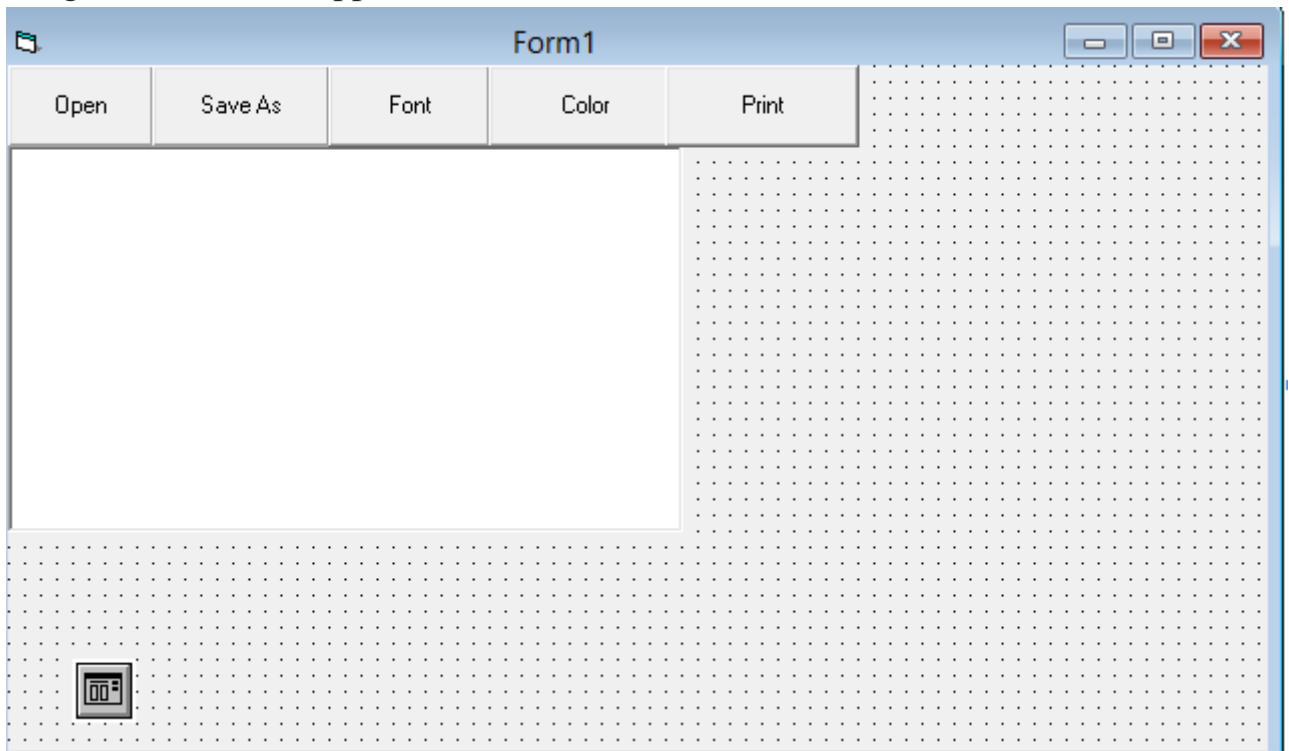
4. **Color Dialog Box:** allows the user to choose a color for a specific purpose, such as changing the background color of the program or changing the background color of an existing control Within the program . We use the following formula to font a box :

CommonDialog1.ShowColor

The box shown below appears :



Example : 1. Design Form with one CommonDialog1, textbox and 5 commonds , The design shown below appears :



2. Write the code listed below:

1. Open Dialog box

```
Private Sub Command1_Click()
```

```
' CancelError is True.
```

```
  On Error GoTo ErrorHandler
```

```
  ' Set filters.
```

```
  CommonDialog1.Filter = "All Files (*.*)|*.*|Text _ Files (*.txt)|*.txt"
```

```
  ' Specify default filter.
```

```
  CommonDialog1.FilterIndex = 2
```

```
intfile = FreeFile
```

```
CommonDialog1.InitDir = "c:\\"
```

```
CommonDialog1.DefaultExt = ".txt"
```

```
  ' Display the Open dialog box.
```

```
  CommonDialog1.ShowOpen
```

```
  Open CommonDialog1.FileName For Input As #intfile
```

```
  Text1.Text = Input$(LOF(intfile), intfile)
```

```
  Close #intfile
```

```
ErrorHandler:
```

```
' User pressed Cancel button.
```

```
  Exit Sub
```

```
End Sub
```

2. Save Dialog box

```
Private Sub Command2_Click()
```

```
' CancelError is True.
```

```
  On Error GoTo ErrorHandler
```

```
  ' Set filters.
```

```
  CommonDialog1.Filter = "All Files (*.*)|*.*|Text _ Files (*.txt)|*.txt"
```

```
  ' Specify default filter.
```

```
  CommonDialog1.FilterIndex = 2
```

```
  ' Display the Open dialog box.
```

```
  CommonDialog1.ShowSave
```

```
If CommonDialog1.FileName <> "" Then
```

```
  apppath = CommonDialog1.FileName
```

```
  valtext = Text1.Text
```

```
Open apppath For Append As #1
Print #1, valtext
Close #1
End If
ErrorHandler:
' User pressed Cancel button.
  Exit Sub
End Sub
```

3. Font Dialog box

```
Private Sub Command3_Click()
' Set Cancel to True.
  CommonDialog1.CancelError = True
  On Error GoTo ErrorHandler
' Set the Flags property.
  CommonDialog1.Flags = cdlCFBoth Or cdlCFEffects
' Display the Font dialog box.
  CommonDialog1.ShowFont
' Set text properties according to user's
' selections.
  Text1.Font.Name = CommonDialog1.FontName
  Text1.Font.Size = CommonDialog1.FontSize
  Text1.Font.Bold = CommonDialog1.FontBold
  Text1.Font.Italic = CommonDialog1.FontItalic
  Text1.Font.Underline = CommonDialog1.FontUnderline
  Text1.ForeColor = CommonDialog1.Color
  Exit Sub
ErrorHandler:
' User pressed Cancel button.
  Exit Sub
End Sub
```

4. Color Dialog box

```
Private Sub Command4_Click()
' Set Cancel to True.
```

```
CommonDialog1.CancelError = True
On Error GoTo ErrHandler
' Set the Flags property.
CommonDialog1.Flags = cdlCCRGBInit
' Display the Color dialog box.
CommonDialog1.ShowColor
' Set the form's background color to the selected
' color.
Form1.BackColor = CommonDialog1.Color
Exit Sub
```

ErrHandler:

```
' User pressed Cancel button.
Exit Sub
```

End Sub

5. Print Dialog box

```
Private Sub Command5_Click()
Dim BeginPage, EndPage, NumCopies, Orientation, i
' Set Cancel to True.
CommonDialog1.CancelError = True
On Error GoTo ErrHandler
' Display the Print dialog box.
CommonDialog1.ShowPrinter
' Get user-selected values from the dialog box.
BeginPage = CommonDialog1.FromPage
EndPage = CommonDialog1.ToPage
NumCopies = CommonDialog1.Copies
Orientation = CommonDialog1.Orientation
For i = 1 To NumCopies
' Put code here to send data to your printer.
Next
Exit Sub
```

ErrHandler:

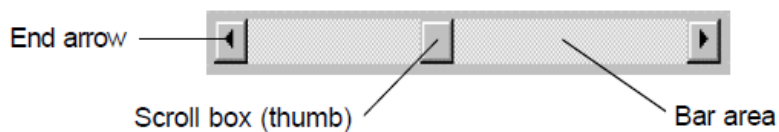
```
' User pressed Cancel button.
Exit Sub
```

End Sub

Horizontal and Vertical Scroll Bars



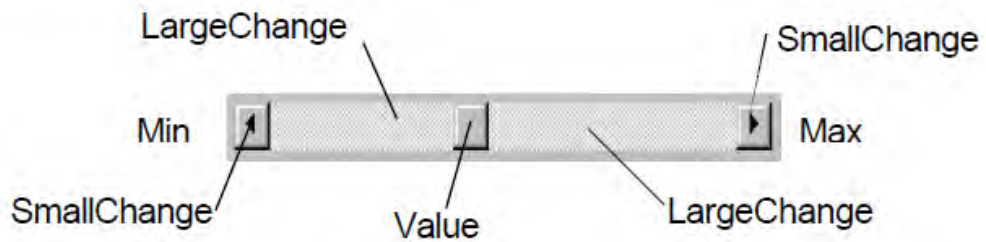
- Horizontal and vertical **scroll bars** are widely used in Windows applications. Scroll bars provide an intuitive way to move through a list of information and make great input devices.
- Both type of scroll bars are comprised of three areas that can be clicked, or dragged, to change the scroll bar value. Those areas are:



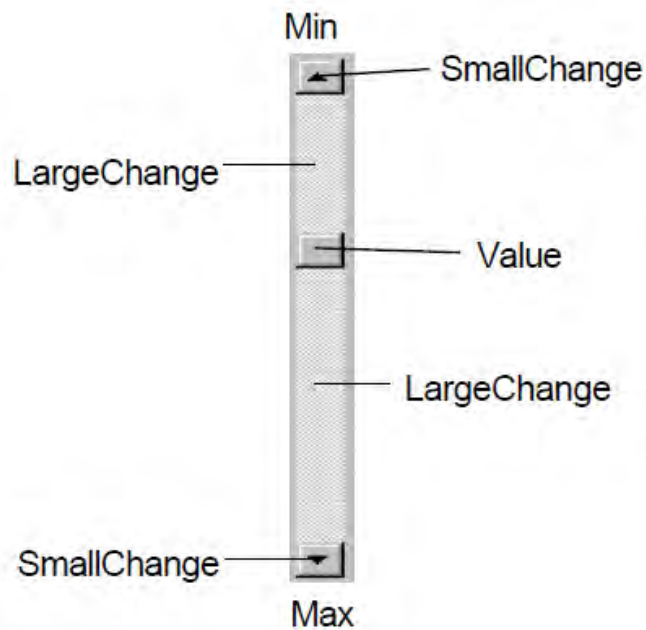
Clicking an **end arrow** increments the **scroll box** a small amount, clicking the **bar area** increments the scroll box a large amount, and dragging the scroll box (thumb) provides continuous motion. Using the properties of scroll bars, we can completely specify how one works. The scroll box position is the only output information from a scroll bar.

- Scroll Bar Properties:
 - LargeChange** Increment added to or subtracted from the scroll bar **Value** property when the bar area is clicked.
 - Max** The value of the horizontal scroll bar at the far right and the value of the vertical scroll bar at the bottom. Can range from -32,768 to 32,767.
 - Min** The other extreme value - the horizontal scroll bar at the left and the vertical scroll bar at the top. Can range from -32,768 to 32,767.
 - SmallChange** The increment added to or subtracted from the scroll bar **Value** property when either of the scroll arrows is clicked.
 - Value** The current position of the scroll box (thumb) within the scroll bar. If you set this in code, Visual Basic moves the scroll box to the proper position.

Properties for horizontal scroll bar:



Properties for vertical scroll bar:



- Scroll Bar Events:

- | | |
|---------------|--|
| Change | Event is triggered after the scroll box's position has been modified. Use this event to retrieve the Value property after any changes in the scroll bar. |
| Scroll | Event triggered continuously whenever the scroll box is being moved. |

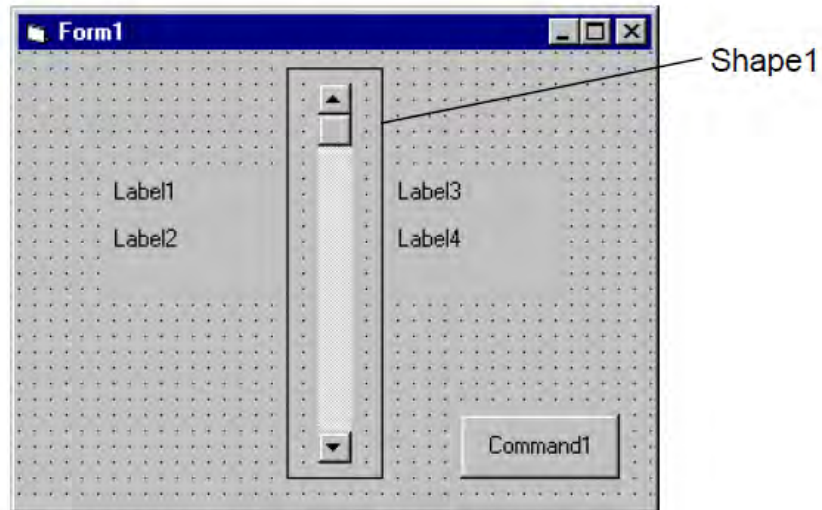
Example :

Temperature Conversion

Start a new project. In this project, we convert temperatures in degrees Fahrenheit (set using a scroll bar) to degrees Celsius. As mentioned in the **Review and Preview** section, you should try to build this application with minimal reference to the notes. To that end, let's look at the project specifications.

One Possible Approach to Temperature Conversion Application:

1. Place a shape, a vertical scroll bar, four labels, and a command button on the form. Put the scroll bar within the shape - since it is in the top-layer of the form, it will lie in the shape. It should resemble this:



2. Set the properties of the form and each object:

Form1:

BorderStyle	1-Fixed Single
Caption	Temperature Conversion
Name	frmTemp

Shape1:

BackColor	White
BackStyle	1-Opaque
FillColor	Red
FillStyle	7-Diagonal Cross
Shape	4-Rounded Rectangle

VScroll1:

LargeChange	10
Max	-60
Min	120
Name	vsbTemp
SmallChange	1
Value	32

Label1:

Alignment	2-Center
Caption	Fahrenheit
FontSize	10
FontStyle	Bold

Label2:

Alignment	2-Center
AutoSize	True
BackColor	White
BorderStyle	1-Fixed Single
Caption	32
FontSize	14
FontStyle	Bold
Name	lblTempF

Label3:

Alignment	2-Center
Caption	Celsius
FontSize	10
FontStyle	Bold

Label4:

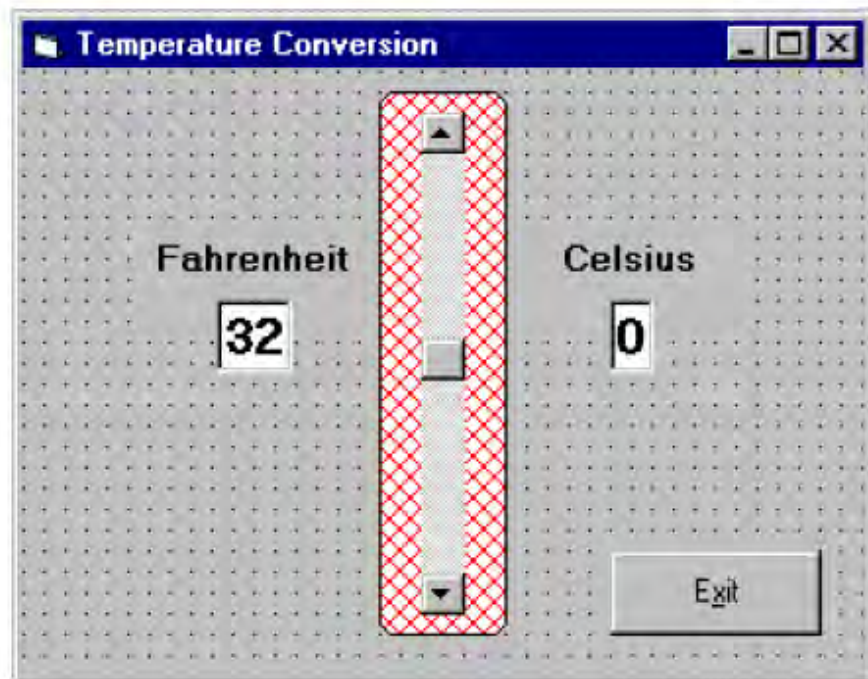
Alignment	2-Center
AutoSize	True
BackColor	White
BorderStyle	1-Fixed Single
Caption	0
FontSize	14
FontStyle	Bold
Name	lblTempC

Command1:

Cancel	True
Caption	E&xit
Name	cmdExit

Note the temperatures are initialized at 32F and 0C, known values.

When done, the form should look like this:



Put this code in the general declarations of your code window.

Put this code in the general declarations of your code window.

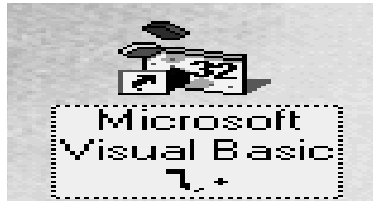
```
Option Explicit
Dim TempF As Integer
Dim TempC As Integer
```

This makes the two temperature variables global.

Attach the following code to the scroll bar **Scroll** event.

```
Private Sub vsbTemp_Scroll()
'Read F and convert to C
TempF = vsbTemp.Value
lblTempF.Caption = Str(TempF)
TempC = CInt((TempF - 32) * 5 / 9)
lblTempC.Caption = Str(TempC)
End Sub
```

This code determines the scroll bar Value as it scrolls, takes that value as Fahrenheit temperature, computes Celsius temperature, and displays both values.



University of Baghdad
Collage of Education For Pure Science
Ibn Alhythim
Dep.of Computer Science

Windows Programming

Microsoft Visual Basic 6.0

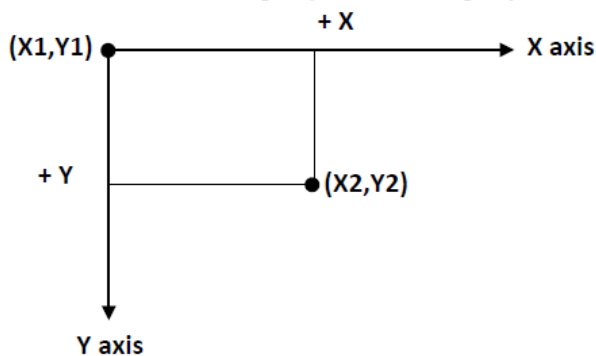
Chapter Five

M.Sc. Lubab Ahmed

7- Graphics in Visual Basic

7.1 Introduction: Graphics are the elements of a picture. Colors, lines, rectangles, patterns, text, etc. are all graphics. Graphics are visual. Visual Basic provides graphics capabilities for drawing shapes in different colors and patterns. Visual basic is also capable of displaying many popular image formats. Although the graphics capabilities may not be as feature rich as graphics software programs, visual basic's graphic capabilities are integral to creating polished windows applications.

7.2 Coordinate Systems: To draw in visual basic, we must understand Visual Basic's coordinate system as shown in figure below, that identifies points on the screen (such as forms or pictureboxes). By default, the upper-left point on the screen has coordinate (0,0), which is commonly called the origin. A coordinate pair is composed of an *x coordinate* (the horizontal coordinate) and *y coordinate* (the vertical coordinate). The x coordinate is the horizontal distance on the x axis from the origin. The y coordinate is the vertical distance on the y axis from the origin. The unit that a coordinate system is measured in is *called a scale*. Visual basic provides eight coordinate system scales. Most controls as well as the form use twips by default. Property *ScaleMode* specifies the scale.



User-defined coordinates are defined using method scale. Two set of coordinates define the scale. The first coordinate set defines the upper-left corner and the second coordinate set defines the lower-right corner. The statement,

Scale (xx1 , yy1) - (xx2 , yy2)

For example:

- Scale(0,0)-(100,100)
- Scale (100,100)-(0,0)
- Scale(-100,100)-(100,-100)

7.2 Graphics Method:

Visual basic provides several methods for creating graphics. The graphics methods, summarized in the following table, apply to forms.

Method	Description
Line	Draws lines on a form. Can also be used to draw rectangles.
Circle	Draws circles on a form. Can also be used to draw ellipses
Pset	Sets a point's color
Print	Draw text on a form.

❖ **Method Line:** draws lines and rectangles between two sets of coordinates. The first set of coordinates is the starting point and the second is the ending point.

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




For example:

- Line (0,0)-(100,100),VbBlue
- Line(100,50)-(50,50),QbColor(5)
- Line(50,50)-(50,100),RGB(45,100,10)

For rectangles (also called boxes) the first coordinate set specifies the upper-left corner and the second specifies the lower-right corner.

Line (x1,y1)-(x2,y2),color, B [or BF]

The visual basic constant (Vb) which represents the color name, the third argument (**B**), indicates that the method should draw a rectangle. A third argument of (**BF**) would indicate that the rectangle should be filled (solid). For example

- Line (0,0)-(55,21), , B 
- Line(25,50)-(75,100), , Bf 
- 

➤ **Note:** There are three ways to specify a color value at run time.

- 1- Use RGB(1 To 255, 1 To 255, 1 To 255) function
- 2- Use the QBColor(1 to 15) function to choose one of 15 Microsoft QuickBasic color as shown in table below
- 3- Enter a color value directly (VbColor) as shown in table below.

Vb Code	Color	Constant	Vb Code	Color	Constant
0	Black	vbBlack	8	Grey	vbGrey
1	Blue	vbBlue	9	Light Blue	vbLightBlue
2	Green	vbGreen	10	Light Green	vbLightGreen
3	Cyan	vbCyan	11	Light Cyan	vbLightCyan
4	Red	vbRed	12	Light Red	vbLightRed
5	Magenta	vbMagenta	13	Light Magenta	vbLightMagenta
6	Brown	vbBrown	14	Yellow	vbYellow
7	White	vbWhite	15	Bright White	vbBrightWhite












❖ **Method Circle:** draws circles, ellipses, arcs, and sectors. A circle's radius is the distance from the circle's center to any circle point. An ellipse differs from a circle in that its aspect ratio (the ratio of height to width) is not 1. Arcs is the curved portion of sectors. Sectors are wedge shaped pieces of a circle. Radians (from **0 to 2π**) must be used for sector and arc angles.

Circle (x1,y1), radius, color, start angle, end angle, proportion

For Example

Scale (0, 0)-(100, 100)

pi = 3.14156

- | | | |
|--|---|---------------------------|
| • Circle (50, 25), 5 |  | circle |
| Circle (15, 15), 5, , 0, pi / 2 |  | Arc (C.C.W angle (+)) |
| • Circle (15, 30), 5, , 0, pi |  | Arc (C.C.W angle (+)) |
| • Circle (15, 45), 5, , pi / 2, 3 * pi / 2 |  | Arc (C.C.W angle (+)) |
| • Circle (15, 60), 5, , pi, 0 |  | Arc (C.C.W angle(+)) |
| • Circle (15, 80), 5, , -pi / 2, -3 * pi / 2 |  | Sector (C.W angle (-)) |
| • Circle (45, 80), 5, , -pi, -pi / 2 |  | Sector (C.W angle (-)) |
| • Circle (65, 80), 5, , -pi / 2, -pi |  | Sector (C.W angle (-)) |
| • Circle (85, 80), 5, , pi / 2, -pi |  | Sector (C.W angle (-)) |
| • Circle (50, 45), 5, , , , 2 |  | Ellipse $\frac{Y}{X} > 1$ |
| • Circle (50, 65), 5, , , , 0.5 |  | Ellipse $\frac{Y}{X} < 1$ |

❖ **Method Pset:** turns on a point by changing the color at the point for example, the statement

Pset (x,y),color

Pset(40,40),VbRed

❖ **Method Print:** To draw text on the form. The default X coordinate is 0 and visual Basic automatically increments the y coordinate to draw on the next line. The current drawing coordinates are stored in properties *currentX* and *currentY*. For Example:

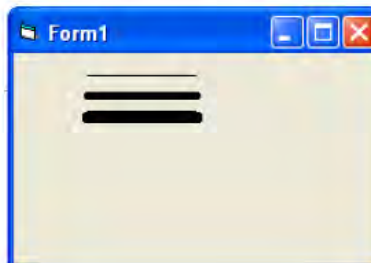
```
CurrentX=1  
CurrentY=3  
Print "Visual Basic"
```

Note: you can use the (*Form1.ForeColor*) property to specify any color to draw text.

7.3 Graphics Properties: Several drawing properties can be used with drawing methods. In this section, we introduce properties:

1- **DrawWidth:** The draw width property specifies the width of line for output from the graphics methods. For Example :

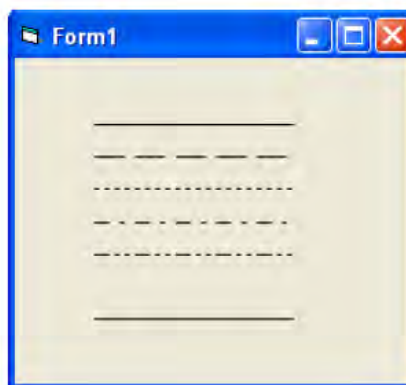
```
Private Sub Form Activate ()  
Scale(0,0)-(100,200)  
Drawwidth=1  
Line(20,20)-(50,20)  
Drawwidth=5  
Line(20,40)-(50,40)  
Drawwidth=8  
Line(20,60)-(50,60)
```



2- **DrawStyle:** the draw style property specifies whether the lines created with graphics methods are solid or have a broken pattern control. There are seven different draw style values (from 0 to 6).

For Example:

```
Private Sub Form_Activate()  
Scale (0, 0)-(100, 100)  
DrawWidth = 1  
Y = 10  
For I = 0 To 6  
DrawStyle = I  
Y = Y + 10  
Line (20, Y) - (70, Y)  
Next I
```

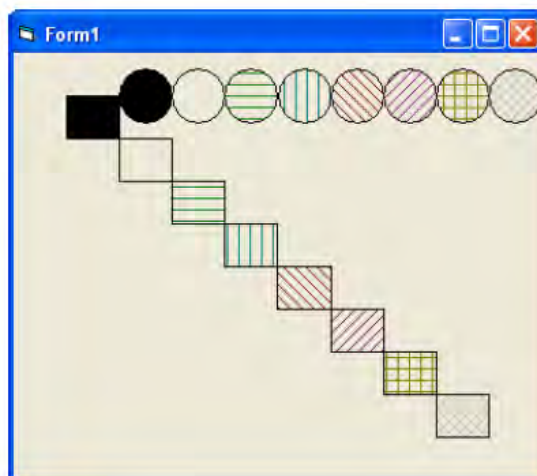


3- **FillStyle:** As long as you don't change the setting of the fill style Property, the box appears empty. (The box does get filled with default *FillStyle* and Settings, But *FillStyle* default to 1-Transparent). You can change the *FillStyle* property to any the settings listed in the following table:

Setting	Description
0	Solid. Fills in box with the color set for the <i>FillColor</i> Property
1	Transparent (the default). Graphical object appears empty, no matter what color is used
2	Horizontal lines
3	Vertical lines
4	Upward diagonal lines
5	Downward diagonal lines
6	Crosshatch
7	Diagonal Crosshatch

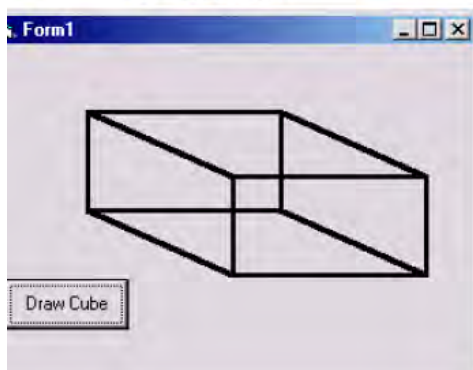
For Example:

```
Private Sub Form_Activate()
    Scale (0, 0)-(100, 100)
    For i = 0 To 7
        Y = Y + 10
        FillStyle = i
        FillColor = QBColor(i)
        Line (Y, Y)-(Y + 10, Y + 10), , B
        Circle (Y + 15, 10), 5
    Next i
```



Example 7-1: Write a code program to draw the figures below.

1- Solution:



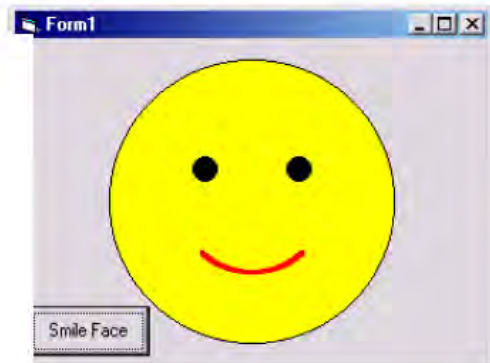
```
Project1 - Form1 (Code)
Command1 Click
Private Sub Command1_Click()
    ' Assign a scale
    Scale (0, 0)-(100, 100)
    DrawWidth = 3

    ' Draw 2 rectangles
    Line (20, 20)-(60, 50), , B
    Line (50, 40)-(90, 70), , B

    ' Draw 4 connecting lines (the last is dotted)
    Line (20, 20)-(50, 40)
    Line (60, 20)-(90, 40)
    Line (20, 50)-(50, 70)
    DrawStyle = 2
    Line (60, 50)-(90, 70)
    ' DrawStyle =1 means solid line
    ' DrawStyle =2 means dotted line
End Sub
```

3-

```
Private Sub Command1_Click()  
    ' Assign a scale  
    Scale (0, 0)-(10, 10)  
  
    ' to draw the face (circle) filled with yellow  
    FillColor = vbYellow  
    FillStyle = 0  
    Circle (5, 5), 3, vbBlack  
  
    ' To draw 2 eyes filled with black color  
    FillColor = vbBlack  
    Circle (4, 4), 0.25, vbBlack  
    Circle (6, 4), 0.25, vbBlack  
  
    ' To draw a smile mouth  
    DrawWidth = 3  
    pi = 3.14159  
    Circle (5, 5), 1.5, vbRed, 225 * pi / 180, 315 * pi / 180  
  
End Sub
```



3-



```
Private Sub Command1_Click()  
    ' Assign a scale  
    Scale (0, 0)-(10, 10)  
  
    ' To draw big face (circle) filled with the color yellow  
    FillStyle = 0  
    FillColor = vbYellow  
    Circle (5, 5), 3, vbBlack, , , 1.2  
  
    ' To draw two eyes filled with black  
    FillColor = vbBlack  
    Circle (4, 4), 0.25, vbBlack  
    Circle (6, 4), 0.25, vbBlack  
    '  
    ' To draw smile mouth  
    Form1.DrawWidth = 3  
    pi = 3.14159  
    Circle (5, 9), 1.5, vbRed, 45 * pi / 180, 135 * pi / 180  
  
    ' To draw a nose  
    Line (5, 4.5)-(5.5, 5.5)  
    Line (5, 4.5)-(4.5, 5.5)  
    Line (4.5, 5.5)-(5.5, 5.5)  
End Sub
```