VISUAL BASIC:- **Visual basic is a high level programming language developed from the earlier DOS version called BASIC. VB stands for Visual Basic, and is a High-Level Programming Language. A programming language basically allows you to create programs or applications, such as Microsoft Word. These can then be run on a number of operating systems, depending on which language you chooseA High-Level programming language essentially means a language that is (relatively) easy to learn, and the code you write is similar to English words. In comparison, a Low-level language would mainly involves working with assembly language (ie what the PC's own language). It would involve far more work creating a program using a Low Level language, so nearly everyone uses High Level programming languages now. Visual Basic,**[**Java**](https://www.developerfusion.com/t/java/)**,**[**C++**](https://www.developerfusion.com/t/cpp/)**, Pascal, and nearly every other language are now high level languages**

**COMMON WORDS**

|  |  |
| --- | --- |
| **Application** | When you 'program', you are creating a program. This is also known as an Application. Word, Excel and Lotus Notes are all applications. |
| **Code** | This is what you type when you are programming! |
| **IDE** | The *IDE*or*Integrated Development Environment* is simply the main Visual Basic window. This is where you create and run your Visual Basic Applications. |
| **Form** | This is what programmers often call a dialog box, or as you probably know it, a window. |
| **Form Window** | In the Visual Basic window, you can see a form or window, within another window. This, is the *Form Window*, and is where you create your own form, and add your own controls and text to it. |
| **Control** | A *control* is something you place on a *form*. When you see a box to enter text, or have a drop-down box of choices, or a button to click, these are all controls. |
| **Toolbox** | This is the window shown down the left-hand side of the *IDE*, which lists all the *controls* that you can add to the *form*. (See Figure 2) |
| **Project Window** | This is the window on the right hand side of the Visual Basic window, which lists all the items in your project, and lets you choose a form to modify, or edit its code. |
| **Property** | When programming, every *control* has properties. These allow you change the appearance of a control, or the text it displays. |
| **Properties Window** | The properties window, not surprisingly allows you to modify these properties |

Tha Visual Basic IDE is made up of a number of components

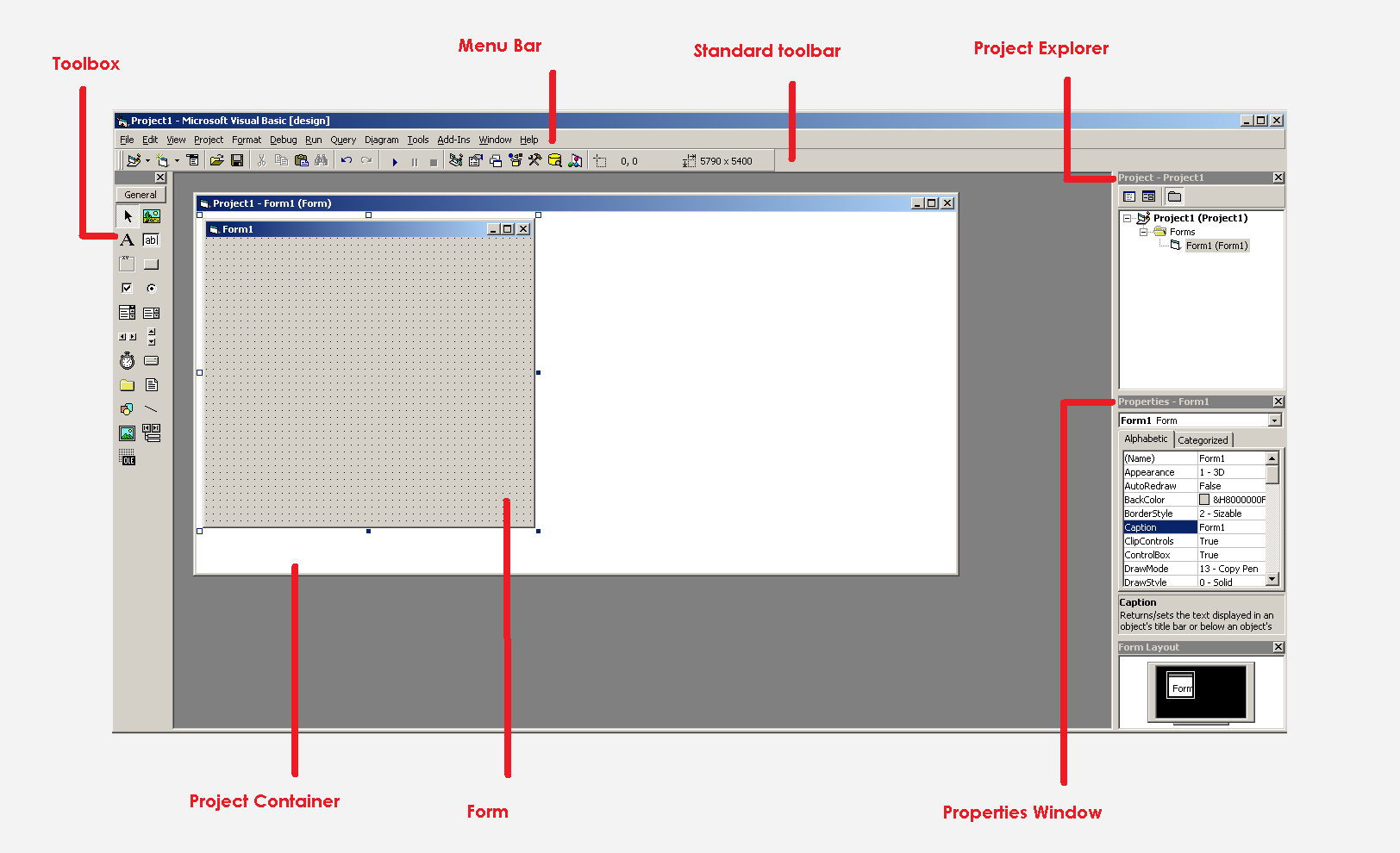
* Menu Bar
* Tool Bar
* Project Explorer
* Properties window
* Form Layout Window
* Toolbox
* Form Designer
* Object Browser

In previous versions of Visual Basic, the IDE was designed as a Single Document Interface (SDI). In a Single Document Interface, each window is a free-floating window that is contained within a main window and can move anywhere on the screen as long as Visual Basic is the current application. But, in Visual Basic 6.0, the IDE is in a Multiple Document Interface (MDI) format. In this format, the windows associated with the project will stay within a single container known as the parent. Code and form-based windows will stay within the main container form.

**The Visual Basic startup dialog box**



**The Integrated Development Environment**



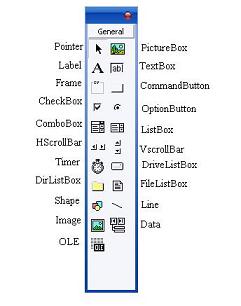
Menu Bar

This Menu Bar displays the commands that are required to build an application. The main menu items have sub menu items that can be chosen when needed. The toolbars in the menu bar provide quick access to the commonly used commands and a button in the toolbar is clicked once to carry out the action represented by it.

Toolbox

The Toolbox contains a set of controls that are used to place on a Form at design time thereby creating the user interface area. Additional controls can be included in the toolbox by using the Components menu item on the Project menu. A Toolbox is represented in figure 2 shown below.

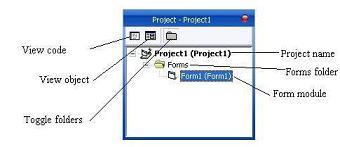
**Toolbox window with its controls available commonly.**



|  |  |
| --- | --- |
| **Control** | **Description** |
| **Pointer** | Provides a way to move and resize the controls form |
| **PictureBox** | Displays icons/bitmaps and metafiles. It displays text or acts as a visual container for other controls. |
| **TextBox** | Used to display message and enter text. |
| **Frame** | Serves as a visual and functional container for controls |
| **CommandButton** | Used to carry out the specified action when the user chooses it. |
| **CheckBox** | Displays a True/False or Yes/No option. |
| **OptionButton** | OptionButton control which is a part of an option group allows the user to select only one option even it displays mulitiple choices. |
| **ListBox** | Displays a list of items from which a user can select one. |
| **ComboBox** | Contains a TextBox and a ListBox. This allows the user to select an ietm from the dropdown ListBox, or to type in a selection in the TextBox. |
| **HScrollBar and VScrollBar** | These controls allow the user to select a value within the specified range of values |
| **Timer** | Executes the timer events at specified intervals of time |
| **DriveListBox** | Displays the valid disk drives and allows the user to select one of them. |
| **0DirListBox** | Allows the user to select the directories and paths, which are displayed. |
| **FileListBox** | Displays a set of files from which a user can select the desired one. |
| **Shape** | Used to add shape (rectangle, square or circle) to a Form |
| **Line** | Used to draw straight line to the Form |
| **Image** | used to display images such as icons, bitmaps and metafiles. But less capability than the PictureBox |
| **Data** | Enables the use to connect to an existing database and display information from it. |
| **OLE** | Used to link or embed an object, display and manipulate data from other windows based applications. |
| **Label** | Displays a text that the user cannot modify or interact with. |

Project Explorer

Docked on the right side of the screen, just under the tollbar, is the Project Explorer window. The Project Explorer as shown in in figure servres as a quick reference to the various elements of a project namely *form*, *classes* and *modules*. All of the object that make up the application are packed in a project. A simple project will typically contain one form, which is a window that is designed as part of a program's interface. It is possible to develop any number of forms for use in a program, although a program may consist of a single form. In addition to forms, the Project Explorer window also lists code modules and classes.



**Properties Window**

**The Properties Window is docked under the Project Explorer window. The Properties Window exposes the various characteristics of selected objects. Each and every form in an application is considered an object. Now, each object in Visual Basic has characteristics such as color and size. Other characteristics affect not just the appearance of the object but the way it behaves too. All these characteristics of an object are called its properties. Thus, a form has properties and any controls placed on it will have propeties too. All of these properties are displayed in the Properties Window.**

**Object Browser**

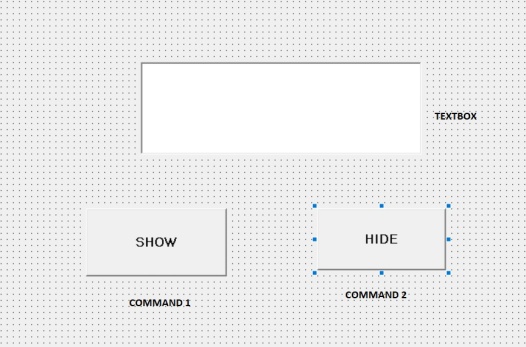
**The Object Browser allows us to browse through the various properties, events and methods that are made available to us. It is accessed by selecting Object Browser from the View menu or pressing the key F2. The left column of the Object Browser lists the objects and classes that are available in the projects that are opened and the controls that have been referenced in them. It is possible for us to scroll through the list and select the object or class that we wish to inspect. After an object is picked up from the Classes list, we can see its members (properties, methods and events) in the right column.**

**A property is represented by a small icon that has a hand holding a piece of paper. Methods are denoted by little green blocks, while events are denoted by yellow lightning bolt icon.**

**Object naming conversions of controls (prefix)**

**Form -frm  
Label-lbl  
TextBox-txt  
CommandButton-cmd  
CheckBox -chk  
OptionButton -opt  
ComboBox -cbo  
ListBox-lst  
Frame-fme  
PictureBox -pic  
Image-img  
Shape-shp  
Line -lin  
HScrollBar -hsb  
VScrollBar -vsb**

**A SIMPLE VISUAL BASIC SHOW HIDE PROGRAM**

DESIGN VIEW

CODE:

Private Sub Command1\_Click()

Text1.Text = "CHANDANNAGAR SADAR YCTC"

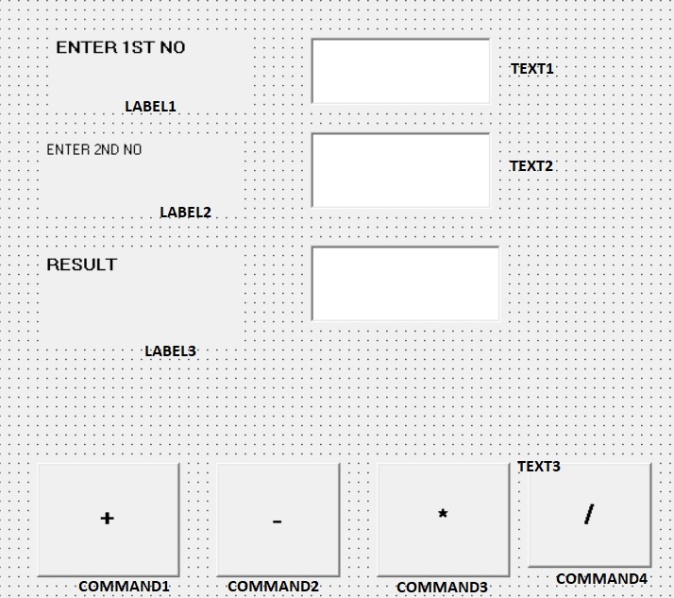
End Sub

Private Sub Command2\_Click()

Text1.Text = ""

End Sub

**SIMPLE CALCULATOR**



CODE:

Private Sub Command1\_Click()

Text3.Text = Val(Text1.Text) + Val(Text2.Text)

End Sub

Private Sub Command2\_Click()

Text3.Text = Val(Text1.Text) - Val(Text2.Text)

End Sub

Private Sub Command3\_Click()

Text3.Text = Val(Text1.Text) \* Val(Text2.Text)

End Sub

Private Sub Command4\_Click()

Text3.Text = Val(Text1.Text) / Val(Text2.Text)

End Sub

**IF ELSE CONDITION**

IF CONDITION 1 THEN

STATEMENT 1

ELSEIF CONDITION 2 THEN

STATEMENT 2

ELSEIF CONDITION 3 THEN

STATEMENT 3

ELSEIF CONDITION 4 THEN

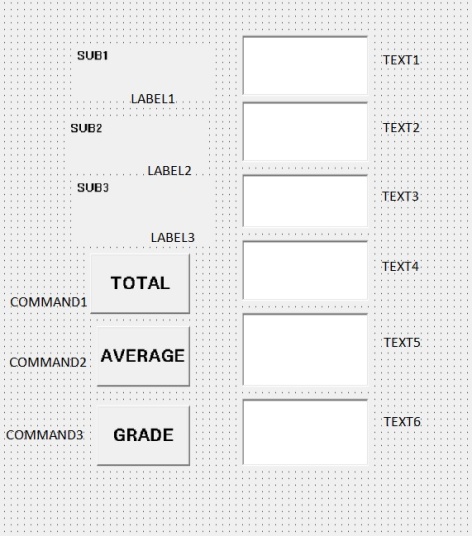
STATEMENT 4

ELSE

STATEMENT 5

END IF

**RESULT SHEET USING ELSE IF CONDITION**



CODE:

Private Sub Command1\_Click()

Text4.Text = Val(Text1.Text) + Val(Text2.Text) + Val(Text3.Text)

End Sub

Private Sub Command2\_Click()

Text5.Text = Val(Text4.Text) / 3

End Sub

Private Sub Command3\_Click()

If Val(Text5.Text) > 90 Then

Text6.Text = "A+"

ElseIf VAL(Text5.Text) > 80 Then

Text6.Text = "A"

ElseIf VAL(Text5.Text) > 70 Then

Text6.Text = "B+"

ElseIf VAL(Text5.Text) > 60 Then

Text6.Text = "B"

ElseIf VAL(Text5.Text) > 40 Then

Text6.Text = "C"

ElseIf VAL(Text5.Text) > 30 Then

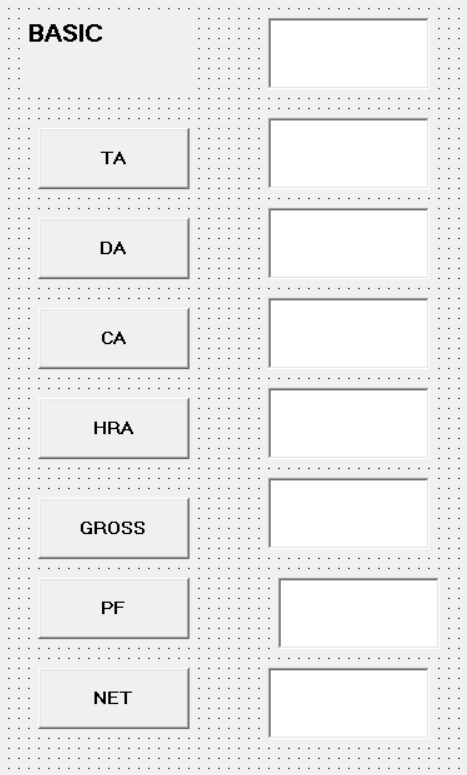
Text6.Text = "D"

Else

Text6.Text = "FAIL"

End If

End Sub

**MAKE A SALARY SHEET USING VISUAL BASIC**

CODE:

Private Sub Command1\_Click()

Text2.Text = Val(Text1.Text) \* 0.07

End Sub

Private Sub Command2\_Click()

Text3.Text = Val(Text1.Text) \* 0.58

End Sub

Private Sub Command3\_Click()

Text4.Text = Val(Text1.Text) \* 0.11

End Sub

Private Sub Command4\_Click()

Text5.Text = Val(Text1.Text) \* 0.23

End Sub

Private Sub Command5\_Click()

Text6.Text = Val(Text1.Text) + Val(Text2.Text) + Val(Text3.Text) + Val(Text4.Text) + Val(Text5.Text)

End Sub

Private Sub Command6\_Click()

Text7.Text = Val(Text1.Text) \* 0.03

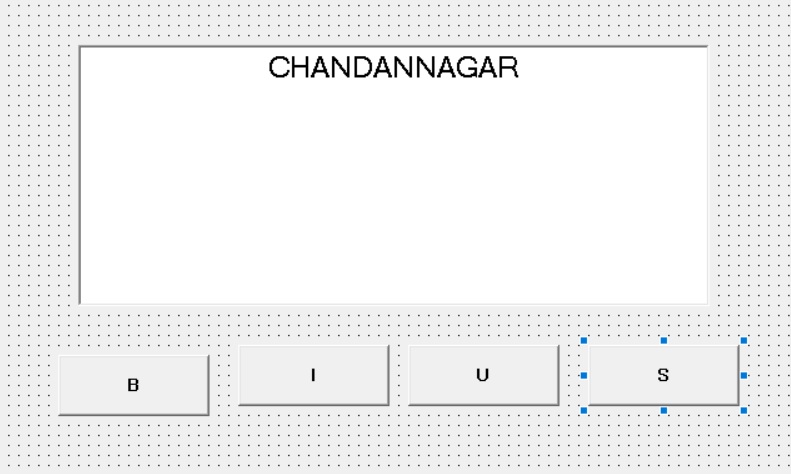
End Sub

Private Sub Command7\_Click()

Text8.Text = Val(Text6.Text) - Val(Text7.Text)

End Sub

**BOLD ITALIC UNDERLINE STRIKETHRU**



CODE:

Private Sub Command1\_Click()

If Text1.FontBold = True Then

Text1.FontBold = False

Else

Text1.FontBold = False

End If

End Sub

Private Sub Command2\_Click()

If Text1.FontItalic = True Then

Text1.FontItalic = False

Else

Text1.FontItalic = False

End If

End Sub

Private Sub Command3\_Click()

If Text1.FontUnderline = True Then

Text1.FontUnderline = False

Else

Text1.FontUnderline = False

End If

End Sub

Private Sub Command4\_Click()

If Text1.FontStrikethru = True Then

Text1.FontStrikethru = False

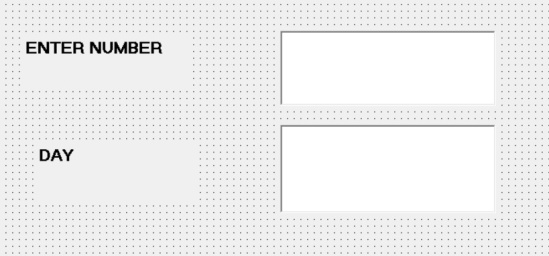
Else

Text1.FontStrikethru = False

End If

End Sub

**2ND PROGRAM BY IF ELSE STATEMENT**



CODE:

Private Sub Text1\_Change()

Dim day As Integer

day = Val(Text1.Text)

If day = 1 Then

Text2.Text = "Monday"

ElseIf day = 2 Then

Text2.Text = "Tuesday"

ElseIf day = 3 Then

Text2.Text = "Wednesday"

ElseIf day = 4 Then

Text2.Text = "Thrusday"

ElseIf day = 5 Then

Text2.Text = "Friday"

ElseIf day = 6 Then

Text2.Text = "Saturday"

ElseIf day = 7 Then

Text2.Text = "Sunday"

Else

Text2.Text = "Enter Number Less Than Equal To 7"

End If

End Sub

**Modules**

**Code in Visual Basic is stored in the form of modules. The three kind of modules are Form Modules, Standard Modules and Class Modules. A simple application may contain a single Form, and the code resides in that Form module itself. As the application grows, additional Forms are added and there may be a common code to be executed in several Forms. To avoid the duplication of code, a separate module containing a procedure is created that implements the common code. This is a standard Module.**

**Class module (.CLS filename extension) are the foundation of the object oriented programming in Visual Basic. New objects can be created by writing code in class modules. Each module can contain:**

**Declarations : May include constant, type, variable and DLL procedure declarations.**

**Procedures : A sub function, or property procedure that contain pieces of code that can be executed as a unit.**

**These are the rules to follow when naming elements in VB - variables, constants, controls, procedures, and so on:**

* A name must begin with a letter.
* May be as much as 255 characters long (but don't forget that somebody has to type the stuff!).
* Must not contain a space or an embedded period or type-declaration characters used to specify a data type; these are ! # % $ & @
* Must not be a reserved word (that is part of the code, like Option, for example)
* The dash, although legal, should be avoided because it may be confused with the minus sign. Instead of First-name use First\_name or FirstName**.**

**Data types in Visual Basic 6**

**By default Visual Basic variables are of variant data types. The variant data type can store numeric, date/time or string data. When a variable is declared, a data type is supplied for it that determines the kind of data they can store. The fundamental data types in Visual Basic including variant are integer, long, single, double, string, currency, byte and boolean. Visual Basic supports a vast array of data types. Each data type has limits to the kind of information and the minimum and maximum values it can hold. In addition, some types can interchange with some other types. A list of Visual Basic's simple data types are given below.**

**1. Numeric**

|  |  |
| --- | --- |
| **Byte** | **Store integer values in the range of 0 - 255** |
| **Integer** | **Store integer values in the range of (-32,768) - (+ 32,767)** |
| **Long** | **Store integer values in the range of (- 2,147,483,468) - (+ 2,147,483,468)** |
| **Single** | **Store floating point value in the range of (-3.4x10-38) - (+ 3.4x1038)** |
| **Double** | **Store large floating value which exceeding the single data type value** |
| **Currency** | **store monetary values. It supports 4 digits to the right of decimal point and 15 digits to the left** |

**2. String**

**Use to store alphanumeric values. A variable length string can store approximately 4 billion characters**

**3. Date**

**Use to store date and time values. A variable declared as date type can store both date and time values and it can store date values 01/01/0100 up to 12/31/9999**

**4. Boolean**

**Boolean data types hold either a true or false value. These are not stored as numeric values and cannot be used as such. Values are internally stored as -1 (True) and 0 (False) and any non-zero value is considered as true.**

**5. Variant**

**Stores any type of data and is the default Visual Basic data type. In Visual Basic if we declare a variable without any data type by default the data type is assigned as default.**

**Operators in Visual Basic**

**Arithmetical Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operators** | **Description** | **Example** | **Result** |
| + | Add | 5+5 | 10 |
| - | Substract | 10-5 | 5 |
| / | Divide | 25/5 | 5 |
| \ | Integer Division | 20\3 | 6 |
| \* | Multiply | 5\*4 | 20 |
| ^ | Exponent (power of) | 3^3 | 27 |
| Mod | Remainder of division | 20 Mod 6 | 2 |
| & | String concatenation | "George"&" "&"Bush" | "George Bush" |

**Relational Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operators** | **Description** | **Example** | **Result** |
| > | Greater than | 10>8 | True |
| < | Less than | 10<8 | False |
| >= | Greater than or equal to | 20>=10 | True |
| <= | Less than or equal to | 10<=20 | True |
| <> | Not Equal to | 5<>4 | True |
| = | Equal to | 5=7 | False |

**Logical Operators**

|  |  |
| --- | --- |
| **And** | **Both sides must be true** |
| **Or** | **One side or other must be true** |
| **Xor** | **One side or other must be true but not both** |
| **Not** | **Negates true** |
| **And** | **Both sides must be true** |

**Variables In Visual Basic 6**

Variables are the memory locations which are used to store values temporarily. A defined naming strategy has to be followed while naming a variable. A variable name must begin with an alphabet letter and should not exceed 255 characters. It must be unique within the same scope. It should not contain any special character like %, &, !, #, @ or $.

There are many ways of declaring variables in Visual Basic. Depending on where the variables are declared and how they are declared, we can determine how they can be used by our application. The different ways of declaring variables in Visual Basic are listed below

1.Explicit Declaration

2.Using Option Explicit statement

3.Scope of Variables

**Explicit Declaration**

Declaring a variable tells Visual Basic to reserve space in memory. It is not must that a variable should be declared before using it. Automatically whenever Visual Basic encounters a new variable, it assigns the default variable type and value. This is called implicit declaration. Though this type of declaration is easier for the user, to have more control over the variables, it is advisable to declare them explicitly. The variables are declared with a Dim statement to name the variable and its type. The As type clause in the Dim statement allows to define the data type or object type of the variable. This is called explicit declaration.

Syntax

Dim variable [As Type]

For example,

Dim strName As String

Dim intCounter As Integer

Using Option Explicit statement

It may be convenient to declare variables implicitly, but it can lead to errors that may not be recognized at run time. Say, for example a variable by name intcount is used implicitly and is assigned to a value. In the next step, this field is incremented by 1 by the following statement

Intcount = Intcount + 1

This calculation will result in intcount yielding a value of 1 as intcount would have been initialized to zero. This is because the intcount variable has been mityped as incont in the right hand side of the second variable. But Visual Basic does not see this as a mistake and considers it to be new variable and therefore gives a wrong result.

In Visual Basic, to prevent errors of this nature, we can declare a variable by adding the following statement to the general declaration section of the Form.

**Option Explicit**

This forces the user to declare all the variables. The Option Explicit statement checks in the module for usage of any undeclared variables and reports an error to the user. The user can thus rectify the error on seeing this error message.

The Option Explicit statement can be explicitly placed in the general declaration section of each module using the following steps.

Click Options item in the Tools menu

Click the Editor tab in the Options dialog box

Check Require Variable Declaration option and then click the OK button

**Scope of variables**

A variable is scoped to a procedure-level (local) or module-level variable depending on how it is declared. The scope of a variable, procedure or object determines which part of the code in our application are aware of the variable's existence. A variable is declared in general declaration section of e Form, and hence is available to all the procedures. Local variables are recognized only in the procedure in which they are declared. They can be declared with Dim and Static keywords. If we want a variable to be available to all of the procedures within the same module, or to all the procedures in an application, a variable is declared with broader scope.

**Local Variables**

A local variable is one that is declared inside a procedure. This variable is only available to the code inside the procedure and can be declared using the Dim statements as given below.

Dim sum As Integer

The local variables exist as long as the procedure in which they are declared, is executing. Once a procedure is executed, the values of its local variables are lost and the memory used by these variables is freed and can be reclaimed. Variables that are declared with keyword Dim exist only as long as the procedure is being executed.

**Static Variables**

Static variables are not reinitialized each time Visual Invokes a procedure and therefore retains or preserves value even when a procedure ends. In case we need to keep track of the number of times a command button in an application is clicked, a static counter variable has to be declared. These static variables are also ideal for making controls alternately visible or invisible. A static variable is declared as given below.

Static intPermanent As Integer

**Module Levele Variables**

A module level variable is available to all the procedures in the module. They are declared using the Public or the Private keyword. If you declare a variable using a Private or a Dim statement in the declaration section of a module—a standard BAS module, a form module, a class module, and so on—you're creating a private module-level variable. Such variables are visible only from within the module they belong to and can't be accessed from the outside. In general, these variables are useful for sharing data among procedures in the same module:

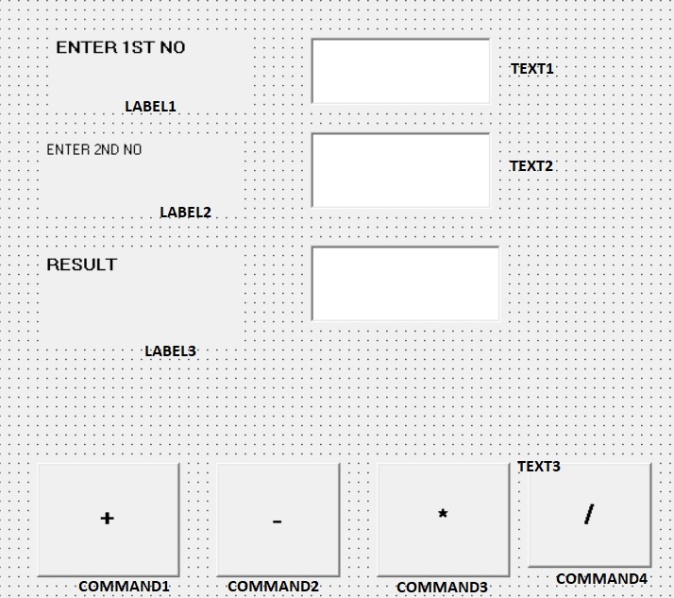
Private LoginTime As Date ' A private module-level variable

Dim LoginPassword As String ' Another private module-level variable

Public vs Local Variables

A variable can have the same name and different scope. For example, we can have a public variable named R and within a procedure we can declare a local variable R. References to the name R within the procedure would access the local variable and references to R outside the procedure would access the public variable.

**A SIMPLE CALCULATOR USING VARIABLE**



CODE:

DIM A,B,C AS INTEGER

Private Sub Command1\_Click()

C=A+B

Text3.Text =C

End Sub

Private Sub Command2\_Click()

C=A-B

Text3.Text =C

End Sub

Private Sub Command3\_Click()

C=A\*B

Text3.Text =C

End Sub

Private Sub Command4\_Click()

C=A/B

Text3.Text =C

End Sub

Private Sub Text1\_Change()

A=Val(Text1.Text)

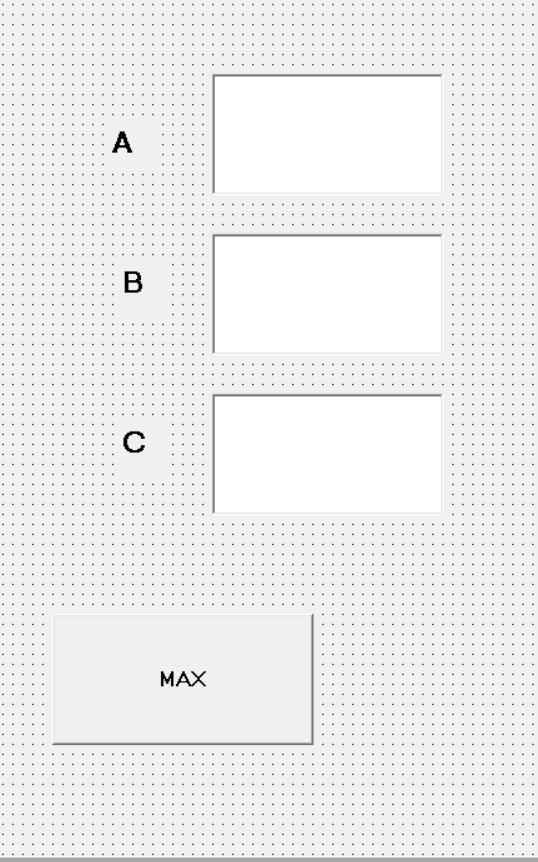
End Sub

Private Sub Text2\_Change()

B=Val(Text2.Text)

End Sub

**FIND THE MAXIMUM NUMBER**



CODE:

Dim A, B, C As Integer

Private Sub Command1\_Click()

If A > B And A > C Then

MsgBox " A IS MAXIMUM"

ElseIf B > A And B > C Then

MsgBox " B IS MAXIMUM"

Else

MsgBox " C IS MAXIMUM"

End If

End Sub

Private Sub Text1\_Change()

A = Val(Text1.Text)

End Sub

Private Sub Text2\_Change()

B = Val(Text2.Text)

End Sub

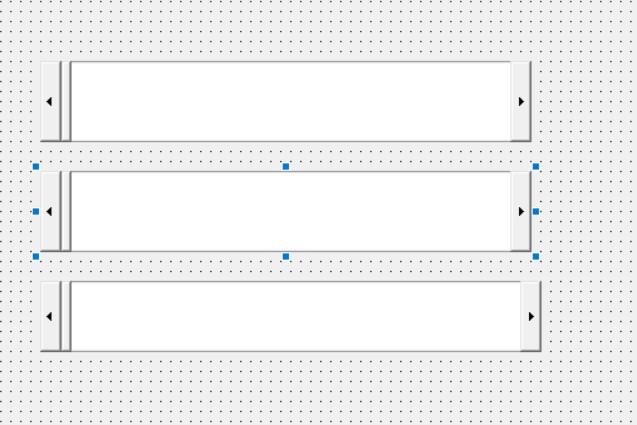
Private Sub Text3\_Change()

C = Val(Text3.Text)

End Sub

**MAKE A PROGRAM TO CHANGE THE BACKCOLOR OF A FORM USING HSCROLBAR**

**(RGB=RED,GREEN,BLUE)**

****

CODE:

Dim X, Y, Z As Integer

Private Sub HScroll1\_Change()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Form1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll1\_Scroll()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Form1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll2\_Change()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Form1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll2\_Scroll()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Form1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll3\_Change()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Form1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll3\_Scroll()

X = HScroll1.Value

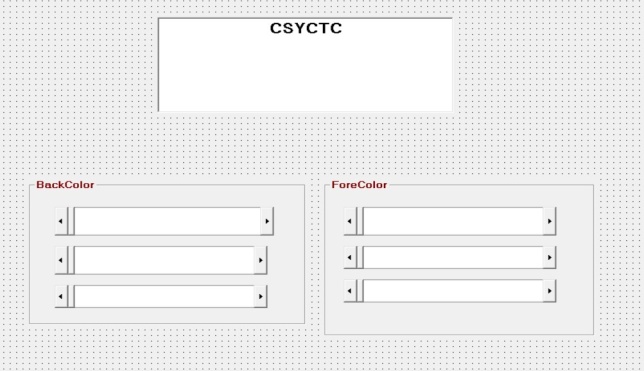
Y = HScroll2.Value

Z = HScroll3.Value

Form1.BackColor = RGB(X, Y, Z)

End Sub

**MAKE A PROGRAM TO CHANGE THE BACKCOLOR AND FORECOLOR OF A TEXTBOX USING HSCROLLBAR**

****

CODE:

Dim X, Y, Z As Integer

Private Sub HScroll1\_Change()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll1\_Scroll()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll2\_Change()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll2\_Scroll()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll3\_Change()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll3\_Scroll()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll4\_Change()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll4\_Scroll()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll5\_Change()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll5\_Scroll()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll6\_Change()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll6\_Scroll()

X = HScroll4.Value

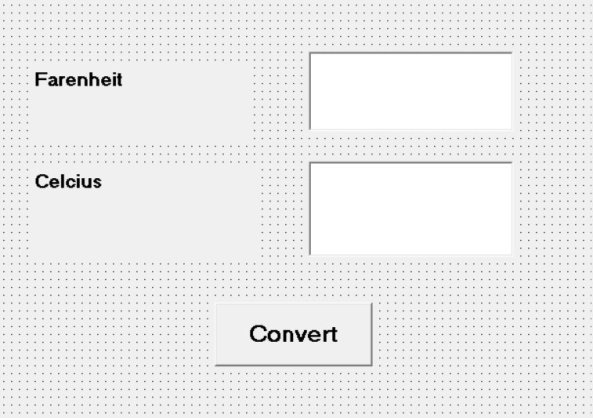
Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

**MAKE A PROGRAM TO CHANGE FARENHEIT TO CELCIUS**



CODE:

Dim c, f As Double

Private Sub Command1\_Click()

c = (5 \* f - 160) / 9

Text2.Text = c

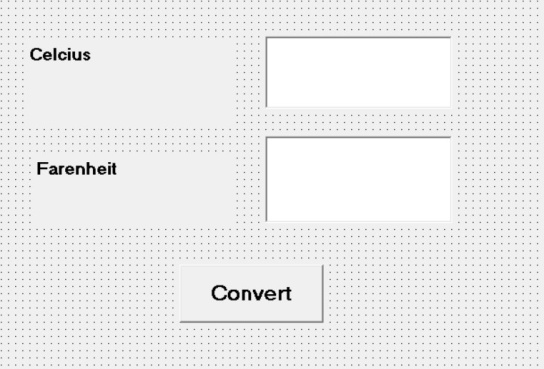
End Sub

Private Sub Text1\_Change()

f = Val(Text1.Text)

End Sub

**MAKE A PROGRAM TO CHANGE CELCIUS TO FARENHEIT**



CODE:

Dim c, f As Double

Private Sub Command1\_Click()

f = c \* 1.8 + 32

Text2.Text = f

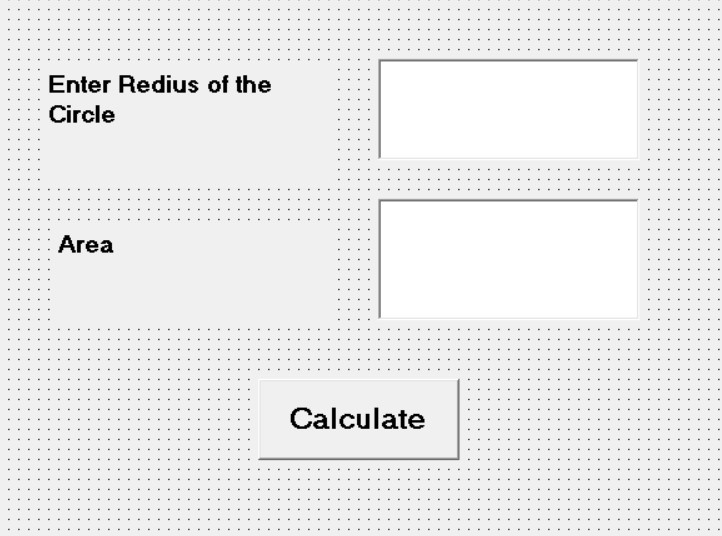
End Sub

Private Sub Text1\_Change()

c = Val(Text1.Text)

End Sub

**MAKE A PROGRAM TO CALCULATE THE AREA OF A CIRCLE**



CODE:

Dim r, a As Double

Private Sub Command1\_Click()

a = (22 / 7) \* r \* r

Text2.Text = a

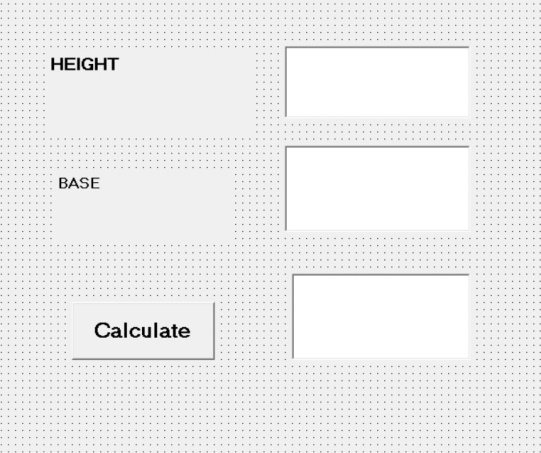
End Sub

Private Sub Text1\_Change()

r = Val(Text1.Text)

End Sub

**MAKE A PROGRAM TO CALCULATE THE AREA OF A TRIANGLE**



CODE:

Dim AREA, HIG, BASE As Double

Private Sub Command1\_Click()

AREA = (HIG \* BASE) / 2

Text3.Text = AREA

End Sub

Private Sub Text1\_Change()

HIG = Val(Text1.Text)

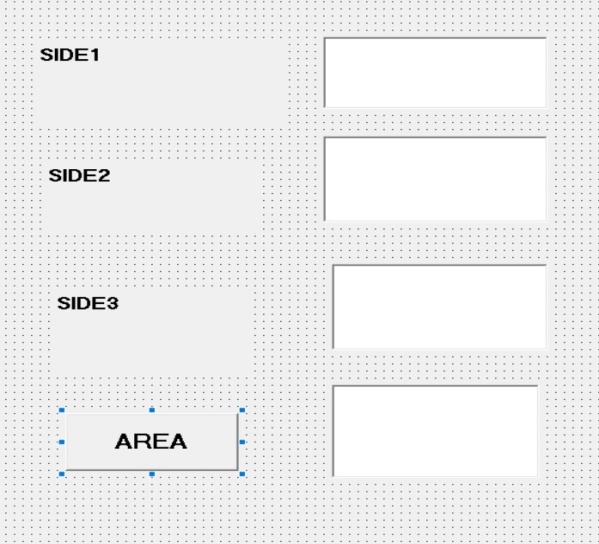
End Sub

Private Sub Text2\_Change()

BASE = Val(Text2.Text)

End Sub

**MAKE A PROGRAM TO CALCULATE THE AREA OF A TRIANGLE WHERE THREE SIDES ARE GIVEN**

****

CODE:

Dim S1, S2, S3, AREA As Double

Private Sub Command1\_Click()

AREA = (S1 + S2 + S3) / 2

Text4.Text = AREA

End Sub

Private Sub Text1\_Change()

S1 = Val(Text1.Text)

End Sub

Private Sub Text2\_Change()

S2 = Val(Text2.Text)

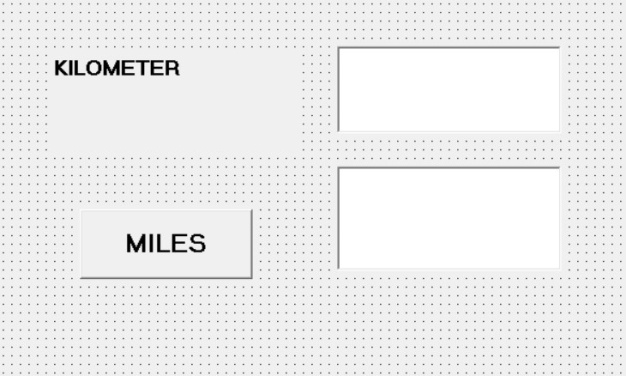
End Sub

Private Sub Text3\_Change()

S3 = Val(Text3.Text)

End Sub

**MAKE A PROGRAM TO CONVERT KILOMETER TO MILES**

****

CODE:

Dim K, M As Double

Private Sub Command1\_Click()

M = k \* 0.62137

Text2.Text = M

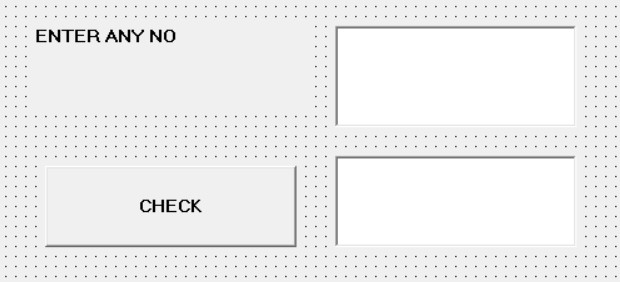
End Sub

Private Sub Text1\_Change()

K = Val(Text1.Text)

End Sub

**MAKE A PROGRAM TO CHECK ODD OR EVEN NO**



**CODE:**

Dim NUM As Integer

Private Sub Command1\_Click()

If NUM Mod 2 = 0 Then

Text2.Text = "THIS IS EVEN NO"

Else

Text2.Text " THIS IS ODD NO"

End If

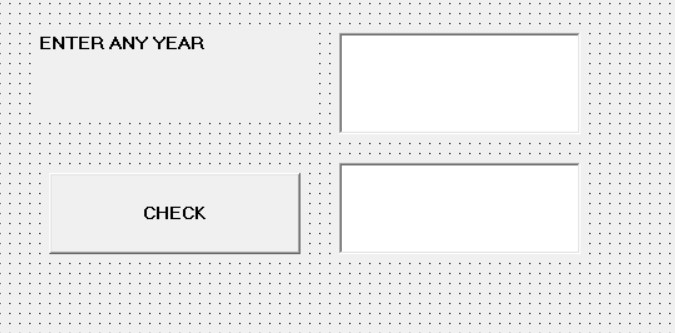
End Sub

Private Sub Text1\_Change()

NUM = Val(Text1.Text)

End Sub

**MAKE A PROGRAM TO CHECK A YEAR LEAP YEAR OR NOT**

****

**CODE:**

**Dim Y As Integer**

**Private Sub Command1\_Click()**

**If Y Mod 4 = 0 Or Y Mod 100 = 0 Then**

**Text2.Text = "LEAP YEAR"**

**Else**

**Text2.Text = " NOT LEAP YEAR"**

**End If**

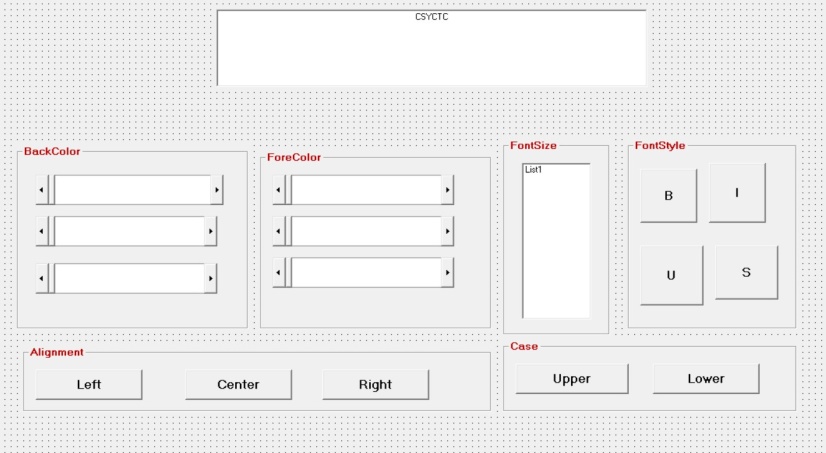
**End Sub**

**Private Sub Text1\_Change()**

**Y = Val(Text1.Text)**

**End Sub**

**A SIMPLE PROJECT**

****

CODE:

Dim i, X, Y, Z As Integer

Private Sub Command1\_Click()

If Text1.FontBold = True Then

Text1.FontBold = False

Else

Text1.FontBold = True

End If

End Sub

Private Sub Command2\_Click()

If Text1.FontItalic = True Then

Text1.FontItalic = False

Else

Text1.FontItalic = True

End If

End Sub

Private Sub Command3\_Click()

If Text1.FontUnderline = True Then

Text1.FontUnderline = False

Else

Text1.FontUnderline = True

End If

End Sub

Private Sub Command4\_Click()

If Text1.FontStrikethru = True Then

Text1.FontStrikethru = False

Else

Text1.FontStrikethru = True

End If

End Sub

Private Sub Command5\_Click()

Text1.Alignment = 0

End Sub

Private Sub Command6\_Click()

Text1.Alignment = 2

End Sub

Private Sub Command7\_Click()

Text1.Alignment = 1

End Sub

Private Sub Command8\_Click()

Text1.Text = UCase(Text1.Text)

End Sub

Private Sub Command9\_Click()

Text1.Text = LCase(Text1.Text)

End Sub

Private Sub Form\_Load()

For i = 8 To 72 Step 2

List1.AddItem i

Next

End Sub

Private Sub HScroll1\_Change()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll1\_Scroll()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll2\_Change()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll2\_Scroll()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll3\_Change()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll3\_Scroll()

X = HScroll1.Value

Y = HScroll2.Value

Z = HScroll3.Value

Text1.BackColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll4\_Change()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll4\_Scroll()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll5\_Change()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll5\_Scroll()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll6\_Change()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

End Sub

Private Sub HScroll6\_Scroll()

X = HScroll4.Value

Y = HScroll5.Value

Z = HScroll6.Value

Text1.ForeColor = RGB(X, Y, Z)

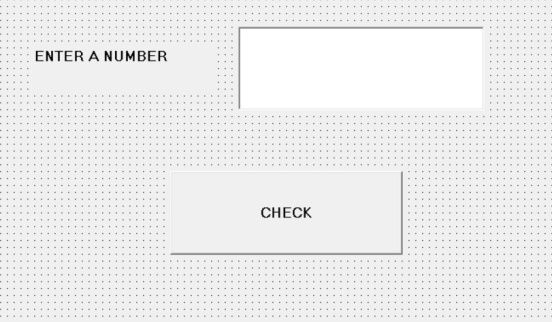
End Sub

Private Sub List1\_Click()

Text1.FontSize = List1.Text

End Sub

**MAKE A PROGRAM TO CHECK A NUMBER POSITIVE OR NEGATIVE**

****

CODE:

Dim A As Integer

Private Sub Command1\_Click()

If A > 0 Then

MsgBox " POSITIVE NO"

ElseIf A < 0 Then

MsgBox " NEGATIVE NO"

Else

MsgBox " ZERO"

End If

End Sub

Private Sub Text1\_Change()

A = Val(Text1.Text)

End Sub

LOOP:- **Visual Basic loop structures** allow you to run one or more lines of code repetitively. You can repeat the statements in a **loop structure** until a condition is True , until a condition is False , a specified number of times, or once for each element in a collection.

**A repetition structure allows the programmer to that an action is to be repeated until given condition is true.**

**Do While... Loop Statement**

**The Do While...Loop is used to execute statements until a certain condition is met. The following Do Loop counts from 1 to 100.**

**Dim number As Integer  
number = 1  
Do While number <= 100  
number = number + 1  
Loop**

**A variable number is initialized to 1 and then the Do While Loop starts. First, the condition is tested; if condition is True, then the statements are executed. When it gets to the Loop it goes back to the Do and tests condition again. If condition is False on the first pass, the statements are never executed.**

**While... Wend Statement**

**A While...Wend statement behaves like the Do While...Loop statement. The following While...Wend counts from 1 to 100**

**Dim number As Integer  
  
number = 1  
While number <=100  
number = number + 1  
Wend**

**Do...Loop While Statement**

**The Do...Loop While statement first executes the statements and then test the condition after each execution. The following program block illustrates the structure:**

**Dim number As Long  
number = 0  
Do  
number = number + 1  
Loop While number < 201**

**The programs executes the statements between Do and Loop While structure in any case. Then it determines whether the counter is less than 501. If so, the program again executes the statements between Do and Loop While else exits the Loop.**

**Do Until...Loop Statement**

**Unlike the Do While...Loop and While...Wend repetition structures, the Do Until... Loop structure tests a condition for falsity. Statements in the body of a Do Until...Loop are executed repeatedly as long as the loop-continuation test evaluates to False.**

**An example for Do Until...Loop statement. The coding is typed inside the click event of the command button**

**Dim number As Long  
number=0  
Do Until number > 1000  
number = number + 1  
Print number  
Loop**

**Numbers between 1 to 1000 will be displayed on the form as soon as you click on the command button.**

**The For...Next Loop**

**The For...Next Loop is another way to make loops in Visual Basic. For...Next repetition structure handles all the details of counter-controlled repetition. The following loop counts the numbers from 1 to 100:**

**Dim x As Integer  
For x = 1 To 50  
Print x  
Next**

**In order to count the numbers from 1 yo 50 in steps of 2, the following loop can be used**

**For x = 1 To 50 Step 2  
Print x  
Next**

**The following loop counts numbers as 1, 3, 5, 7..etc  
  
The above coding will display numbers vertically on the form. In order to display numbers horizontally the following method can be used.**

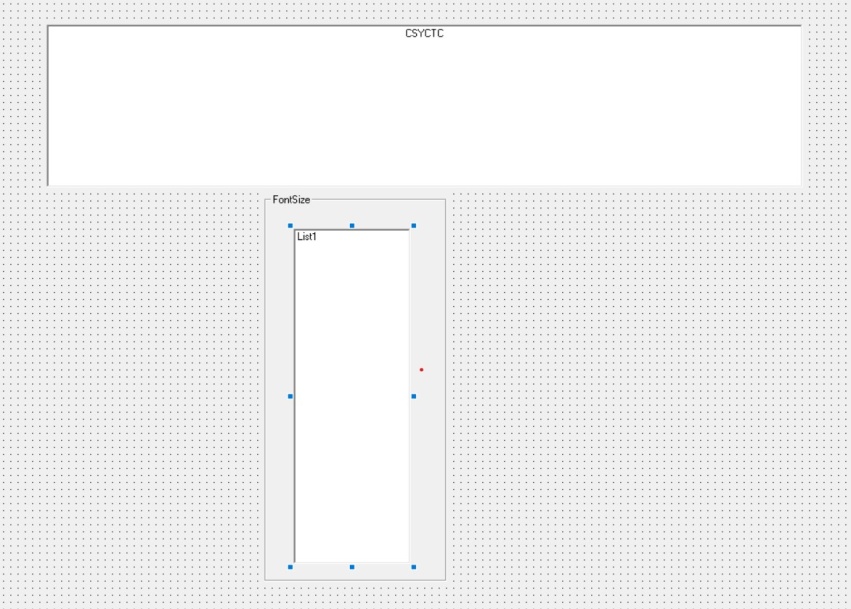
**For x = 1 To 50  
Print x & Space$ (2);  
Next**

**To increase the space between the numbers increase the value inside the brackets after the & Space$.**

**Following example is a For...Next repetition structure which is with the If condition used.**

**Dim number As Integer  
For number = 1 To 10  
If number = 4 Then  
Print "This is number 4"  
Else  
Print number  
End If  
Next**

**In the output instead of number 4 you will get the "This is number 4".**

**MAKE A PROGRAM TO CHANGE THE FONTSIZE OF A TEXTBOX**

CODE:

Private Sub Form\_Load()

Dim i As Integer

For i = 8 To 72 Step 2

List1.AddItem i

Next

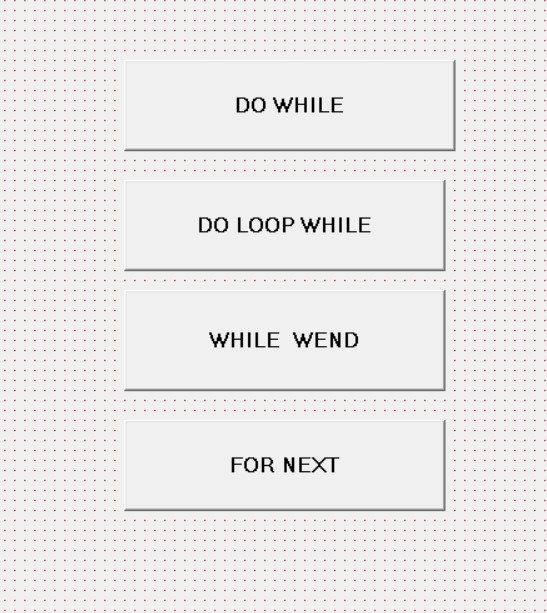
End Sub

Private Sub List1\_Click()

Text1.FontSize = List1.Text

End Sub

**LOOP EXAMPLE**



CODE:

Private Sub Command1\_Click()

Dim I As Integer

I = 1

Do While I <= 20

Print I;

I = I + 1

Loop

End Sub

Private Sub Command2\_Click()

Dim I As Integer

I = 1

Do

Print I;

I = I + 1

Loop While I <= 20

End Sub

Private Sub Command3\_Click()

Dim I As Integer

I = 1

While I <= 20

Print I;

I = I + 1

Wend

End Sub

Private Sub Command4\_Click()

Dim I As Integer

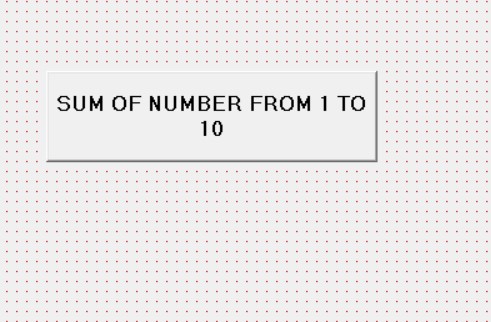
For I = 1 To 20

Print I;

Next

End Sub

**SUM OF NUMBER FROM 1 TO 10**

****

Private Sub Command1\_Click()

Dim I, J As Integer

For I = 1 To 10

J = J + I

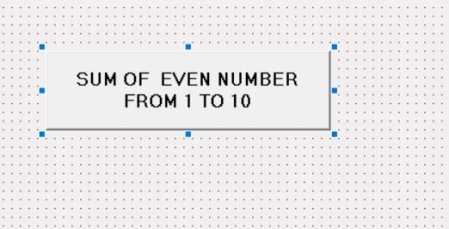
Next

MsgBox " SUM OF NUMBER IS " & J

End Sub

&🡺AMPERSAND

**SUM OF EVEN NUMBER FROM 1 TO 10**

****

Private Sub Command1\_Click()

Dim I, J As Integer

For I = 2 To 10 Step 2

J = J + I

Next

MsgBox " SUM OF NUMBER IS " & J

End Sub

**PATERN PRINT**

****

Private Sub Command1\_Click()

Dim I, J As Integer

For I = 1 To 5

For J = 1 To I

Print J;

Next

Print ""

Next

End Sub

**PATERN PRINT**

****

Private Sub Command1\_Click()

Dim I, J As Integer

For I = 1 To 5

For J = 1 To I

Print "\*";

Next

Print ""

Next

End Sub

**PATERN PRINT**

****

Private Sub Command1\_Click()

Dim I, J As Integer

For I = 5 To 1 Step -1

For J = 1 To I

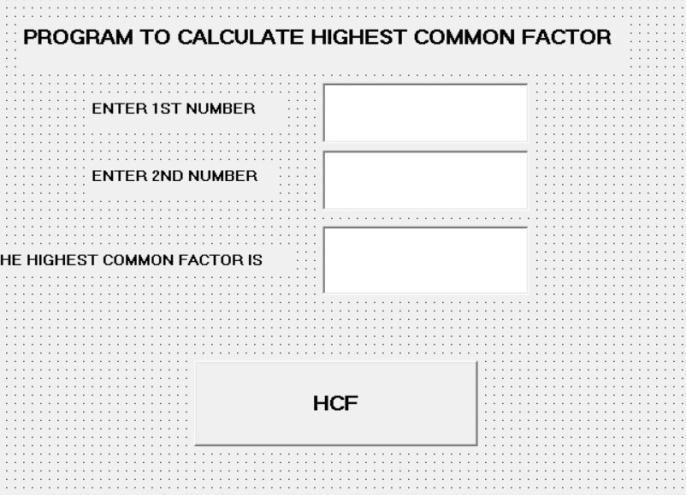
Print J;

Next

Print ""

Next

End Sub

**PROGRAM TO CALCULATE HIGHEST COMMON FACTORS**

Dim num1, num2, temp As Integer

Private Sub Command1\_Click()

While num2 <> 0

temp = num1 Mod num2

num1 = num2

num2 = temp

Wend

Text3.Text = num1

End Sub

Private Sub Text1\_Change()

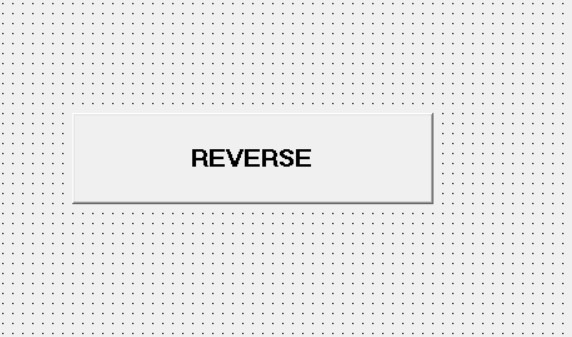
num1 = Val(Text1.Text)

End Sub

Private Sub Text2\_Change()

num2 = Val(Text2.Text)

End Sub

**REVERSE OF A NUMBER**

**CODE:**

**Private Sub Command1\_Click()**

**Dim N, D, R As Integer**

**N = Val(InputBox(" ENTER ANY NO"))**

**While N <> 0**

**D = N Mod 10**

**R = R \* 10 + D**

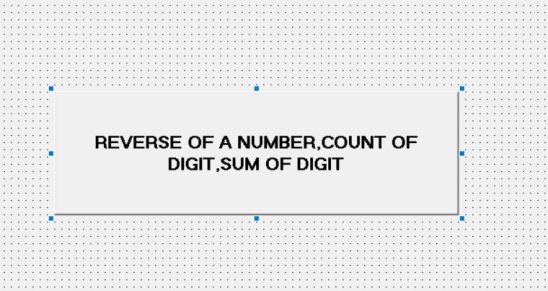
**N = N \ 10**

**Wend**

**MsgBox " REVERSE IS " & R**

**End Sub**

**PROGRAM TO REVERSE OF A NUMBER,COUNT OF DIGIT,SUM OF DIGIT**



**Private Sub Command1\_Click()**

**Dim N, D, R, S, C As Integer**

**N = Val(InputBox(" ENTER ANY NO"))**

**While N <> 0**

**D = N Mod 10**

**S = S + D**

**C = C + 1**

**R = R \* 10 + D**

**N = N \ 10**

**Wend**

**MsgBox " REVERSE IS " & R**

**MsgBox " SUM OF DIGIT IS " & S**

**MsgBox " COUNT OF DIGIT IS " & C**

**End Sub**

**MAKE A PROGRAM TO CHECK A TEXT PALINDROME OR NOT**

****

**CODE:**

**Private Sub Command1\_Click()**

**Dim N, N1, R As String**

**Dim I, L As Integer**

**N = InputBox("ENTER ANY TEXT")**

**N1 = N**

**L = Len(N)**

**For I = L To 1 Step -1**

**R = R + Mid(N, I, 1)**

**Next**

**MsgBox " REVERSE OF THE TEXT IS " & R**

**If Trim(N1) = Trim(R) Then**

**MsgBox " PALINDROME"**

**Else**

**MsgBox " NOT PALINDROME"**

**End If**

**End Sub**

**MAKE A PROGRAM TO CHECK A NUMBER AMSTRONG OR NOT**

****

Private Sub Command1\_Click()

Dim N, D, R, N1 As Integer

N = Val(InputBox("ENTER ANY Number "))

N1 = N

While N <> 0

D = N Mod 10

R = R + D ^ 3

N = N \ 10

Wend

If N1 = R Then

MsgBox " AMSTRONG NO"

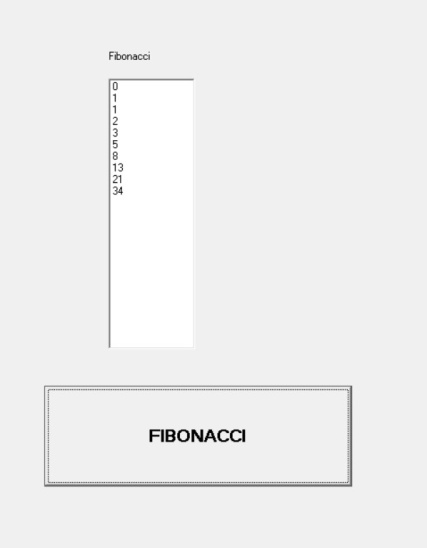
Else

MsgBox " NOT AMSTRONG NO"

End If

End Sub

**MAKE A PROGRAM TO PRINT FIBONACCI SERIES**

****

**Private Sub Command1\_Click()**

**Dim fno, sno, tno, i, L As Integer**

**fno = 0**

**sno = 1**

**L = Val(InputBox("enter limit"))**

**List1.AddItem fno**

**List1.AddItem sno**

**For i = 3 To L**

**tno = fno + sno**

**List1.AddItem tno**

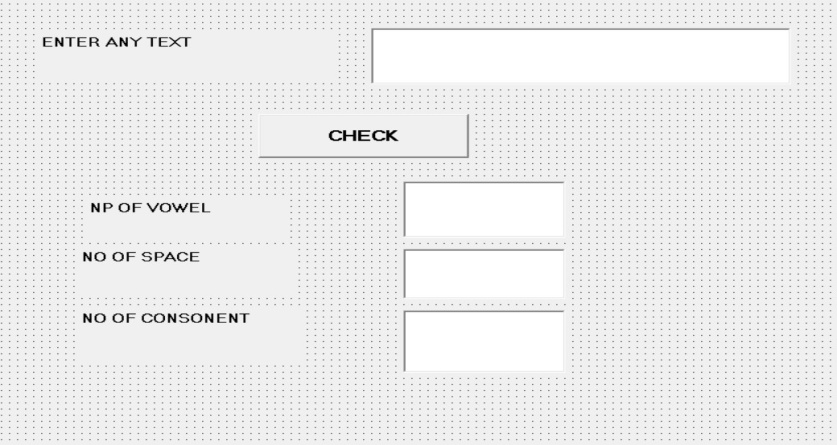
**fno = sno**

**sno = tno**

**Next**

**End Sub**

**MAKE A PROGRAM TO CHECK VOWEL CONSONENT AND SPACE FROM STRING**

****

**Dim V, C, S, I, L As Integer**

**Dim N, R As String**

**Private Sub Command1\_Click()**

**L = Len(Text1.Text)**

**For I = 1 To L**

**R = Mid(N, I, 1)**

**If R = "A" Or R = "E" Or R = "I" Or R = "O" Or R = "U" Or R = "a" Or R = "e" Or R = "i" Or R = "o" Or R = "u" Then**

**V = V + 1**

**End If**

**If R = " " Then**

**S = S + 1**

**End If**

**Next**

**C = L - S - V**

**Text2.Text = V**

**Text3.Text = S**

**Text4.Text = C**

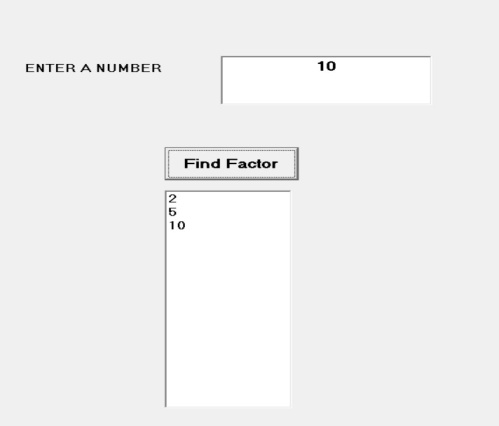
**End Sub**

**Private Sub Text1\_Change()**

**N = Text1.Text**

**End Sub**

**FIND FACTORS**

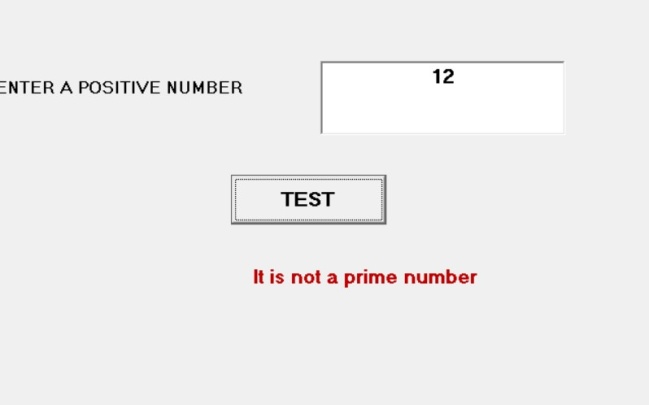
****

**CODE:**

**Private Sub Command1\_Click()  
Dim N, x As Integer  
N = Val(Text1.Text)  
For x = 2 To N - 1  
If N Mod x = 0 Then  
List1.AddItem x  
End If  
Next  
List1.AddItem N**

**End Sub**

**CHECK PRIME AND NOT PRIME NO**

****

Private Sub Command1\_Click()

Dim N, D As Single

Dim tag As String

N = Val(Text1.Text)

Select Case N

Case Is < 2

Label2.Caption = "It is not a prime number"

Case Is = 2

Label2.Caption = "It is a prime number"

Case Is > 2

D = 2

Do

If N / D = Int(N / D) Then

Label2.Caption = "It is not a prime number"

tag = "Not Prime"

Exit Do

End If

D = D + 1

Loop While D <= N - 1

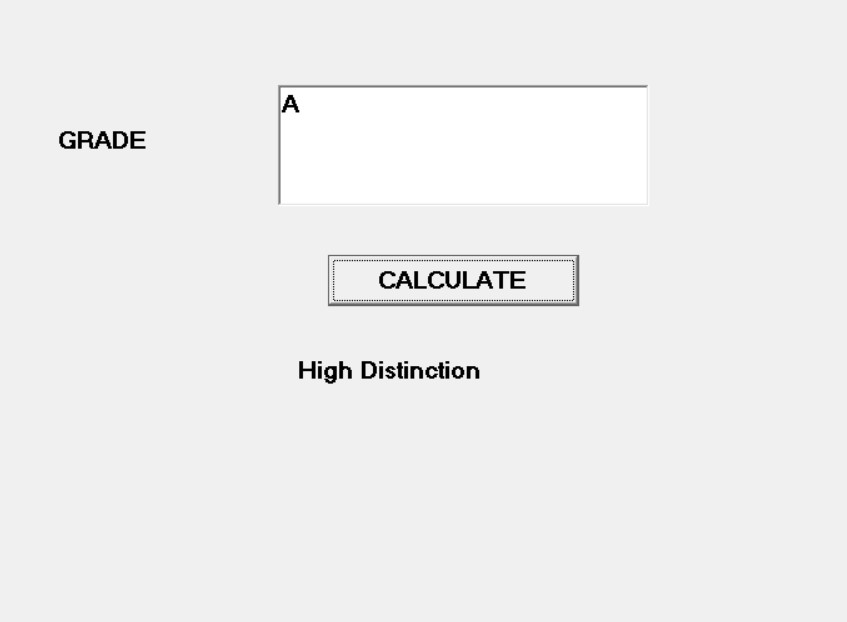
If tag <> "Not Prime" Then

Label2.Caption = "It is a prime number"

End If

End Select

End Sub



Dim grade As String

Private Sub Command1\_Click()

grade = Text1.Text

Select Case grade

Case "A"

Label1.Caption = "High Distinction"

Case "A-"

Label1.Caption = "Distinction"

Case "B"

Label1.Caption = "Credit"

Case "C"

Label1.Caption = "Pass"

Case Else

Label1.Caption = "Fail"

End Select

End Sub