



## Introduction to Computer programming languages

### 1. Introduction to computers

A computer is a multipurpose electronic device that can receive process and store data. They are used as tools in every part of society together with the Internet. Computers nowadays are complex; there are a lot of different components inside them, and they all serve different purposes. They all need to work together for the computer to work; knowing how a computer works makes it easier to use a computer by being able to understand how a computer will respond.

### Computer

A computer is an electronic device, operating under the control of instructions stored in its own memory that can accept data (input), process the data according to specified rules, produce information (output), and store the information for future use.



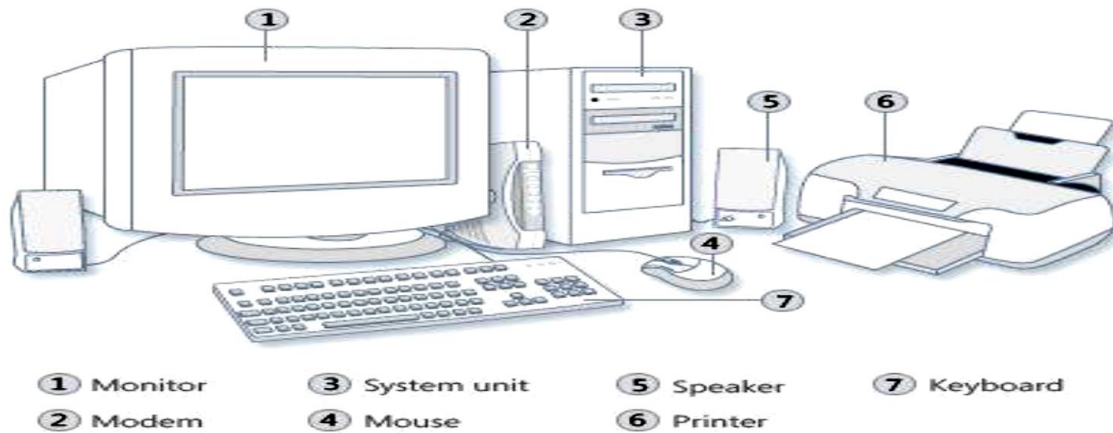
### Computer Components

Any kind of computers consists of **Hardware and Software**.

**Hardware:** Computer hardware is the collection of physical elements that constitutes a computer system. Computer hardware refers to the physical parts or components of a computer such as the

- ❖ **Input units:** Mouse, keyboard, scanner ..., etc.
- ❖ **Output units:** Monitor printer..., etc.
- ❖ **Storage units:** computer data storage, hard drive disk (HDD), flash... etc.
- ❖ **System unit:** graphic cards, sound cards, motherboard and chips. ..., etc.

❖ **Processing unit:** processor CPU.



## **Software**

Software is a generic term for organized collections of computer data and instructions, often broken into two major categories:

**System Software** that provides task specific functions of the computer. System software consists of an operating system and some fundamental utilities such as disk formatters, file managers, display managers, text editors, user authentication (login) and management tools, and networking and device control software.

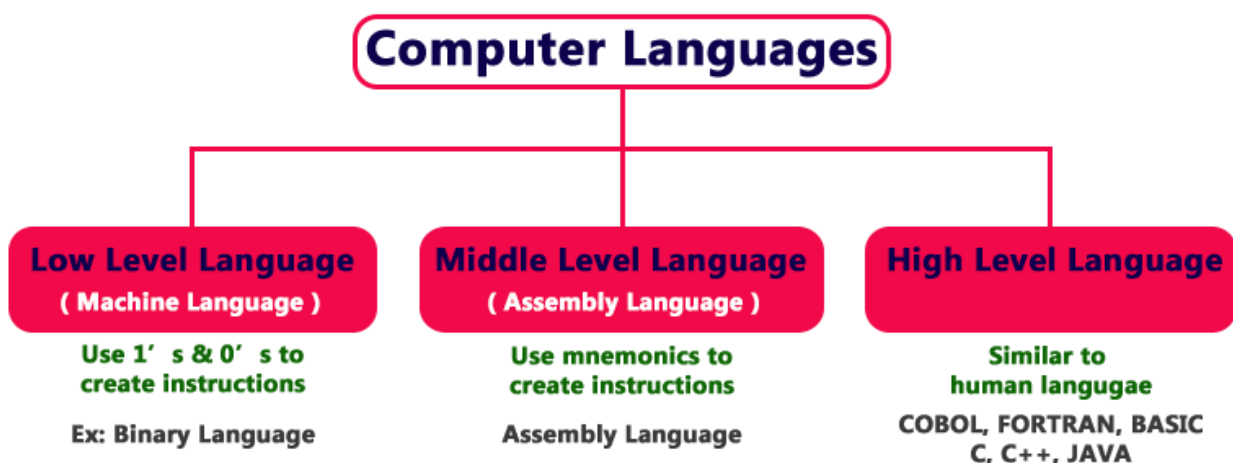
**Application Software** which is used by users to accomplish specific tasks. Application software may consist of a single program, such as an image viewer; such as Microsoft Office, a software system such as a database management system.

## **2. Computer programming language**

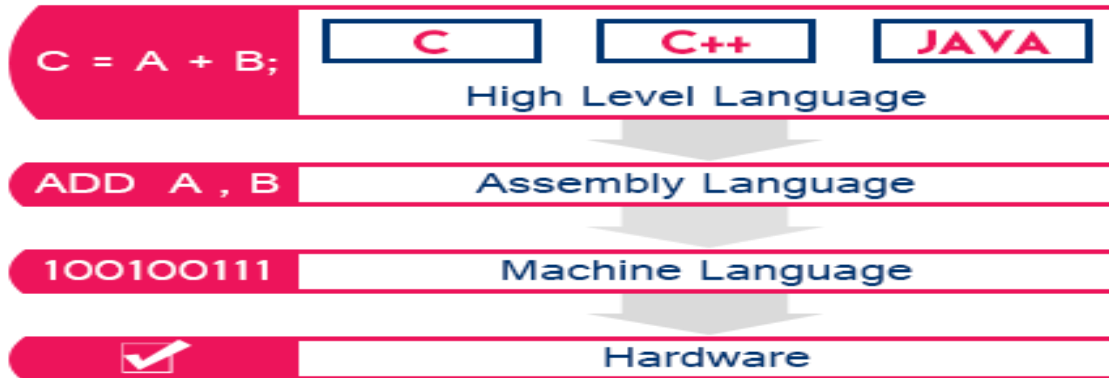
It is the process of designing and building an executable computer program for accomplishing a specific computing task. Programming involves tasks such as analysis, generating algorithms, profiling algorithms accuracy and resource consumption, and the implementation of algorithms in a chosen programming language (commonly referred to as coding). The source code of a program is written in one or more programming languages. The purpose of programming is to find a sequence of

instructions that will automate the performance of a task for solving a given problem. The process of programming thus often requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic. Programs are written either in one of high-level programming languages (such as BASIC, C, Java, pascal, matlab, python) which are easier but execute relatively slowly, or in one of low-level languages (assembly language or machine language) which are very complex but execute very fast.

Generally, we use languages like English, Hindi, and French etc., to make communication between two persons. That means, when we want to make communication between two persons we need a language through which persons can express their feelings. Similarly, when we want to make communication between user and computer or between two or more computers we need a language through which user can give information to computer and vice versa. When user wants to give any instruction to the computer the user needs a specific language and that language is known as computer language. User interacts with the computer using programs and that programs are created using computer programming languages like C, C++, Java, etc. Computer Languages are classified into three languages :

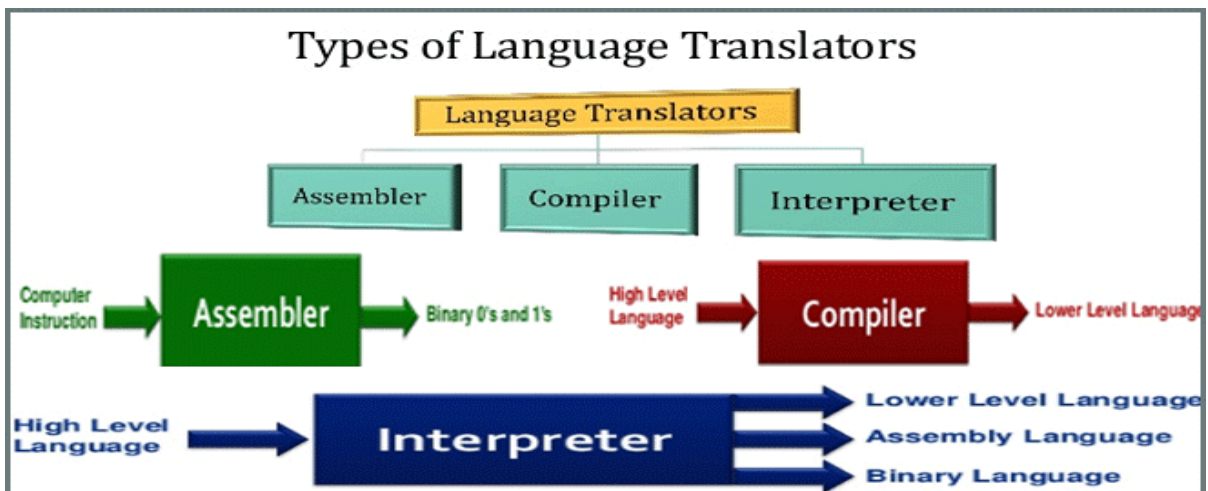


The following figure provides few key points related to the computer languages.



### 3. Language translators

We generally write a computer program using a high-level language. A high-level language is one which is understandable by us humans. It contains words and phrases from the English (or other) language. But a computer does not understand high-level language. It only understands program written in 0's and 1's in binary, called the machine code. A program written in high-level language is called a source code. We need to convert the source code into machine code and this is accomplished by compilers and interpreters. Hence, a compiler or an interpreter is a program that converts program written in high-level language into machine code understood by the computer.



#### **4. A computer program**

In the form of a human-readable, computer programming language is called source code. Translators check program syntax during the translation process. The syntax of a computer language is the set of rules that defines the combinations of symbols that are considered to be a correctly structured document or fragment in that language.

Computer language syntax is generally distinguished into three levels:

- Words – the lexical level, determining how characters form tokens;
- Phrases – the grammar level, narrowly speaking, determining how tokens form phrases;
- Context – determining what objects or variables names refer to, if types are valid, etc.

The meaning given to a combination of symbols is handled by semantics; the idea is to examine these notions not from a programmer's point of view but from the language designer's point of view.

#### **5. Algorithm Definition**

An algorithm is a set of instructions, sometimes called a procedure or a function that is used to perform a certain task. This can be a simple process, such as adding two numbers together, or a complex function, such as adding effects to an image. Most computer programmers spend a large percentage of their time creating algorithms. (The rest of their time is spent debugging the algorithms that don't work properly.) The goal is to create efficient algorithms that do not waste more computer resources (such as RAM and CPU time) than necessary. This can be difficult, because an algorithm that performs well on one set of data may perform poorly on other data. As you might guess, poorly written algorithms can cause programs to run slowly and even crash. Therefore, software updates are often introduced, touting "improved stability and performance". While this sounds impressive, it also means that the algorithms in the previous versions of the software were not written as well as they could have been.

A main benefit of the use of an algorithm comes from the improvement it makes possible. If the problem solver does not know what was done, he or she will not know what was done wrong. As time goes by and results are compared with goals, the existence of a specified solution process allows identification of weaknesses and errors in the process. Reduction of a task to a specified set of steps or algorithm is an important part of analysis, control, and evaluation.

EXAMPLE:

**Write an algorithm and draw a flow chart to calculate 24.**

**Algorithm power of number.**

**Step 1:** Input Base (2), Power (4)

**Step 2:** Product = Base

**Step 3:** Product = Product \* Base

**Step 4:** Product = Product \* Base

**Step 5:** Product = Product \* Base

**Step 6:** Print Product

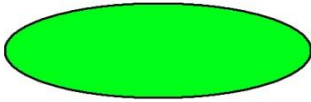


## 6. Flowchart Definition

Flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing. The process of drawing a flowchart for an algorithm is known as “flowcharting”.

### Basic Symbols used in Flowchart Designs

1. **Terminal:** The oval symbol indicates Start, Stop and Halt in a program’s logic flow. A pause/halt is generally used in program logic under some error conditions. Terminal is the first and last symbols in the flowchart.



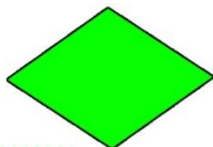
2. **Input/Output:** A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



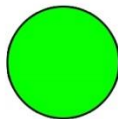
3. **Processing:** A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



4. **Decision** Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.



5. **Connectors:** Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.



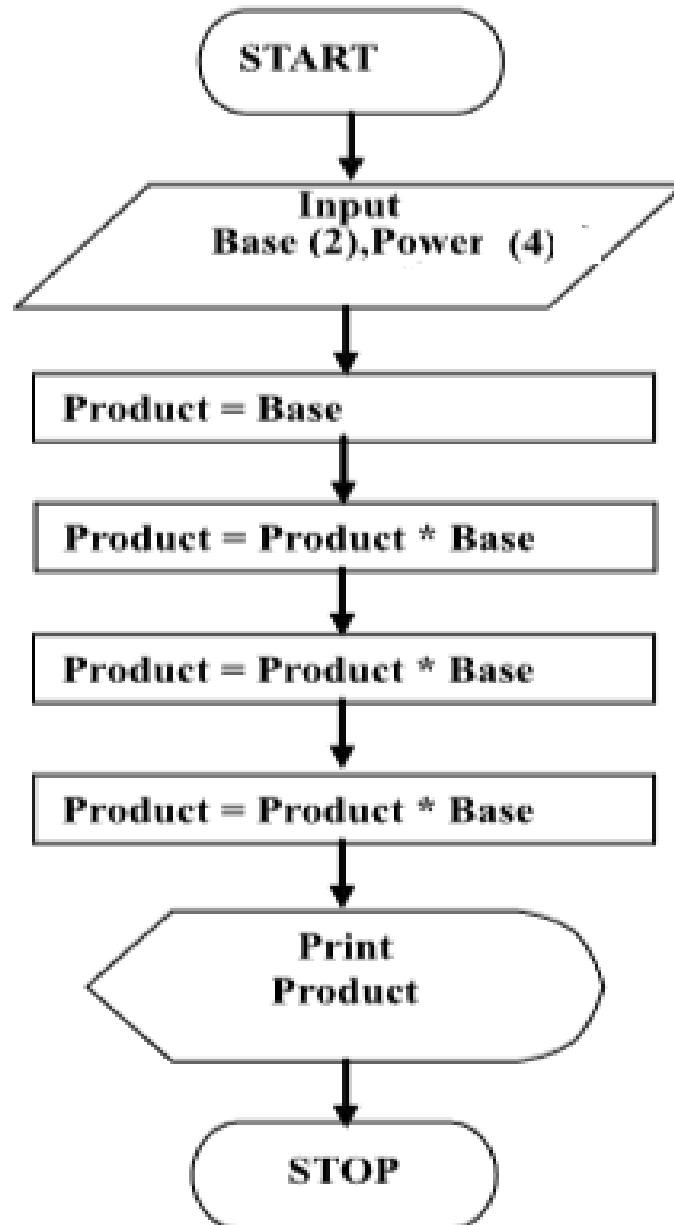
6. **Flow lines:** Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.





**EXAMPLE1:**

**Draw a flow chart to calculate 24.**

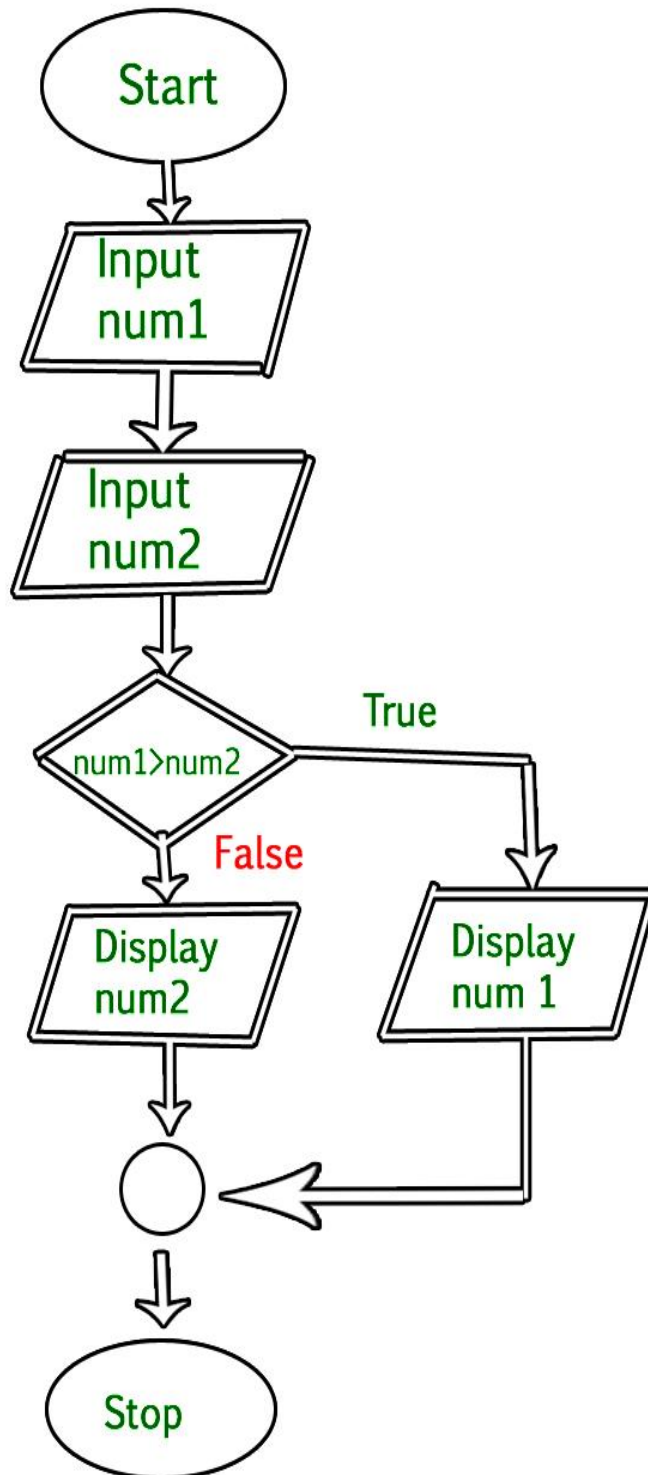






**EXAMPLE2:**

Draw a flowchart to input two numbers from user and display the largest of two numbers.



## **Introduction VB.NET 2008**

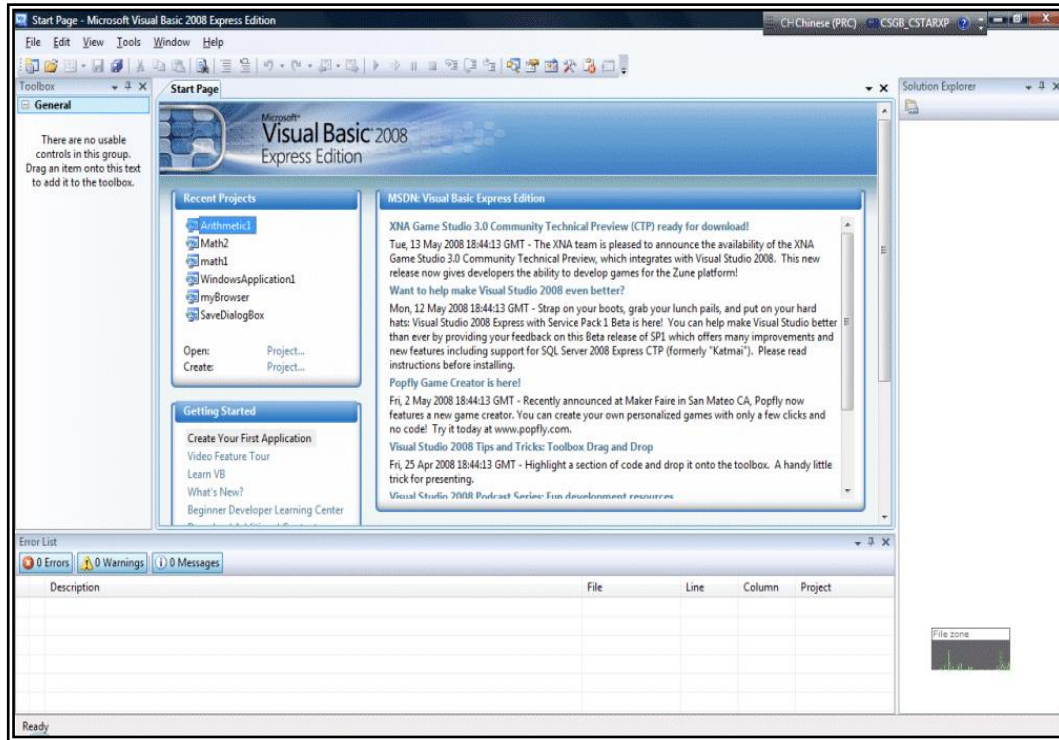
### **1. About Visual Basic**

Visual basic evolved from **BASIC** (**B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode). The BASIC language was created by Professors John Kemeny and Thomas Kurtz of Dartmouth College in the mid1960. It is a carefully constructed English-Like language basically used by the programmers to write simple computer programs. Some of the versions are Microsoft QBASIC, QUICKBASIC, GWBASIC, IBM BASICA, Apple BASIC and etc.

The popularity and widespread use of BASIC with different types of computers brought further enhancements of the language, and eventually led to the GUI-based Visual Basic in tandem with the development of Microsoft Windows. Visual Basic made programming even easier for beginners and season programmers a like as it save considerable programming time by providing many ready-made components. Since then, Visual Basic has also evolved into many versions, until recently, Visual Basic 2008. VB2008 is so far the most powerful version of Visual Basic.

### **1. Introduction Visual Basic 2008**

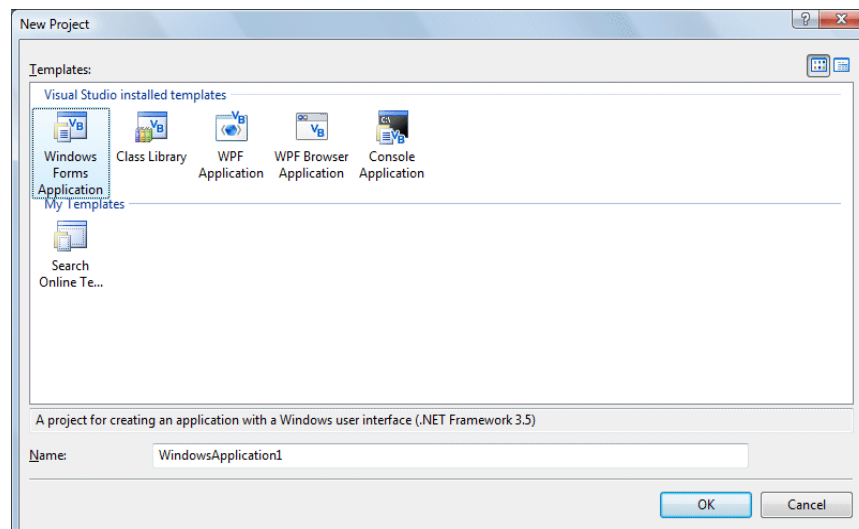
Visual Basic 2008 is the latest version of Visual Basic. It is almost similar to Visual Basic 2005 but it has added many new features. Visual Basic 2008 is a full-fledged Object-Oriented Programming (OOP) Language, so it has caught up with other OOP languages such as C++, Java, C# and others. However, you don't have to know OOP to learn VB2008. In fact, if you are familiar with Visual Basic 6, you can learn VB2008 effortlessly because the syntax and interface are similar. The Integrated Development Environment (IDE) when you launch VB2008 Express is shown in the diagram below.



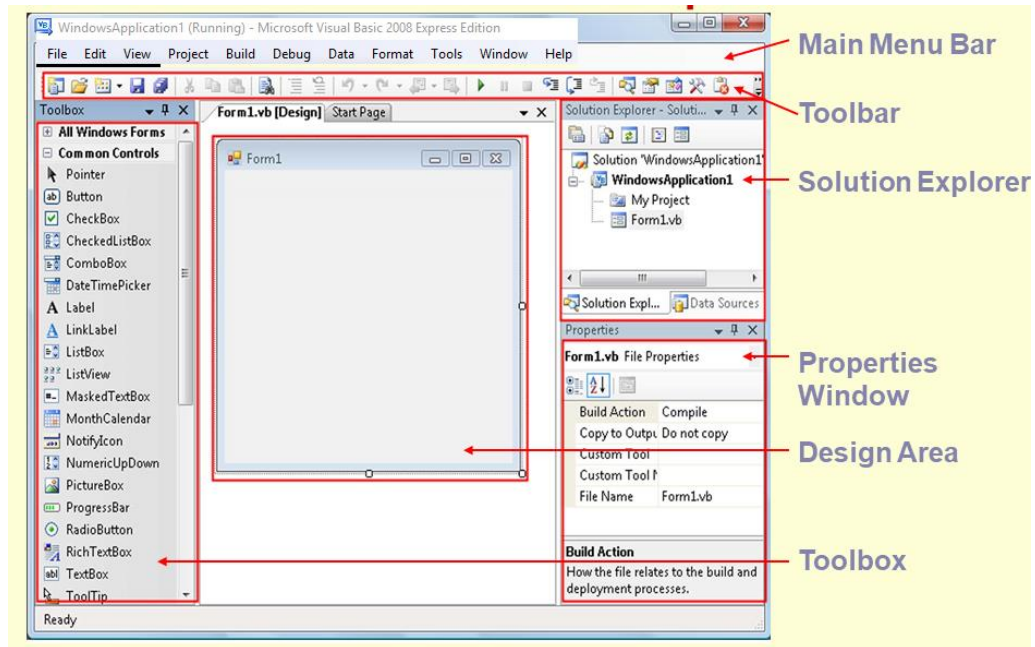
The IDE consists of a few panes, namely:

- The Recent Projects Pane: it shows the list of projects that have been created by you recently.
- The Getting Started Pane: it provides some helpful tips to quickly develop your applications.
- The VB Express Headlines pane: it provides latest online news about Visual Basic 2008 Express. It will announce new releases and updates

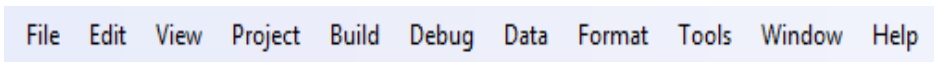
To start creating your first application, you need to click on file and select new project or press create project , the following VB2008 New Project dialog box will appear.



The dialog box offers you five types of projects that you can create. As we are going to learn to create windows Applications, we will select Windows Forms Application. The following IDE Windows will appear, it is almost similar to Visual Basic 6.



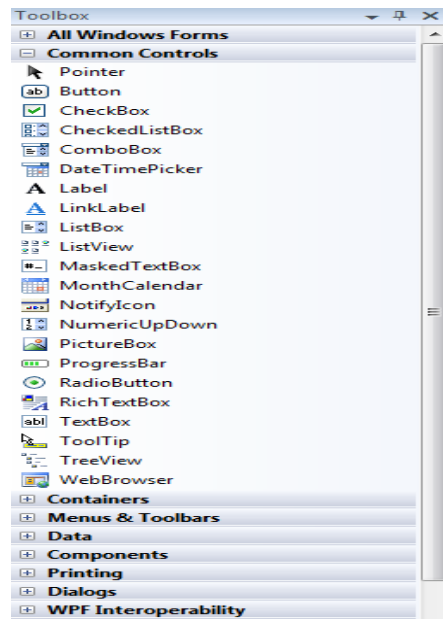
It consists of an empty form, the common controls toolbox, the solution explorer and the properties. The menu bar below:



It consists of all the commands provided by the IDE under the different menus based on their functionality provides access to such functions as opening files, interacting with an application, or help. Standard ToolBar below:

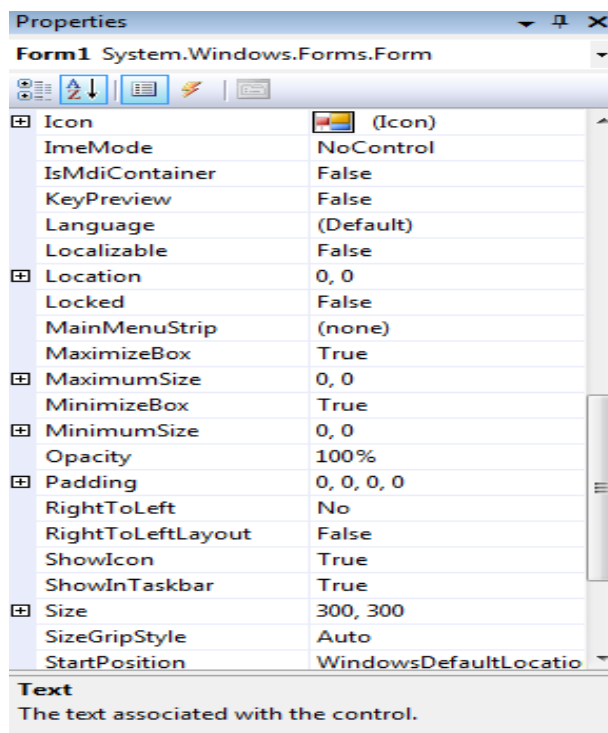


It provides quick access to frequently used commands. The ToolBox below:

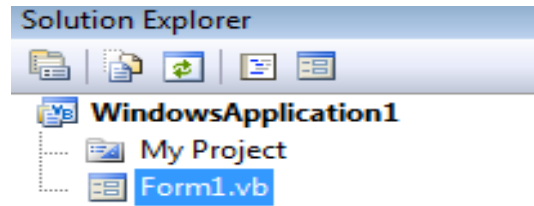


It consists of the various controls that can be used to design the graphical user interfaces of the application.

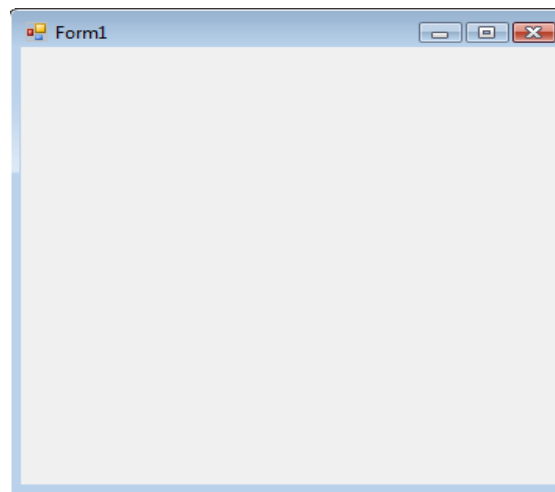
Properties Window describes the properties of the form and its control lists all the properties of the object that's currently selected and gives you the opportunity to modify them.



## Solution Explorer Window (Project Explorer Window)



## Form Designer Window



A design space which contains objects or controls is used for the application. It allows GUI creation by providing the foundation for controls.

### **2. Steps in Creating Simple Applications**

1. Create the User Interface (GUI)
2. Set the Properties of the Objects
3. Enter the appropriate Source Code
4. Run the Program

## Some important Keywords in VB.NET

### 1. Object Properties:

Any tool (also called object or control in vb.net has properties which can be changed ) directly in design window or it can be changed by writing code. To write code for change properties the following rule is applied:

**Control name. Property = property Value**

The properties window displays the properties for a form or control. Properties are attributes such as size, position, etc. like a form; each control type has its own set of properties. Some properties, like width and height, such as, are common to both forms and controls, while other properties are unique to form or control. Controls often differ in the number and type of properties. Properties are listed either alphabetically (by selecting the alphabetic tab) or categorically (by selecting the categorized tab). The most important properties of the objects in general are listed in the following table:

Property	Description
<b>Font</b>	Gets or sets the font of the text displayed by the control.
<b>ForeColor</b>	Gets or sets the foreground color of the control which is the string written on the object to change it from the code use as example: <code>TextBox1.ForeColor = Color.Red</code> The color is selected by designer.
<b>Background</b>	Background color for object. <code>TextBox1.BackColor = Color.Yellow</code>
<b>Text</b>	Allows inputting and editing text in object. <code>TextBox1.Text = " ali "</code>
<b>Visible</b>	The tool is visible or invisible. <code>TextBox1.Visible = True</code>
<b>Enable</b>	The tool enable or disable <code>TextBox1.Enabled = False</code>
<b>Height</b>	Length of object <code>Button1.Height = 150</code>
<b>Width</b>	Width of object <code>Button1.Width = 100</code>
<b>Top</b>	Coordinates of top of object on screen <code>TextBox1.Top = True</code>

<b>Left</b>	Coordinates of left of object on screen <code>TextBox1.Left = True</code>
-------------	---

Some special properties of Textbox control:

Property	Description
<b>Multiline</b>	Gets or sets a value indicating whether this is a multiline TextBox control. <code>TextBox1.Multiline = True</code>
<b>PasswordChar</b>	Gets or sets the character used to mask characters of a password in a single-line TextBox control. <code>TextBox1.PasswordChar = "*" </code>
<b>ReadOnly</b>	Gets or sets a value indicating whether text in the text box is read-only. <code>TextBox1.ReadOnly = True</code>
<b>ScrollBars</b>	Gets or sets which scroll bars should appear in a multiline TextBox control. This property has values : <ul style="list-style-type: none"> <li>• None</li> <li>• Horizontal</li> <li>• Vertical</li> <li>• Both</li> </ul> <code>TextBox1.ScrollBars = ScrollBars.Both</code> <code>TextBox1.ScrollBars = ScrollBars.Horizontal</code> ScrollBars to be viewed must multiline property set to true.
<b>TextAlign</b>	Gets or sets how text is aligned in a TextBox control. This property has values: <ul style="list-style-type: none"> <li>• Left</li> <li>• Right</li> <li>• Center</li> </ul> <code>TextBox1.TextAlign = HorizontalAlignment.Right</code>
<b>MaxLength</b>	Gets the length of text in the control. <code>TextBox1.MaxLength = 300</code>
<b>WordWrap</b>	Indicates whether a multiline text box control automatically wraps words to the beginning of the next line when necessary. <code>TextBox1.WordWrap = False</code>



## Some other Keywords in VB.NET

### 1. Public Class

The Form you've started out with is a Class. If you look right at the top of the code window for a Form, you'll see:

#### Public Class Form1

The word "Public" means that other code can see it. Form1 is the name of the Class  
If you look at the bottom of the coding window, you'll see

#### End Class

, signifying the end of the code for the Class.

### 2. Sub Procedures

A Sub Procedure is a block of code that is executed in response to an event. By breaking the code in a module into Sub procedures, it becomes much easier to find or modify the code in your application. The syntax for a Sub procedure is:

```
Private Sub procedurename (arguments)
Statements
End Sub
```

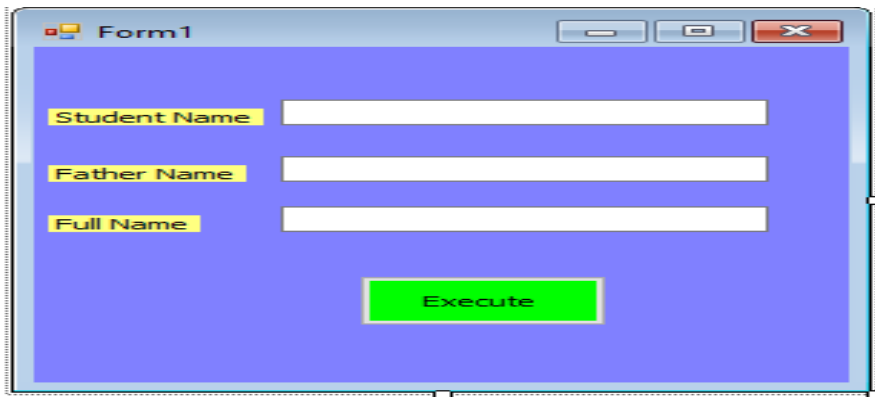
### 3. Events:

Events are like electrical switches. The electrical switches are of many types, so are the events. The form and controls support events (generation, interaction with mouse and keyboard). The most important events in Visual Basic are described in the following table.

<b>Event</b>	<b>Action taken when</b>
Click	Single click on object.
DbClick	Double click on object.
Mouse move	Mouse pointer move object.
Key press	Pressing a key of the key board.
DragDrop	Move object to another place.


**Example:**

Design a form shown in figure below, with three text boxes and one Button. Write code in the button (Execute). So when run project enter the Student Name in TextBox1 and the Father Name in TextBox2 When click on button (Execute) display the Full Name in the TextBox3.



```
Private Sub Button1_Click (-----)
    TextBox3.Text = TextBox1.Text + " " + TextBox2.Text
End Sub
```

**4. Running the Application**

To run the application, choose start from the run menu, or click the start button on the toolbar , or F5.

**5. Comment Statement**

This statement will not compiled or execute it is used only to explain the code. When used comments the code will be changed to green color, there are two forms for comments :

- ✓ REM
- ✓ '

**EXAMPLE**

```
REM comment line in vb.net
' comment line in vb.net
```

## Fundamentals of programming in Visual Basic

### 1. Data Types

Data Types (Constant and Variable): Data types control of internal storage of data in Visual Basic. There are a number of variable data types that allow you to optimize your code for speed and size.

- 1- **Boolean:** A variable of type Boolean requires 2 bytes of memory and holds either the value (True or False).
- 2- **Currency:** The currency data type is extremely useful for calculations involving money. A variable of type Currency requires 8 bytes of memory.
- 3- **Date:** A variable of type Date requires 8 bytes of memory and holds numbers representing dates for example, 5/12/1996.
- 4- **Single:** A variable of type Single requires 4 bytes of memory and can hold 0, the numbers from  $(1.40129 \times 10^{-45}$  to  $3.40283 \times 10^{38})$  with the most seven significant digits, and the negatives of these numbers.
- 5- **Double:** A variable of type Double requires 8 bytes of memory and can hold 0, the numbers from  $(4.94065 \times 10^{-324}$  to  $1.7976 \times 10^{308})$  with at most 14 significant digits and the negatives of these numbers.
- 6- **Integer:** A variable of type integer requires 2 bytes of memory and can hold the whole numbers from  $(-32,768$  to  $32,767)$ .
- 7- **Long:** A variable of type Long requires 4 bytes of memory and can hold the whole numbers.
- 8- **String:** a variable of type string requires 1 byte of memory per character and can hold a string of up to 32,767 characters.
- 9- **Variant:** A variable of type variant can be assigned numbers, Strings and several other types of data. A variable of type variant requires 16 bytes of memory and can hold any type of data.

## 2. Variables

In Visual Basic, uses variable for storage values must start with character and maximum length 255 characters and not contain any point.

### Declaration of a variable

The declaration means defining the variable type. The variable has to be declared with the Dim Statement, supplying a name for the variable:

```
Dim variable name [As type]
```

Variables declared with the Dim statement within a procedure exist only as long as the procedure is executing. When the procedure finishes the value of the variable disappears. In addition, the value of a variable in a procedure is local to that procedure can't access a variable in one procedure from another procedure.

### Conditions about variables name:

- ✓ Must begin with letter.
- ✓ Can't contain an embedded period or embedded type-declaration character.
- ✓ Must not exceed 255 characters. The names of controls, forms, and modules must not exceed 40 characters.
- ✓ They can't be the same as restricted keywords (a restricted keyword is a word that Visual Basic uses as part of its language. This includes predefined statements such as "If and Loop", functions such as "Len and Abs", and operators such as "Or and Mod").

### **Examples:**

```
Dim X As Integer  
Dim Balance As Currency  
Dim Y As Long  
Dim A AS Double, B As Double  
Dim Month As Date  
Dim Max As Single  
Dim Name As String
```

### Error examples:

```
Dim x As string : Dim A, B, C, X (Two Dim statement)
Dim 1st As date (first character is number)
Dim (Ad#1) As string (symbol)
Dim MyName.is As string (point)
Dim Num one As long (space)
```

The types of variables are used the corresponding suffix shown below in the data type table.

Variable Type	Suffix	Example
Boolean, Variant, and Date	None	-----
Integer	%	Dim A%
Long(Integer)	&	Dim Ab&
Single	!	Dim Ac!
Double	#	Dim ACC#
Currency	@	Dim AB1@
String	\$	Dim AA\$

### 3. Constants

Constant also store values, but as the name implies, those values remains constant throughout the execution of an application. Using constants can make your code more readable by providing meaningful names instead of numbers. There are a number of built in constants in Visual Basic. There are two sources for constants:

- ✓ System-defined constants are provided by applications and controls.
- ✓ User-defined constants are declared using the **Const** statement. It is a space in memory filled with fixed value that will not be changed. For example

Constant for procedure	<code>Const X=3.14156</code>
Constant for form and all procedure	<code>Private Const X=3.14156</code>
Constant for all forms	<code>Public Const X=3.14156</code>



## Visual Basic Operators:

The basic operators in VB.NET are listed in table below:

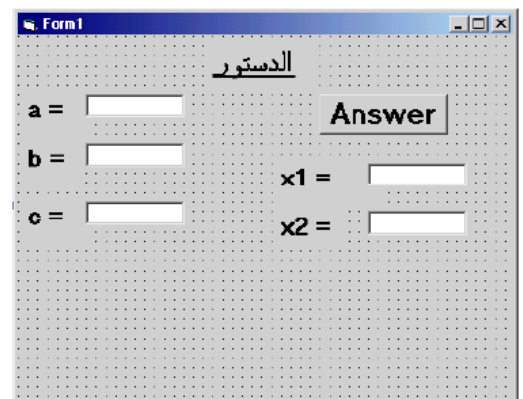
Operation Type	Operation Code	Description
<b>Arithmetic Operation</b>	<b>^</b>	Exponent
	<b>*, /</b>	Multiplication and division
	<b>\</b>	Integer division
	<b>Mod</b>	Modulus rest of division
	<b>- , +</b>	Subtraction and addition
	<b>=</b>	Assignment
<b>String Operation</b>	<b>+</b>	Concatenate two strings
<b>Comparison operators</b>	<b>&gt;</b>	Greater than
	<b>&lt;</b>	Less than
	<b>&gt;=</b>	Greater than or equal to
	<b>&lt;=</b>	Less than or equal to
	<b>=</b>	Equal to
	<b>&lt;&gt;</b>	Not equal to
<b>Logical Operator</b>	<b>Not</b>	Logical not
	<b>And</b>	Logical and
	<b>Or</b>	Logical or

**Examples: Convert the following arithmetic formula to visual Basic language.**

Arithmetic formula	Visual Basic language
$\sqrt[3]{\frac{e^5 + \sin 30}{\log(2) - \tan(35)}}$	<code>((exp(5)+sin(30*3.14159/180))/(log(2)/log(10)-tan(35*3.14159/180))^(1/3))</code>
$\frac{\pi \left(\frac{U_{av}}{100}\right)^2}{4}$	<code>3.14159/4*(Uav/100)^2</code>
$\frac{\pi \left(\frac{U_{av}}{100}\right)^2}{4} \frac{1}{\left[1 - \left(\frac{U_{av}}{100}\right)^{5.63}\right]^{0.533}}$	<code>3.14159/4*(Uav/100)^2/(1-(Uav/100)^5.63)^0.533</code>
$\frac{-b + \sqrt{b^2 - 4 * a * c}}{2 * a}$	<code>(-b+sqr(b^2-4*a*c))/(2*a)</code>

**Home Work:**

Create a Visual Basic project to solve for the roots of the quadratic equation  $aX^2 + bX + c = 0$ , using quadratic formula as:  $X_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ . Design the program so that the values of a, b, and c are entered into separate (labeled) text boxes and display  $X_1$  and  $X_2$  in separate (labeled) text boxes?



**Example: write a vb program to find the area of rectangle, circle and square**

```
Private Sub Button1_Click()  
    Dim L, W, R, D, area As Double  
    L = InputBox("Enter rectangle length")  
    W = InputBox("Enter rectangle width")  
    area = L * W  
    MsgBox("The rectangle area is " & area)  
    R = InputBox("Enter the circle radius")  
    area = R * R * Math.PI  
    MsgBox("The circle area is " & area)  
    D = InputBox("Enter length of squar side")  
    area = D * D  
    MsgBox("The squar area is " & area)  
End Sub
```

**Example: subtract the contents of two textbox.**

```
Private Sub Button1_Click()  
    Dim Z as integer  
    Z=Txtbox1.text - Textbox2.text  
    Listbox1.items.add (Z)  
End sub
```

**Example: find average of three numbers.**

```
Private Sub Button1_Click()  
    Dim i, x,y,z sum, avg As Integer  
    sum = 0  
    x = InputBox("enter numbers")  
    y = InputBox("enter numbers")  
    z = InputBox("enter numbers")  
    sum = x + y + z  
    avg = sum / 3  
    MsgBox(avg)  
End Sub
```



## Input/ Output in Visual Basic

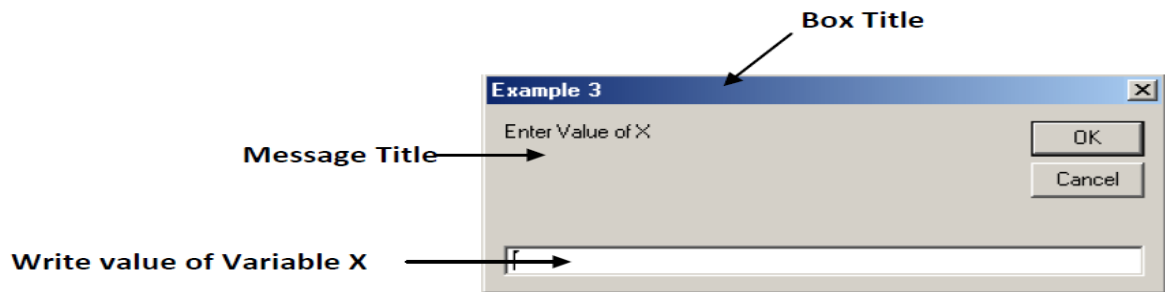
### 1. InputBox Function:

InputBox function is used to input a value or character for one variable from keyboard at running stage.

**Variable-Name =InputBox ("Message" , "Title")**

For Example     X=InputBox( "Enter Value of X " , "Example ")

Note: The type value for InputBox function is string value.



### Example:

```
Private sub button1_click ()  
Dim x, y as integer  
x=inputbox("enter x")  
y=inputbox("enter y")  
textbox1.text = x+y  
End sub
```

## 2. Message Boxes (MsgBox Function):

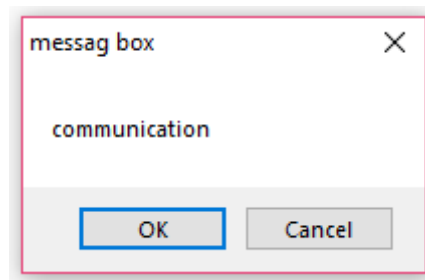
The objective of MsgBox is to produce a pop-up message box and prompt to click on a command button before can continue. This format is as follows:

### **MsgBox ( “Prompt”, Style Value, “Title”)**

The first argument, Prompt, will display the message in the message box. The Style Value will determine what type of command buttons appear on the message box. The Title argument will display the title of the message board. The Style values are listed below.

- `MsgBoxStyle.OkCancel`
- `MsgBoxStyle.OkOnly`
- `MsgBoxStyle.RetryCancel`
- `MsgBoxStyle.YesNo`
- `MsgBoxStyle.YesNoCancel`

Example:



### **Example:**

```
Private sub button1_click ()  
Dim x, y as integer  
x=inputbox("enter x")  
y=inputbox("enter y")  
msgbox(" The result of sum", " x + y ")  
End sub
```

## Control Structures

In this lesson, you will learn how to write VB.NET code that can make decision when it process input from the users, and control the program flow in the process. Decision making process is an important part of programming because it will help solve practical problems intelligently so that it can provide useful output or feedback to the user. For example, we can write a program that can ask the computer to perform certain task until a certain condition is met, or a program that will reject non-numeric data. In order to control the program flow and to make decisions, we need to use the **conditional operators** and the **logical operators** together with the If control structure.

There are basically three types of control structures, namely:

- **If - Then**
  - **If - Then - Else**
  - **If - Then - ElseIf - Then**
  - **Select Case**
- **If - Then Statement:**

This is the simplest control structure which asks the computer to perform a certain action specified by the VB expression if the condition is true. However, when the condition is false, no action will be performed. The general format for the (If- Then) statement is:

```
If condition Then  
VB Expression  
End If
```

**Example:** Write a program to enter the value of two variables (X and Y). Find and print the Maximum value for two variables.

```
Private Sub Button1_Click()  
    Dim x, y, max As Integer  
    x = InputBox("x")  
    y = InputBox("y")  
    If x > y Then  
        max = x  
    End If  
    If y > x Then  
        max = y  
    End If  
    MsgBox(max)  
End Sub
```

- **The If – Then - Else statement**

Allows the programmer to specify that a different action is to be performed when a certain action specified by the VB expression if the condition is True then when the condition is false, an alternative action will be executed. The general format for the **If - Then - Else** statement is:

```

If condition Then
  VB expression
Else
  VB expression
End If
  
```

Same pervious example:

```

Private Sub Button1_Click()
  Dim x, y, max As Integer
  x = InputBox("x")
  y = InputBox("y")
  If x > y Then
    max = x
  Else
    max = y
  End If
  MsgBox(max)
End Sub
  
```

- **Nested (If – Then – Else) statement**

If there are more than two alternative choices, using just If – Then - Else statement will not be enough. In order to provide more choices, we can use If...Then...Else statement inside If...Then...Else structures. The general format for the Nested If...Then.. Else statement is

<u>Method 1</u>	<u>Method 2</u>
<pre> 1 → If <i>condition 1</i> Then    <i>VB expression</i>    Else    2 → If <i>condition 2</i> Then     <i>VB expression</i>     Else     3 → If <i>condition 3</i> Then      <i>VB expression</i>      Else      <i>VB expression</i>      End If     End If    End If   End If   </pre>	<pre> If <i>condition 1</i> Then   <i>VB expression</i> ElseIf <i>condition 2</i> Then   <i>VB expression</i> ElseIf <i>condition 3</i> Then   <i>VB expression</i> Else   <i>VB expression</i> End If   </pre>

**Example:** Write a program to enter the value of variable (Mark). Find the grade using If – Block statement and display the value of grade in a text box. When the value of variable (Mark) exceed 100 or less than 0, write a Message Box (Wrong entry, please Re-enter the Mark). Design form window and select all the control objects are used.

```
Private Sub Button1_Click()  
    Dim Mark As Single  
    Dim Grade As String  
    Mark = InputBox("enter the your degree ")  
    If (Mark > 100) Or (Mark < 0) Then  
        MsgBox("Wrong entry, please Re-enter the mark")  
    ElseIf (Mark >= 90) And (Mark <= 100) Then  
        Grade = "Excellent"  
    ElseIf (Mark >= 80) Then  
        Grade = "Very Good"  
    ElseIf Mark >= 70 Then  
        Grade = "Good"  
    ElseIf Mark >= 60 Then  
        Grade = "Medium"  
    ElseIf Mark >= 50 Then  
        Grade = "Pass"  
    Else  
        Grade = "Fail"  
    End If  
    TextBox1.Text = Grade  
End Sub
```

**Example: write a vb program to find the max of two numbers.**

```
Private Sub Button1_Click()  
Dim i, max, x, f As Integer  
x = InputBox("enter number")  
y = InputBox("enter number")  
If x > y Then  
  
ListBox1.Items.Add(x)  
Else  
ListBox1.Items.Add(y)  
End If
```

**Example: subtract the contents of two textbox.**

```
Private Sub Button1_Click()  
Dim Z as integer  
Z=Textbox1.text - Textbox2.text  
Listbox1.items.add (Z)  
End sub
```

**Example: find average of three numbers.**

```
Private Sub Button1_Click()  
Dim i, x,y,z sum, avg As Integer  
sum = 0  
x = InputBox("enter numbers")  
y = InputBox("enter numbers")  
z = InputBox("enter numbers")  
sum = x + y + z  
avg = sum / 3  
MsgBox(avg)  
End Sub
```

## Select Case

Select - Case structure is an alternative to If – Then - ElseIf for selectively executing a single block of statements from among multiple block of statements. The Select Case control structure is slightly different from the If - ElseIf control structure.

The format of the Select Case control structure is as follows:

<b>Select Case</b> <i>test expression</i>	<b>Select Case</b> <i>test expression</i>
<b>Case</b> expression list 1 VB statements	<b>Case IS</b> expression VB statements
<b>Case</b> expression list 2 VB Statements	<b>Case IS</b> expression VB Statements
<b>Case</b> expression list 3 VB statements	<b>Case IS</b> expression VB statements
<b>Case</b> expression list 4 .....	<b>Case IS</b> expression .....
<b>Case Else</b> VB Statements	<b>Case Else</b> VB Statements
<b>End Select</b>	<b>End Select</b>



**Example:** write vb program to perform mathematical operations between variables(x,y) as operator entered by user.(use select case)

```
Private Sub Button1_click ()
    Dim x, y As Integer
    Dim op As String
    x = InputBox("x")
    y = InputBox("y")
    op = InputBox("op")
    Select Case op
        Case "+"
            ListBox1.Items.Add("x + y= " & x + y)
        Case "-"
            ListBox1.Items.Add("x - y= " & x - y)
        Case "/"
            ListBox1.Items.Add("x / y= " & x / y)
        Case "*"
            ListBox1.Items.Add("x * y= " & x * y)
        Case "^"
            ListBox1.Items.Add("x ^ y= " & x ^ y)
        Case Else
            ListBox1.Items.Add(" not defined ")
    End Select
End Sub
```





**Example** : write vb program to find if input is positive or negative or zero.

```
Private Sub Button1_click ()
    Dim x As Integer
    x = InputBox("x")
    Select Case x
        Case Is > 0
            MsgBox("+ve")
        Case Is < 0
            MsgBox("-ve")
        Case Else
            MsgBox("zero")
    End Select
End Sub
```



## Select Case Examples

**Example:** using select case to classify each input to symbol,character,number and operators ,other inputs will be undifined.

```
Private Sub Button1_click()  
    Dim x As String  
    x = InputBox("x")  
    Select Case x  
        Case "A" To "Z"  
            MsgBox(" C letter")  
        Case "a" To "z"  
            MsgBox(" S letter")  
        Case "/", ">", "<", ".", "!", "?", "@", "#"  
            MsgBox(" Symbol")  
        Case "/", "*", "-", "+", "^"  
            MsgBox(" operators ")  
    End Select  
End Sub
```

**Example:** write program to enter number and distinguish if it is even or odd.  
(use select case)

```
Private Sub Button1_click ()  
    Dim x As Integer  
    x = InputBox("number")  
    Select Case (x Mod 2)  
        Case 0  
            MsgBox(" even ")  
        Case 1  
            MsgBox("odd ")  
    End Select
```

**Example:** write a vb program to enter student degree and evaluate it as excellent, very good, good, medium, pass and fail.

```
Dim Mark As Integer
Dim Grade As String
Mark = Val (Textbox1.Text)
Select Case Mark
    Case Is >100, Is < 0
        MsgBox "Wrong entry, please Re-enter the mark"
        Exit Sub
    Case Is > = 90
        Grade="Excellent"
    Case Is > = 80
        Grade="Very Good"
    Case Is >= 70
        Grade="Good"
    Case Is >= 60
        Grade="Medium"
    Case Is >=50
        Grade="Pass"
    Case Else
        Grade="Fail"
End Select
Textbox2.Text=Grade
End Sub
```

**Example :** write vb program to find if the input number is in range less than twenty for range(1-20) and more thirty in range (30-50)

```
Private Sub Button1_Click()  
    Dim x As Integer  
    x = InputBox("x")  
    Select Case x  
        Case 1, 2, 3 To 20  
            MsgBox("x less 20")  
        Case 31, 32 To 50  
            MsgBox("x more than 30")  
    End Select  
End Sub
```

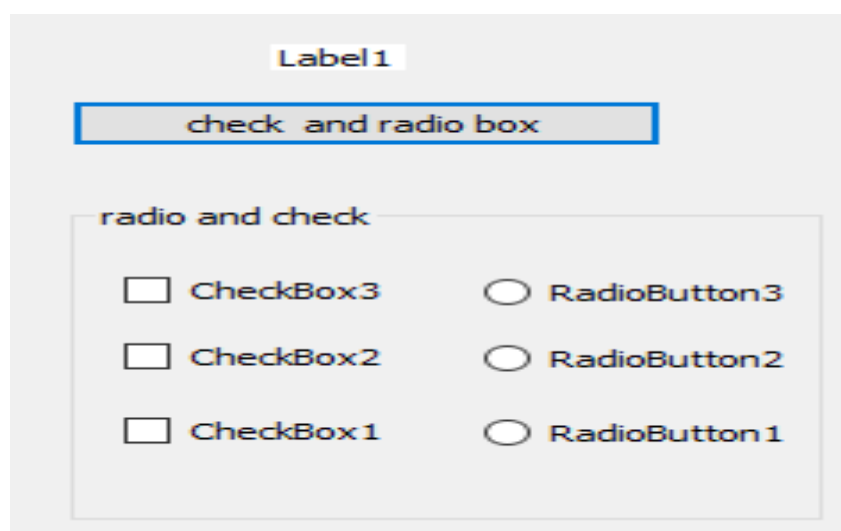
**Example:** use select case in vb.net to write program to classify the letters into small and capital letters ,symbols and numbers for the input variable.

```
Private Sub Button1_Click()  
    Dim x As String  
    x = InputBox("x")  
    Select Case x  
        Case "A" To "Z"  
            MsgBox(" C letter")  
        Case "a" To "z"  
            MsgBox(" S letter")  
        Case "/", ">", "<", ".", "!", "?", "@", "#"  
            MsgBox(" Symbol")  
        Case "/", "*", "-", "+", "^"  
            MsgBox(" operators ")  
        Case 0 To 9  
            MsgBox(" number")  
        Case Else  
            MsgBox(" undefined ")  
    End Select  
End Sub
```

### Option Controls ( radiobuttons and CheckBox ):

Option controls are ( radiobuttons and CheckBox ) known as option tools because of their shape. Radiobuttons always use these controls in a group of two or more because their purpose is to offer a number of mutually exclusive choices. Anytime you click on a button in the group, it switches to a selected state and all the other controls in the group become unselected. Preliminary operations for an radiobuttons control are similar to those already described for CheckBox controls. You set an option control's text property to a meaningful string. If the control is the one in its group that's in the selected state, you also set its (checked) property to True. (The checked property is a Boolean value because only two states are possible). The event used with option controls is (CheckedChanged)

**Example:** Use checkbox and Rdiobutton tools to change the background color of button1 and label1.



## Example code:

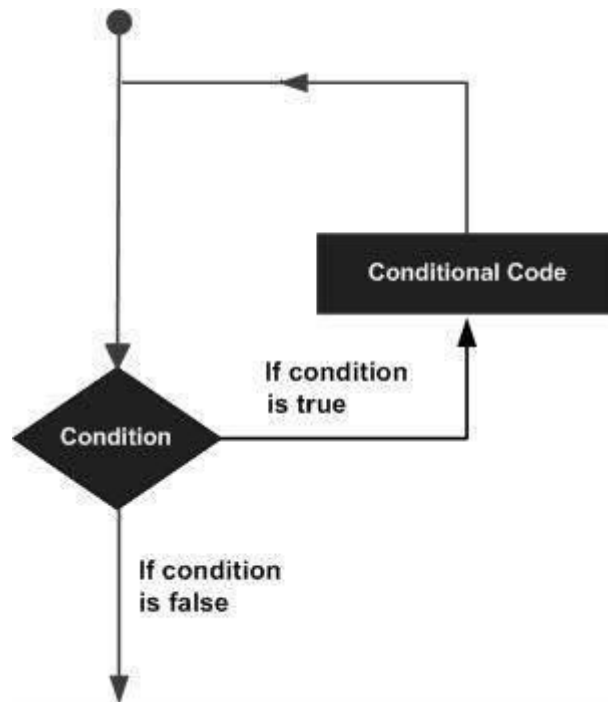
```
Private Sub CheckBox1_CheckedChanged()  
    If CheckBox1.Checked = True Then  
        Label1.BackColor = Color.SeaGreen  
    Else  
        Label1.BackColor = Color.Coral  
    End If  
End Sub  
-----  
Private Sub CheckBox2_CheckedChanged()  
    If CheckBox2.Checked = True Then  
        Label1.BackColor = Color.Cyan  
    Else  
        Label1.BackColor = Color.Coral  
    End If  
End Sub  
-----  
Private Sub CheckBox3_CheckedChanged()  
    If CheckBox3.Checked = True Then  
        Label1.BackColor = Color.Black  
    Else  
        Label1.BackColor = Color.Coral  
    End If  
End Sub  
-----  
Private Sub RadioButton3_CheckedChanged()  
    If RadioButton3.Checked = True Then  
        Button1.BackColor = Color.Green  
    Else  
        Button1.BackColor = Color.White  
    End If  
End Sub  
-----  
Private Sub RadioButton2_CheckedChanged()  
    If RadioButton2.Checked = True Then  
        Button1.BackColor = Color.Wheat  
    Else  
        Button1.BackColor = Color.Coral  
    End If  
End Sub  
-----  
Private Sub RadioButton1_CheckedChanged()  
    If RadioButton2.Checked = True Then  
        Button1.BackColor = Color.Tomato  
    Else  
        Button1.BackColor = Color.Coral  
    End If  
End Sub
```

## Loops (Repetition) Structures

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages.



VB.Net provides following types of loops to handle looping requirements.

- **For-Next**
- **Do –loop While**
- **Do –loop until**
- **While –End While**
- **Nested loops**



For- Next	
<b>Description</b>	It repeats a group of statements a specified number of times and a loop index counts the number of loop iterations as the loop executes.
<b>syntax</b>	<b>For</b> counter = Start <b>To</b> End <b>Step</b> [Increment] Statements <b>Next</b> [counter]
<b>Example</b>	<pre>Dim i As Integer     For i = 1 To 10 Step 2         ListBox1.Items.Add(i)     Next i</pre>
Do -While	
<b>Description</b>	It repeats the enclosed block of statements while a Boolean condition is True. It could be terminated at any time with the Exit Do statement.
<b>syntax</b>	<b>Do</b> Statement <b>Loop While</b> (condition)
<b>Example</b>	<pre>Dim i As Integer = 1     Do         ListBox1.Items.Add(i)         i = i + 1     Loop While (i &lt;= 10)</pre>
While... End While	
<b>Description</b>	It executes a series of statements as long as a given condition is True.
<b>syntax</b>	<b>While</b> Statement <b>End While</b> (condition)
<b>Example</b>	<pre>Dim i As Integer = 1     While (i &lt;= 10)         ListBox1.Items.Add(i)         i = i + 1     End While</pre>
Nested loops	
<b>Example</b>	<p>You can use one or more loops inside any another loop.</p> <pre>Dim i As Integer For i = 1 To 5     For j = 1 To 3         ListBox1.Items.Add(i)         ListBox2.Items.Add(j)     Next j Next i</pre>



## Examples

**Write a vb program to display only odd numbers in the range (1-10) and display the result in listbox and combobox.**

```
Private Sub Button1_Click()
Dim i As Integer
For i = 1 To 10 Step 2
ListBox1.Items.Add(i)
ComboBox1.Items.Add(i)
Next i
End Sub
```

\*\*\*\*\*  
**Example for using (step) in (For Loop)**

a) Private Sub Button1\_Click()

```
Dim i As Integer
For i = 10 To -5 Step -5
ListBox1.Items.Add(i)
Next i
MsgBox(i)
End Sub
```

b) Private Sub Button1\_Click()

```
Dim i As Integer
For i = 1 To 10 Step 10
ListBox1.Items.Add(i)
Next i
MsgBox(i)
End Sub
```

\*\*\*\*\*  
**Use while loop to display numbers in range (1-10) in listbox**

```
Private Sub Button1_Click()
Dim i As Integer
i = 1
While (i <= 10)
ListBox1.Items.Add(i)
i = i + 1
End While
End Sub
```

\*\*\*\*\*

## Examples

### Example about nested loops

```
Private Sub Button1_Click()  
Dim i, j, k As Integer  
For i = 1 To 5  
For j = 1 To 3  
For k = 1 To 4  
ListBox1.Items.Add(i)  
ListBox2.Items.Add(j)  
ListBox3.Items.Add(k)  
Next k  
Next j  
Next i  
End Sub
```

\*\*\*\*\*

### Use Do-While loop to display numbers in range(1-10)

```
Private Sub Button1_Click()  
Dim i As Integer  
i = 1  
Do  
ListBox1.Items.Add(i)  
i = i + 1  
Loop While (i <= 10)  
End Sub
```

\*\*\*\*\*

## Find factorial of any input number

```
Private Sub Button1_Click()  
Dim i, x, f As Integer  
x = InputBox("enter your number")  
f = 1  
For i = 1 To x  
f = f * i  
ListBox1.Items.Add(f)  
Next  
End Sub
```

## Examples

**write vb program that enter ten numbers and find the sum of odd number and the sum of even number**

```
Private Sub Button12_Click()  
Dim i, x, sumE, sumO As Integer  
'initialize sum variable with value  
'zero which not effect the summation result  
sumE = 0  
sumO = 0  
For i = 1 To 10  
x = InputBox("enter numbers")  
If (x Mod 2 = 0) Then 'check for even numbers  
sumE = sumE + x 'find sum of no.  
ListBox1.Items.Add(x) 'display the even no.  
Else 'find sum of odd no.  
sumO = sumO + x  
ListBox2.Items.Add(x)  
End If  
Next  
ListBox1.Items.Add("sum Even")  
ListBox1.Items.Add(sumE)  
ListBox2.Items.Add("sum Odd")  
ListBox2.Items.Add(sumO)  
End Sub
```

\*\*\*\*\*

## Find the factorial of maximum number for ten numbers

```
Private Sub Button1_Click()  
Dim i, max, x, f As Integer  
max = 0  
f = 1  
For i = 1 To 10  
x = InputBox("enter number")  
ListBox1.Items.Add(x)  
If x > max Then  
max = x  
End If  
Next  
ListBox1.Items.Add("max is")  
ListBox1.Items.Add(max)  
f = 1  
For i = 1 To max  
f = f * i  
Next  
ListBox1.Items.Add("factorial max is" & f )  
End Sub
```

## Examples

### Find the max number between (5) numbers

```
Private Sub Button11_Click()  
Dim i, max, x As Integer  
max = 0  
For i = 1 To 5  
x = InputBox("enter number")  
If x > max Then  
max = x  
End If  
Next  
ListBox1.Items.Add(max)  
End Sub
```

### write vb program to display the multiplication number for the given number.

```
Private Sub Button1_Click()  
Dim i, x, mult As Integer  
x = InputBox("enter number")  
For i = 0 To 10  
mult = i * x  
ListBox1.Items.Add(mult)  
Next  
End Sub
```

**write vb program to find the average of (8) numbers**

```
Private Sub Button1_Click()  
Dim i, x, sum, avg As Integer  
sum = 0  
For i = 1 To 10  
x = InputBox("enter numbers")  
sum = sum + x  
Next  
avg = sum / 10  
MsgBox (avg)  
End Sub
```



## Examples

### Find the sum of odd negative of ten numbers

```
Private Sub Button1_Click()  
    Dim x, i, sumon, sumep As Integer  
    sumon = 0  
    For i = 1 To 10  
        x = InputBox("")  
        ListBox2.Items.Add(x)  
        If (x Mod 2 <> 0) And (x < 0) Then  
            sumon = sumon + x  
        End If  
    Next  
    ListBox1.Items.Add("sumon " & sumon)  
  
End Sub
```





## Find the sum of even positive of ten numbers

```
Private Sub Button18_Click()  
    Dim x, i, sumep As Integer  
    sumep = 0  
    For i = 1 To 10  
        x = InputBox("")  
        ListBox2.Items.Add(x)  
        If (x Mod 2 = 0) And (x > 0) Then  
            sumep = sumep + x  
        End If  
    Next  
    ListBox1.Items.Add("sumep " & sumep)  
End Sub
```

## Examples

### Find the sum of odd negative of ten numbers

```
Private Sub Button1_Click()  
    Dim x, i, sumon, sumep As Integer  
    sumon = 0  
    For i = 1 To 10  
        x = InputBox("")  
        ListBox2.Items.Add(x)  
        If (x Mod 2 <> 0) And (x < 0) Then  
            sumon = sumon + x  
        End If  
    Next  
    ListBox1.Items.Add("sumon    " & sumon)  
  
End Sub
```

## Find the sum of even positive of ten numbers

```
Private Sub Button18_Click()  
    Dim x, i, sumep As Integer  
    sumep = 0  
    For i = 1 To 10  
        x = InputBox("")  
        ListBox2.Items.Add(x)  
        If (x Mod 2 = 0) And (x > 0) Then  
            sumep = sumep + x  
        End If  
    Next  
    ListBox1.Items.Add("sumep " & sumep)  
End Sub
```



## Examples

**Find the sum of odd negative and sum of even positive of ten numbers and then find maximum sum between them.**

```
Private Sub Button1_Click()  
    Dim x, i, sumon, sumep As Integer  
    sumep = 0  
    sumon = 0  
    For i = 1 To 10  
        x = InputBox("")  
        ListBox2.Items.Add(x)  
        If (x Mod 2 <> 0) And (x < 0) Then  
            sumon = sumon + x  
        ElseIf (x Mod 2 = 0) And (x > 0) Then  
            sumep = sumep + x  
        End If  
    Next  
    ListBox1.Items.Add("sumon    " & sumon)  
    ListBox1.Items.Add("sumep    " & sumep)  
    If sumep > sumon Then  
        ListBox1.Items.Add("max    " & sumep)  
    Else  
        ListBox1.Items.Add("max    " & sumon)  
    End If  
End Sub
```

**Write a vb program to enter the names of (5) studentes and for each student entering (8) degrees , the name of student and his degrees will displayed in listbox.**

```
Private Sub Button1_Click()  
  
    Dim i, j, d As Integer  
    Dim n As String  
    ListBox1.Items.Clear()  
    For i = 1 To 5  
        n = InputBox(" name")  
        ListBox1.Items.Add(n)  
        ListBox2.Items.Add(i)  
  
        For j = 1 To 8  
            d = InputBox("degree")  
            ListBox3.Items.Add(d)  
            ListBox4.Items.Add(j)  
  
        Next j  
    Next i  
End Sub
```

## Built in functions

The built in functions are classified in many library according to their operations such as math and strings library .the following functions are math library function.

### 1. Fix function is used to truncate the fraction part from the number

```
x = 89.78  
TextBox1.Text = Fix(x)           o/p      89
```

### 2. PI function give the constant ratio (3.14)

```
TextBox1.Text = Math.PI           o/p      3.14159265358979
```

### 3. abs function give the absolute value of number

```
MsgBox (Math.Abs (-567.8))           o/p      567.8
```

### 4. pow function is used to power operation in mathmatics.

```
MsgBox (Math.Pow (2, 4))           o/p      16
```

```
MsgBox (Math.Pow (5, 0))           o/p      1
```

### 5. sqrt function is used to find the squar root of a number.

```
MsgBox (Math.Sqrt (16))           o/p      4
```

**6. sign function gives (1) for positive number , (-1) for negative number and (0) for zero number**

<code>msgBox (Math.Sign (-916))</code>	o/p	-1
<code>MsgBox (Math.Sign (16))</code>	o/p	1
<code>MsgBox (Math.Sign (0))</code>	o/p	0

**6. Exp function is used to calculate the exponential of the number.**

<code>MsgBox (Math.Exp (0))</code>	o/p	1
<code>TextBox1.Text = Math.Exp (20)</code>	o/p	485165195.40979
<code>TextBox2.Text = Math.Exp (5)</code>	o/p	148.413159102577





## Other Built in Functions

In mathematics, the trigonometric functions are functions of an angle. They relate the angles of a triangle to the lengths of its sides. Trigonometric functions are important in the study of triangles and modeling periodic phenomena, among many other applications. These functions are sin ,cos ,tan, sinh ....

The vb.net functions as below:

- `Sin(x) MsgBox(Math.Sin(5))`  
o/p `-0.958924274663138`
- `Cos(x) MsgBox(Math.Sin(5))`  
o/p `0.283662185463226`
- `Tan(x) MsgBox(Math.tan(5))`  
o/p `-3.3805150062465`

the inverse of the functions is also available by the functions (`Asin(x)`, `Acos(x)`, `Atan(x)`)

String library contains many functions some of them are:

- **Ucase function used to convert input string to upper case character**

```
MsgBox(Strings.UCase("computer"))
```

o/p `COMPUTER`

- **Lcase function used to convert input string to lower case character**

```
MsgBox(Strings.LCase("COMPUTER"))
```

o/p `computer`

- **Len function is used to compute the number of characters in string (length of string).**

```
MsgBox(Strings.Len("computer"))
```

o/p `8`