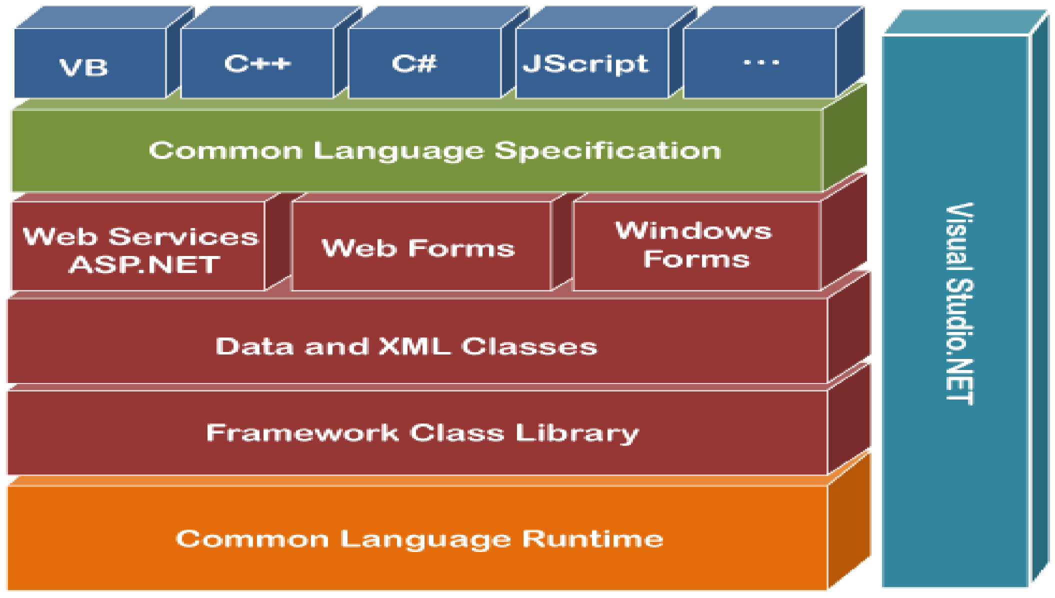
Unit I

INTRODUCTION TO .NET

.NET Framework

- It is a virtual machine that provide a common platform to run an application that was built using the different language such as C#, VB.NET, Visual Basic, etc.
- It is also used to create a form based, console-based, mobile and web-based application or services that are available in Microsoft environment.
- Furthermore, the .NET framework is a pure object oriented, that similar to the Java language.
- But it is not a platform independent as the Java. So, its application runs only to the windows platform.
- The main objective of this framework is to develop an application that can run on the windows platform. The current version of the .Net framework is 4.8.

4.0. NET FRAMEWORK FEATURES & ARCHITECTURE



Components of .NET Framework

There are following components of .NET Framework:

1. CLR (Common Language Runtime)
2. CTS (Common Type System)
3. BCL (Base Class Library)
4. CLS (Common Language Specification)
5. FCL (Framework Class Library)
6. .NET Assemblies
7. XML Web Services
8. Window Services

CLR (common language runtime)

- It is an important part of a .NET framework that works like a virtual component of the .NET Framework to executes the different languages program like c#, Visual Basic, etc.
- A CLR also helps to convert a source code into the byte code, and this byte code is known as CIL (Common Intermediate Language) or MSIL (Microsoft Intermediate Language).
- After converting into a byte code, a CLR uses a JIT compiler at run time that helps to convert a CIL or MSIL code into the machine or native code.

CTS (Common Type System)

- It specifies a standard that represent what type of data and value can be defined and managed in computer memory at runtime.
- A CTS ensures that programming data defined in various languages should beinteract with each other to share information.
- For example, in C# we define data type as int, while in VB.NET we define integer as a data type.

BCL (Base Class Library)

- The base class library has a rich collection of libraries features and functions that help to implement many programming languages in the .NET Framework, such as C #, F #, Visual C ++, and more. Furthermore, BCL divides into two parts:

1. **User defined class library**
    - **Assemblies -** It is the collection of small parts of deployment an application's part. It contains either the DLL (Dynamic Link Library) or exe (Executable) file.
        1. In DLL, it uses code reusability, whereas in exe it contains only output file/ or application.
        2. DLL file can't be open, whereas exe file can be open.
        3. DLL file can't be run individually, whereas in exe, it can run individually.
        4. In DLL file, there is no main method, whereas exe file has main method.
2. **Predefined class library**
    - **Namespace -** It is the collection of predefined class and method that present in .Net. In other languages such as, C we used header files, in java we used package similarly we used "using system" in .NET, where using is a keyword and system is a namespace.

CLS (Common language Specification)

- It is a subset of common type system (CTS) that defines a set of rules and regulations which should be followed by every language that comes under the .net framework.
- In other words, a CLS language should be cross-language integration or interoperability.
- For example, in C# and VB.NET language, the C# language terminate each statement with semicolon, whereas in VB.NET it is not end with semicolon, and

when these statements execute in .NET Framework, it provides a common platform to interact and share information with each other.

Microsoft .NET Assemblies

- A .NET assembly is the main building block of the .NET Framework. It is a small unit of code that contains a logical compiled code in the Common Language infrastructure (CLI), which is used for deployment, security and versioning.
- It defines in two parts (process) DLL and library (exe) assemblies.
- When the .NET program is compiled, it generates a metadata with Microsoft Intermediate Language, which is stored in a file called Assembly.

FCL (Framework Class Library)

- It provides the various system functionality in the .NET Framework that includes classes, interfaces and data types, etc. to create multiple functions and different types of application such as desktop, web, mobile application, etc.
- In other words, it can be defined as, it provides a base on which various applications, controls and components are built in .NET Framework.

Key Components of FCL

1. Object type
2. Implementation of data structure
3. Base data types
4. Garbage collection
5. Security and database connectivity
6. Creating common platform for window and web-based application

Characteristics of .NET Framework

1. CLR (Common Language Runtime)
2. Namespace - Predefined class and function

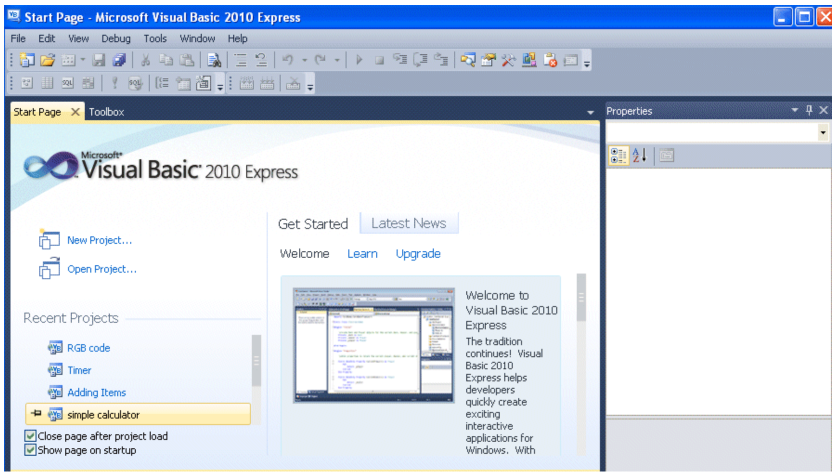
3. Metadata and Assemblies
4. Application domains
5. It helps to configure and deploy the .net application
6. It provides form and web-based services
7. NET and ASP.NET AJAX,
8. LINQ
9. Security and Portability
10. Interoperability
11. It provides multiple environments for developing an application

INTRODUCTION TO VISUAL STUDIO 2010

1.1 The Visual Basic 2010 Integrated Development Environment

- The IDE Start Page consists of a few sections, namely:
    - The New Project/Open Project section.

      The Recent Projects section that shows a 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 Latest News section- It provides latest online news about Visual Basic 2010 Express. It will announce new releases and updates
- The Properties section

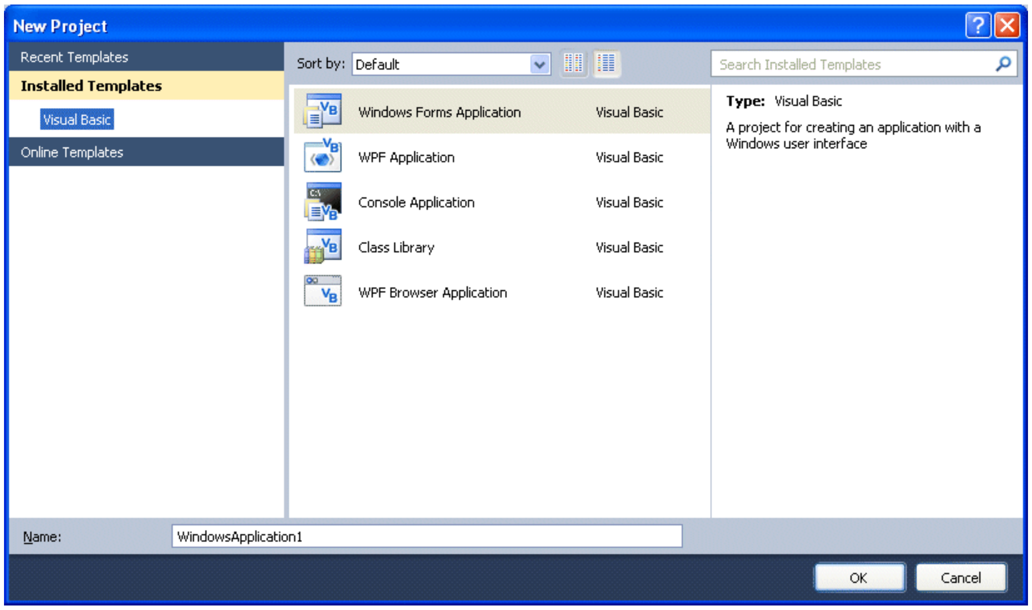


Welcome to Visual Basic 2010 Express

The tradition continues! Visual Basic 2010 Express helps developers quickly create exciting interactive applications for Windows. With

1.2 Creating Your First Application

- To start creating your first application, you need to click on New Project.
- The following VB2010 New Project dialog box will appear.



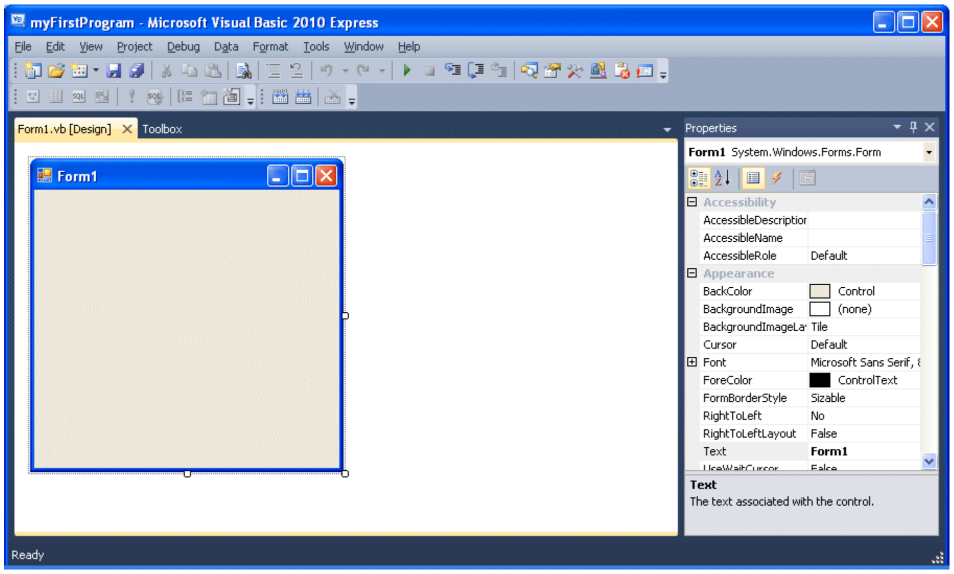
Windows Forms Application	Visual Basic
WPF Application	Visual Basic
Console Application	Visual Basic
Class Library	Visual Basic
WPF Browser Application	Visual Basic

Type: Visual Basic

A project for creating an application with a Windows user interface

Name: WindowsApplication1

- 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.
- At the bottom of this dialog box, you can change the default project name WindowsApplication1 to some other name you like, for example, myFirstProgram.
- After you have renamed the project, click OK to continue. The following IDE Windows will appear, it is almost similar to Visual Basic 6.
- It consists of an empty form, the toolbox tab and the properties. The layout is slightly different from vb2008 as the Toolbox is not shown until you click on the Toolbox tab.

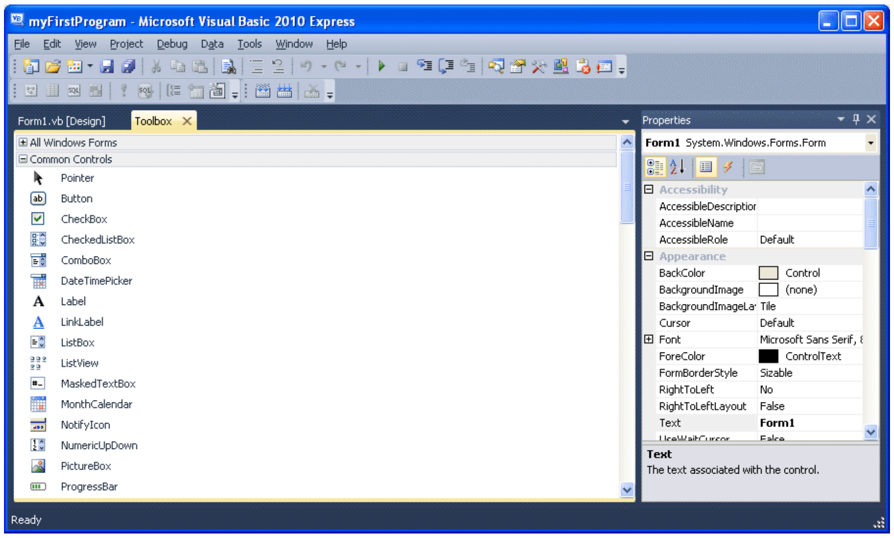

myFirstProgram - Microsoft Visual Basic 2010 Express
File Edit View Project Debug Data Format Tools Window Help
Form1.vb [Design]
Toolbox
Form1
Properties
Ready

Form1 System.Windows.Forms.Form

Accessibility	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayout	Tile
Cursor	Default
Font	Microsoft Sans Serif, 8
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	Form1
UseWaitCursor	False

Text
The text associated with the control.

- When you click on the Toolbox tab, the common controls Toolbox will appear.

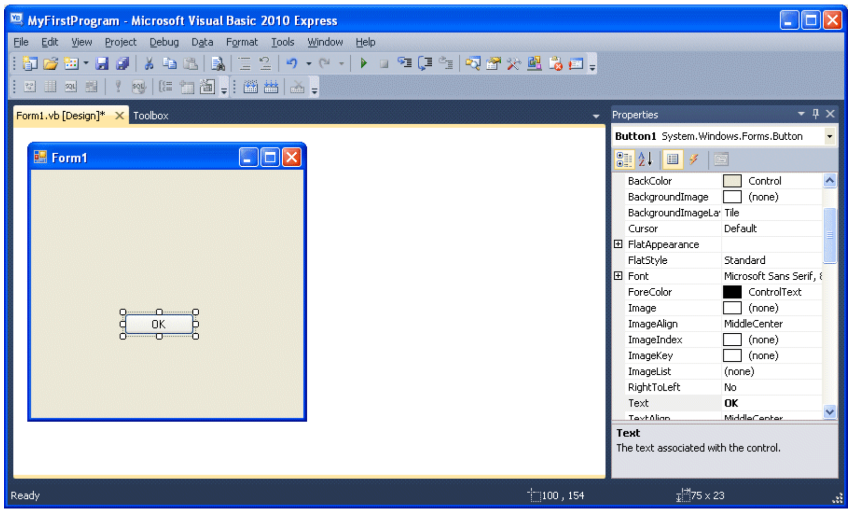


Form1 System.Windows.Forms.Form

Accessibility	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayout	Tile
Cursor	Default
Font	Microsoft Sans Serif, 8
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	Form1
UseWaitCursor	False

Text
The text associated with the control.

- Now drag the button control into the form, and change its default Text Button1 to OK in the properties window, the word OK will appear on the button in the form, as shown below:


MyFirstProgram - Microsoft Visual Basic 2010 Express
File Edit View Project Debug Data Format Tools Window Help
Form1.vb [Design]*
Toolbox
Form1
OK
Properties
Ready

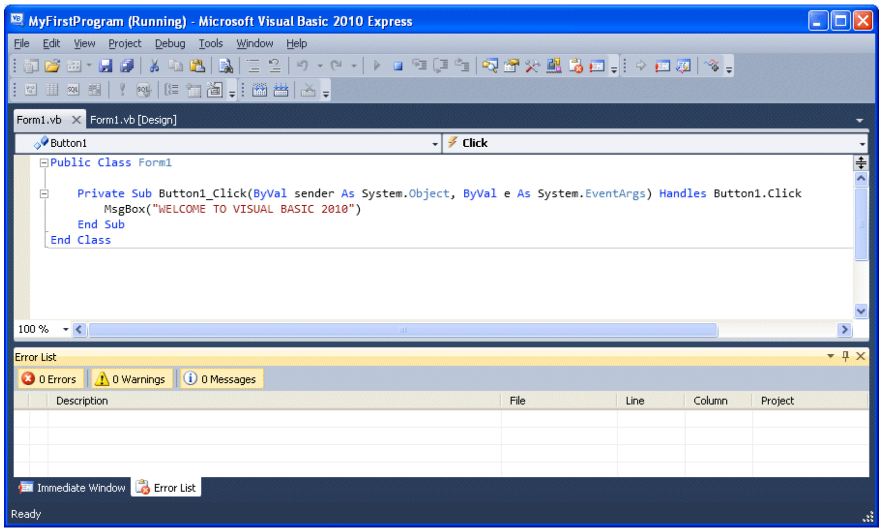
Button1 System.Windows.Forms.Button

BackColor	Control
BackgroundImage	(none)
BackgroundImageLa	Tile
Cursor	Default
FlatAppearance	
FlatStyle	Standard
Font	Microsoft Sans Serif, 8
ForeColor	ControlText
Image	(none)
ImageAlign	MiddleCenter
ImageIndex	(none)
ImageKey	(none)
ImageList	(none)
RightToLeft	No
Text	OK
TextAlign	MiddleCenter

Text
The text associated with the control.

100 , 154 75 x 23

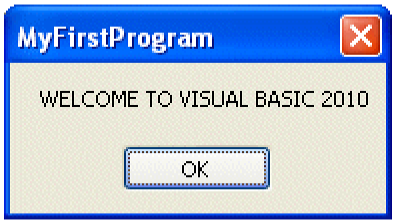
Next, click on the OK button and the code window appears. Enter the code as follows:



```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        MsgBox("WELCOME TO VISUAL BASIC 2010")
    End Sub
End Class
```

When you run the the program and click on the OK button, a dialog box will appear and display the "WELCOME TO VISUAL BASIC 2010" message,as shown below:



WELCOME TO VISUAL BASIC 2010

EVENT DRIVEN PROGRAMMING

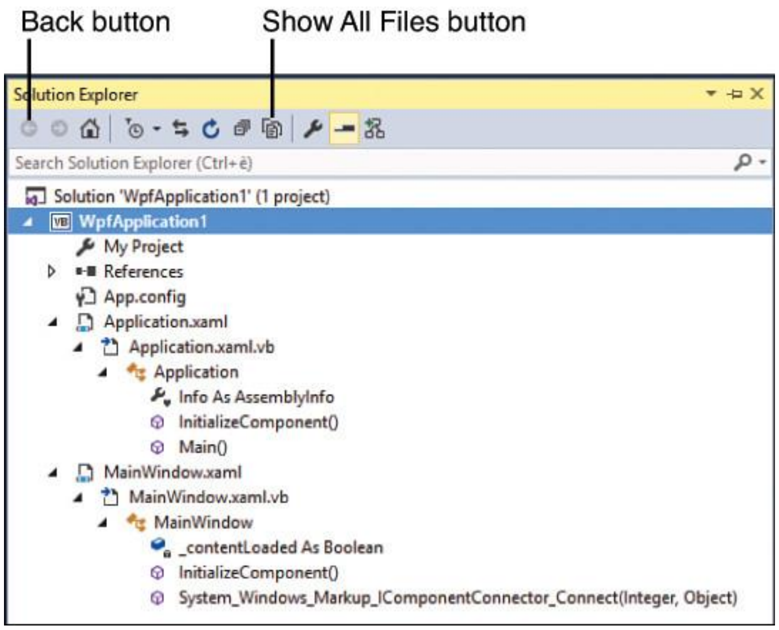
- In traditional or procedural application, the application itself determines which portion of code is to be executed and in what sequence.
- Generally execution starts with the 1st line of code and follow the coding sequence define in the application.
- Whereas application written in VB are ‘Event-Driven’. In an event-driven application the code doesn’t follow a pre-determined path rather it execute different code sections in response to events.
- Event can be triggered by user’s action, by message from system, other applications or even from the application itself.
- The sequences of these events determine the order in which the code execute and associated with the objects of application.
- They either act on an object or are triggered by an object to control the flow of execution when it is running. That is why VB called Event-Driven programming language

VB.NET DEVELOPMENT ENVIRONMENT

- Microsoft provides the following development tools for VB.Net programming −
  - Visual Studio 2010 (VS)
  - Visual Basic 2010 Express (VBE)
  - Visual Web Developer
- The last two are free. Using these tools, you can write all kinds of VB.Net programs from simple command-line applications to more complex applications.
- Visual Basic Express and Visual Web Developer Express edition are trimmed down versions of Visual Studio and has the same look and feel.
- They retain most features of Visual Studio. In this tutorial, we have used Visual Basic 2010 Express and Visual Web Developer

SOLUTION EXPLORER

- Solution Explorer is a special window that enables you to manage solutions, projects, and files.
- It provides a complete view of the files in a project, and it enables you to add or remove files and to organize files into subfolders.



- As you can see, the project is at the root level. Nested under it are code files and subfolders containing pictures, data, and documents.
- You can also get a list of all the references in the project. You use Solution Explorer to add and manage items in projects.

TOOLBOX

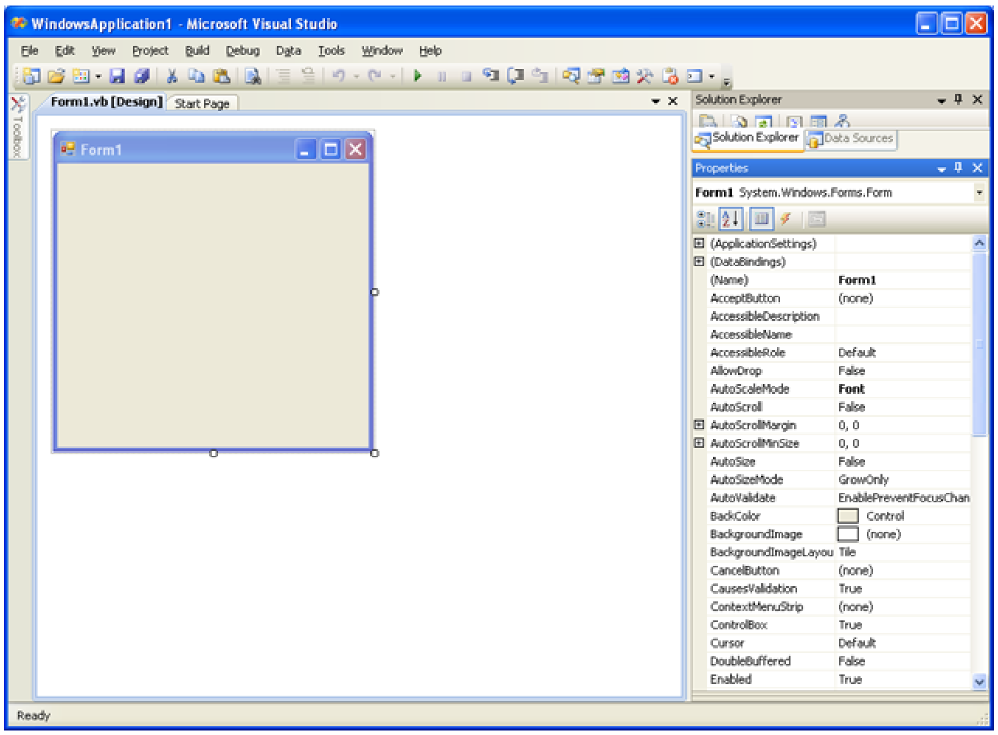
- Toolbox in Visual Basic.net consist of Controls, Containers, Menu Options, Crystal Report Controls, Data Controls, Dialogs, Components, Printing controls, that are used in a form to design the interfaces of an application.

Images	Control Name	Description
↖	Pointer	Used to move and resize controls and forms.
ab	Button	This Control triggers an action when accessed.
☑	Check Box	Control that has values either true or false
	CheckedList Box	Lists check box next to each item
	Combo Box	A combination of list and text box controls that enables to select as well as edit text.
	DateTimePicker	Display a calender picker to choose the day and date.
A	Label	Displays a label text.
A	LinkLabel	Displays a label with a link text.
	List Box	Control that lists number of items.
	List View	Extension of ListBox control with options to add icons,headings.
#_	Masked Text Box	Uses a Mask to differetiate proper and improper text input.
	MonthCalendar	Enable to select date at runtime
	Notify Icon	Displays an icon in the Windows Tray
	NumericUpDown	Allows to input a integer of specific decimal places within a specific range.
	Picture Box	Display image files
	Progress Bar	Display the progress of a task.

	Radio Button	Allows to choose a choice from a group of choices.
	Rich TextBox	Allows to edit, input rich text.
	Text Box	Control used to input or display text.
	ToolTip	Displays tooltip text.
	TreeView	Displays the hierarchy of nodes.
	WebBrowser	Allows to open an html document in form.

PROPERTIES WINDOW

- Windows form properties in Visual Basic.net lists the properties of an selected object.
- Every object in VB has it own properties that can be used to change the look and even the functionality of the object.
  Properties Window lists the properties of the forms and controls in an alphabetical order by default.
- User also has options to view the properties based on the category they belong to.


WindowsApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Help
Form1.vb [Design]
Start Page
Toolbox
Form1
Solution Explorer
Solution Explorer
Data Sources
Properties
Ready

```
Form1 System.Windows.Forms.Form
⊞ (ApplicationSettings)
⊞ (DataBindings)
  (Name)                 Form1
  AcceptButton           (none)
  AccessibleDescription
  AccessibleName
  AccessibleRole         Default
  AllowDrop              False
  AutoScaleMode          Font
  AutoScroll             False
⊞ AutoScrollMargin       0, 0
⊞ AutoScrollMinSize      0, 0
  AutoSize               False
  AutoSizeMode           GrowOnly
  AutoValidate           EnablePreventFocusChan
  BackColor              Control
  BackgroundImage        (none)
  BackgroundImageLayou   Tile
  CancelButton           (none)
  CausesValidation       True
  ContextMenuStrip       (none)
  ControlBox             True
  Cursor                 Default
  DoubleBuffered         False
  Enabled                True
```

FORM DESIGNER

Windows Forms Designer in Visual Studio provides a rapid development solution for creating Windows Forms-based applications. Windows Forms Designer lets you easily add controls to a form, arrange them, and write code for their events.

Creating a Form Using Visual Studio . NET

1. Select File→New→Project.
2. Select Visual Basic Projects in the Project Types pane on the left side of the dialog box.
3. Select Windows Application in the Templates pane on the right side of the dialog box.
4. Enter a name in the Name text box.
5. Click OK.

OUTPUT WINDOW

The Output window is **where many of the tools, including the compiler, send their output**. Every time you start an application, a series of messages is displayed in the Output window. These messages are generated by the compiler, and you need not understand them at this point.

OBJECT BROWSER

The **Object Browser** allows you to browse through all available objects in your project and see their properties, methods and events. In addition, you can see the procedures and constants that are available from object libraries in your project. You can easily display online Help as you browse. Use the **Object Browser** to find and use objects that you create, as well as objects from other applications.

THE VB.NET LANGUAGE - DATA TYPES

In **VB.NET, data type** is used to define the type of a variable or function in a program. Furthermore, the conversions of one data type to another type using the data conversion function.

A **Data Type** refers to which type of data or value is assigning to a variable or function so that a variable can hold a defined data type value. For example, when we declare a variable, we have to tell the compiler what type of data or value is allocated to different kinds of variables to hold different amounts of space in computer memory.

Syntax:

Dim Variable_Name as DataType

VariableName: It defines the name of the variable that you assign to store values.

DataType: It represents the name of the data type that you assign to a variable.

Data Types	Required Space	Value Range
Boolean	A Boolean type depends on the implementing platform	True or False
Byte	1 byte	Byte Range start from 0 to 255 (unsigned)
Char	2 bytes	Char Range start from 0 to 65535 (unsigned)
Date	8 bytes	Date range can be 0:00:0 (midnight) January 1, 0001 to 11:5959 PM of December 31, 9999.
Decimal	16 bytes	Range from 0 to +/- 79,228,162,514,264,337,593,543,950,335 (+/-7.9…E+28) without any decimal point; And 0 to +/-

		7.9228162514226433759354395035 with 28 position to the right of the decimal
Double	8 bytes	-1.79769313486231570E+308 to -4.94-65645841246544E-324 for negative values; 4.94065645841246544E-324 to 1.79769313486231570E+308, for positive values
Integer	4 bytes	-2,147,483,648 to 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (9.2…E + 18) (signed)
Object	Object size based on the platform such as 4 bytes in 32-bit and 8 bytes in 64-bit platform	It can store any type of data defined in a variable of type Object
SByte	1 byte	-128 to 127 (signed)
Short	2 bytes	-32,768 to 32,767 (signed)
Single	4 bytes	-3.4028235E + 38 to -1.401298E-45 for negative values; And for positive value: 1.401298E-45 to 3.4028235E + 38.
String	String Datatype depend on the implementing platform	It accepts Unicode character from 0 to approximately 2 billion characters.
UInteger	4 bytes	The range start from 0 to 4,294,967,295 (unsigned)
ULong	8 bytes	The range of ULong start from 0 to 18,446,744,073,709,551,615 (1.8…E + 19) (unsigned)
User-Defined	A user-defined data type depends on the	Each member of the structure has its own data type and limits independent of the other

(structure)	implementing platform	members' ranges.
UShort	2 bytes	Range from 0 to 65,535 (unsigned)

Program on various data types in a VB.NET

Data_type.vb

```
Module Data_type
   Sub Main()
      Dim b As Byte = 1
      Dim num As Integer = 5
      Dim si As Single
      Dim db As Double
      Dim get_date As Date
      Dim c As Char
      Dim str As String
      b = 1
      num = 20
      si = 0.12
      db = 2131.787
      get_date = Today
      c = "A"
      str = "Hello Friends..."
      Console.WriteLine("Welcome to the JavaTpoint")
      Console.WriteLine("Byte is: {0}", b)
      Console.WriteLine("Integer number is: {0}", num)
      Console.WriteLine("Single data type is: {0}", si)
      Console.WriteLine("Double data type is: {0}", db)
      Console.WriteLine("Today is: {0}", get_date)
      Console.WriteLine("Character is: {0}", b)
      Console.WriteLine("String message is: {0}", str)
      Console.ReadKey()
```

```
    End Sub
End Module
```

Output:

```
Welcome to the JavaTpoint
Byte is: 1
Integer number is: 20
Single data type is: 0.12
Double data type is: 2131.787
Today is: 31-05-2020 00:00:00
Character is: 1
String message is: Hello Friends...
```

Type Conversion Functions in VB.NET

1. **CBool(expression):** It is used to convert an expression into a Boolean data type.
2. **CByte(expression):** It is used to convert an expression to a Byte data type.
3. **CChar(expression):** It is used to convert an expression to a Char data type.
4. **CDate(expression):** It is used to convert an expression to a Date data type.
5. **CDbl(expression):** It is used to convert an expression into a Double data type.
6. **CDec(expression):** It is used to convert an expression into a Decimal data type.
7. **CInt(expression):** It is used to convert an expression to an Integer data type.
8. **CLng(expression):** It is used to convert an expression to a Long data type.
9. **CObj(expression):** It is used to convert an expression to an Object data type.
10. **CSByte(expression):** It is used to convert an expression to an SByte data type.
11. **CShort(expression):** It is used to convert an expression to a Short data type.
12. **CSng(expression):** It is used to convert an expression into a Single data type.
13. **CStr(expression):** It is used to convert an expression into a String data type.
14. **CUInt(expression):** It is used to convert an expression to a UInt data type.
15. **CULng(expression**): It is used to convert an expression to a ULng data type.
16. **CUShort(expression)**: It is used to convert an expression into a UShort data type.

Program we have performed different conversion.

DB_Conversion.vb

```
Module DB_Conversion
    Sub Main()
        Dim dblData As Double
        dblData = 5.78
        Dim A, B As Char
        Dim bool As Boolean = True
        Dim x, Z, B_int As Integer
        A = "A"
        B = "B"
        B_int = AscW(B)
        Console.WriteLine(" Ascii value of B is {0}", B_int)
        x = 1
        Z = AscW(A)
        Z = Z + x
        Console.WriteLine("String to integer {0}", Z)
        Console.WriteLine("Boolean value is : {0}", CStr(bool))
        Dim num, intData As Integer
         num = CInt(dblData)
        intData = CType(dblData, Integer)
        Console.WriteLine(" Explicit conversion of Data type " & Str(intData))
        Console.WriteLine(" Value of Double is: {0}", dblData)
        Console.WriteLine("Double to Integer: {0}", num)
        Console.ReadKey()
    End Sub
End Module
```

Output:

```
Ascii value of B is 66
String to integer 66
Boolean value is: True
 Explicit conversion of Data type 6
 Value of Double is: 5.78
Double to Integer: 6
```

What is a Variable?

- A variable is a simple name used to store the value of a specific data type in computer memory.
- In VB.NET, each variable has a particular data type that determines the size, range, and fixed space in computer memory.
- With the help of variable, we can perform several operations and manipulate data values in any programming language.

VB.NET Variables Declaration

- The declaration of a variable is simple that requires a variable name and data type followed by a Dim.
- A Dim is used in Class, Module, structure, Sub, procedure.

Syntax:

Dim [Variable_Name] As [Defined Data Type]

Name	Descriptions
Dim	It is used to declare and allocate the space for one or more variables in memory.
Variable_Name	It defines the name of the variable to store the values.
As	It is a keyword that allows you to define the data type in the declaration statement.
Data Type	It defines a data type that allows variables to store data types such as Char, String, Integer, Decimal, Long, etc.
Value	Assign a value to the variable.

There are some valid declarations of variables along with their data type definition, as shown below:

```
Dim Roll_no As Integer
Dim Emp_name As String
Dim Salary As Double
```

Dim Emp_id, Stud_id As Integer

Dim result_status As Boolean

Further, if we want to **declare more than one variable** in the same line, we must separate each variable with a comma.

Syntax

Dim Variable_name1 As DataType1, variable_name2 As DataType2, Variable_name3 As DataType3

Static name As String

Public bill As Decimal = 0

VB.NET Variable Initialization

After the declaration of a variable, we must assign a value to the variable. The following syntax describes the initialization of a variable:

Syntax:

Variable_name = value

For example:

Dim Roll_no As Integer

Roll_no = 101

Initialize the Emp_name

Dim Emp_name As String

Emp_name = "David"

Dim status As Boolean

status = True

We can also initialize a variable at the time of declaration:

Dim Roll_no As Integer = 101

Dim Emp_name As String = " Stephen Robert "

Program to use different types of variable declaration and initialization in VB.NET.

Variable1.vb

```
Module Variable1
Sub Main()
  Dim intData As Integer
  Dim CharData As Char
  Dim strData As String
  Dim dblData As Double
  Dim single_data As Single
  intData = 10
  CharData = "A"
  strData = " VB.NET is a Programming Language."
  dblData = 4567.676
  single_data = 23.08
  Console.WriteLine(" Value of intData is: {0}", intData)
  Console.WriteLine(" Value of CharData is: {0}", CharData)
  Console.WriteLine(" Value of strData is: {0}", strData)
  Console.WriteLine(" Value of dblData is: {0}", dblData)
  Console.WriteLine(" Value of single_data is: {0}", single_data)
  Console.WriteLine("press any key to exit...")
  Console.ReadKey()
 End Sub
End Module
```

Output:

```
Value of intData is: 10
 Value of CharData is: A
 Value of strData is:  VB.NET is a Programming Language.
 Value of dblData is: 4567.676
 Value of single_data is: 23.08
press any key to exit...
```

Getting Values from the User in VB.NET

In VB.NET, the Console class provides the Readline() function in the System namespace. It is used to take input from the user and assign a value to a variable. For example:

```
Dim name As String
name = Console.ReadLine()
Or name = Console.ReadLine
```

Program that takes input from the user.

User_Data.vb

```
Imports System
Module User_Data
Sub Main()
  Dim num As Integer
  Dim age As Double
  Dim name As String
  Console.WriteLine("Enter your favourite number")
  num = Console.ReadLine
  Console.WriteLine(" Enter Your Good name")
  name = Console.ReadLine
  Console.WriteLine(" Enter your Age")
  age = Console.ReadLine
  Console.WriteLine(" You have entered {0}", num)
  Console.WriteLine(" You have entered {0}", name)
  Console.WriteLine(" You have entered {0}", age)
  Console.ReadKey()
End Sub
End Module
```

Output:

```
Enter your favourite number
7
 Enter Your Good name
```

```
Alexander
 Enter your Age
27.5
 You have entered 7
 You have entered Alexander
 You have entered 27.5
```

SCOPE OF A VARIABLE

- The scope of a variable determines the accessible range of a defined variable at the time of declaration in any block, module, and class.
- The following are the methods to represent the scope of a variable in VB.NET.
  1. Procedure Scope
  2. Module Scope
  3. Public Scope

Procedure (local) scope

- A **local variable** is a type of variable defined within a procedure scope, block, or function.
- It is available with a code inside the procedure, and it can be declared using the **Dim or static** statement.
- These variables are not accessible from outside of the local method.
- However, the local variable can be easily accessed by the nested programming function in the same method.
- Dim X As Integer
- Local variables exist until the procedure in which they are declared is executed. Once a procedure is executed, the values of its local variables will be lost, and the resources used by these variables will be released. And when the block is executed again, all the local variables are rearranged.

Program that displays the local scope of a variable within a function.

Local_Scope.vb

```
Module Local_scope
    Sub Main()
        Console.WriteLine(" Scope of local varibale within a function")
        local()
        local2()
        Console.WriteLine("press any key to exit...")
        Console.ReadKey()
    End Sub
    Sub local()
        Dim X As Integer
        X = 50
        Console.WriteLine(" Value of Local value X is {0}", X)
     End Sub
    Sub local2()
        Dim X As String
        X = "JavaTpoint"
        Console.WriteLine(" Value of X is {0}", X)
    End Sub
End Module
```

Output:

```
Scope of local variable within a function
 Value of Local value X is 50
 Value of X is JavaTpoint
press any key to exit...
```

Module Scope

- All existing procedures can easily identify a variable that is declared inside a module sheet is called a **module-level variable**.
- The defined module variable is **visible to all procedure**s within that module only, but it is not available for other module's procedures.
- The **Dim or private statement** at the top of the first procedure declaration can be declared the module-level variables.
- It means that these variables cannot be declared inside any procedure block.
- Further, these variables are useful to share information between the procedures in the same module.
- And one more thing about the module-level variable is that these variables can remains existence as long as the module is executed.
- It is the declaration section of the module
- Private num As Integer ' A **private** module-level variable
- Dim name As String ' Another **private** module-level variable

Program that display the module level variable in VB.NET.

Module_scope.vb

```
Module Module_scope
   Dim x As Integer
   Private y As Integer
   Private name As String = "JavaTpoint"
   Sub example()
      x = 10
      y = x + 10
      Console.WriteLine(" Value of Y is {0}", y)
   End Sub
   Sub example2()
      Console.WriteLine(" Value of X is {0}", x)
      Console.WriteLine(" Value of Y is {0}", y)
```

```
        Console.WriteLine(" Name is {0}", name)
    End Sub
    Sub example3()
        Dim A As Integer  ' local variable or local scope
        A = x + y
        Console.WriteLine(" Local scope within a function of variable A {0}", A)
    End Sub
    Sub Main()
        Console.WriteLine(" Module scope of variable")
        example()
        example2()
        example3()
        Console.WriteLine("Press any key to exit...")
        Console.ReadKey()
    End Sub
End Module
```

Output:

```
Module scope of variable
 Value of Y is 20
 Value of X is 10
 Value of Y is 20
 Name is JavaTpoint
 Local scope within a function of variable A 30
Press any key to exit...
```

Global (Public) Scope

- As the name defines, a global variable is a variable that is used to access the variables **globally in a program**.
- It means these variables can be accessed by all the **procedures or modules** available in a program.

- To access the variables globally in a program, you need to use **the friend or public keyword** with a variable in a module or class at the top of the first procedure function.
- Global scope is also known as the **Namespace scope**.

Program that uses the global variable.

Global_scope1.vb

```
Module Global_scope1
  Public str As String = "Hello, Programmer."
  Public topic As String
  Public exp As Integer
   Sub Main()
     Console.WriteLine(" You have passed {0}", str)
     Console.WriteLine(" Enter the topic name")
     topic = Console.ReadLine
     Console.WriteLine(" Topic Name :{0}", topic)
     Console.WriteLine("How many years of experienced in {0}?", topic)
     exp = Console.ReadLine
     Console.WriteLine(" Your Experienced is {0} ", exp)
     Console.ReadKey()
  End Sub
 End Module
```

Output:

```
You have passed Hello, Programmer
 Enter the topic name
VB.NET
 Topic Name :VB.NET
How many years of experienced in VB.NET?
10
 Your Experienced is 10
```

TYPE CASTING

Casting is the process of converting one data type to another. For example, casting an Integer type to a String type. The following functions are available for conversion.

- **CBool(expression):** It is used to convert an expression into a Boolean data type.
- **CByte(expression):** It is used to convert an expression to a Byte data type.
- **CChar(expression):** It is used to convert an expression to a Char data type.
- **CDate(expression):** It is used to convert an expression to a Date data type.
- **CDbl(expression):** It is used to convert an expression into a Double data type.
- **CDec(expression):** It is used to convert an expression into a Decimal data type.
- **CInt(expression):** It is used to convert an expression to an Integer data type.
- **CLng(expression):** It is used to convert an expression to a Long data type.
- **CObj(expression):** It is used to convert an expression to an Object data type.
- **CSByte(expression):** It is used to convert an expression to an SByte data type.
- **CShort(expression):** It is used to convert an expression to a Short data type.
- **CSng(expression):** It is used to convert an expression into a Single data type.
- **CStr(expression):** It is used to convert an expression into a String data type.
- **CUInt(expression):** It is used to convert an expression to a UInt data type.
- **CULng(expression**): It is used to convert an expression to a ULng data type.
- **CUShort(expression)**: It is used to convert an expression into a UShort data type.

Program we have performed different conversion.

DB_Conversion.vb

```
Module DB_Conversion
    Sub Main()
        Dim dblData As Double
        dblData = 5.78
        Dim A, B As Char
        Dim bool As Boolean = True
        Dim x, Z, B_int As Integer
        A = "A"
        B = "B"
        B_int = AscW(B)
       Console.WriteLine(" Ascii value of B is {0}", B_int)
        x = 1
        Z = AscW(A)
        Z = Z + x
        Console.WriteLine("String to integer {0}", Z)
        Console.WriteLine("Boolean value is : {0}", CStr(bool))
        Dim num, intData As Integer
         num = CInt(dblData)
        intData = CType(dblData, Integer)
        Console.WriteLine(" Explicit conversion of Data type " & Str(intData))
        Console.WriteLine(" Value of Double is: {0}", dblData)
        Console.WriteLine("Double to Integer: {0}", num)
        Console.ReadKey()
    End Sub
End Module
```

Output:

```
Ascii value of B is 66
String to integer 66
Boolean value is: True
 Explicit conversion of Data type 6
 Value of Double is: 5.78
Double to Integer: 6
```

CONSTANTS

- As the name suggests, the name constant refers to a fixed value that cannot be changed during the execution of a program. It is also known as **literals**.
- These constants can be of any data type, such as Integer, Double, String, Decimal, Single, character, enum, etc.

Declaration of Constants

- In VB.NET, **const** is a keyword that is used to declare a variable as constant. The Const statement can be used with module, structure, procedure, form, and class.

Syntax:

Const constname As datatype = value

Item Name	**Descriptions**
Const	It is a Const keyword to declare a variable as constant.
Constname	It defines the name of the constant variable to store the values.
As	It is a keyword that allows you to define the data type in the declaration statement.
Data Type	It defines a data type that allows variables to store data types such as Char, String, Integer, Decimal, Long, etc.
Value	Assign a value to the variable as constant.

Further, if we want to **declare more than one variable** in the same line, we must separate each variable with a comma, as shown below.

The Syntax for defining the multiple variables as constant is:

Dim Variable_name1 As DataType1, variable_name2 As DataType2, Variable_name3 As DataType3

```
Const num As Integer = 10
Static name As String
Public Const name As String = "JavaTpoint"
Private Const PI As Double = 3.14
```

Example of Const keyword

Const1.vb

```
Module Const1
    Sub main()
        Const intData As Integer = 20
        Const name As String = "JavaTpoint"
        Const topic As String = "VB.NET"
        Const PI = 3.14
        Dim radius, area As Integer
         Console.WriteLine(" Constant integer is {0}", intData)
        Console.WriteLine(" You have entered {0}", name)
        Console.WriteLine(" Your Topic is {0}", topic)
        Console.WriteLine("Enter the Radius")
        radius = Console.ReadLine()
        area = PI * radius * radius
        Console.WriteLine(" Area of Circle is {0}", area)
        Console.ReadKey()
      End Sub
End Module
```

Output:

```
Constant integer is 20
 You have entered JavaTpoint
 Your Topic is VB.NET
Enter the Radius
7
 Area of Circle is 154
```

OPERATORS AND EXPRESSIONS

In VB.NET, **operator** is a special symbol that tells the compiler to perform the specific logical or mathematical operation on the data values. The data value itself (which can be either a variable or a constant) is called an **operand,** and the Operator performs various **operations** on the operand.

For example: In the expression,

3 + 2 - 1

The symbol **+** and **-** are the Operators, and the 3, 2, and 1 are operands.

Different Types of VB.NET Operators

Following are the different types of Operators available in VB.NET:

- Arithmetic Operators
- Comparison Operators
- Logical and Bitwise Operators
- Bit Shift Operators
- Assignment Operators
- Concatenation Operators

Arithmetic Operators

The Arithmetic Operators in VB.NET, used to perform mathematical operations such as **subtraction, addition, multiplication, division,** etc. on the operands in VB.NET. These are as follows:

Operators	**Description**	**Example**
^	It is an exponentiation Operator that is used to raises one operand to the power of another operand.	Y ^ X (X to the power Y)
+	The addition Operator is used to add numeric data, as well as concatenate two string variables.	X + Y
-	It is a subtraction Operator, which is used to subtract the second operand from the first operand.	X - Y
*****	The multiplication Operator is used to multiply the operands	X * Y
/	It is a division Operator used to divide one operand by another operand and returns a floating-point result.	X / Y
\	It is an integer division Operator, which is similar to division Operator, except that it returns an integer result while dividing one operand to another operand.	X \\ Y
Mod	It is a modulo (Modulus) Operator, which is used to divide two operands and returns only a remainder.	X Mod Y

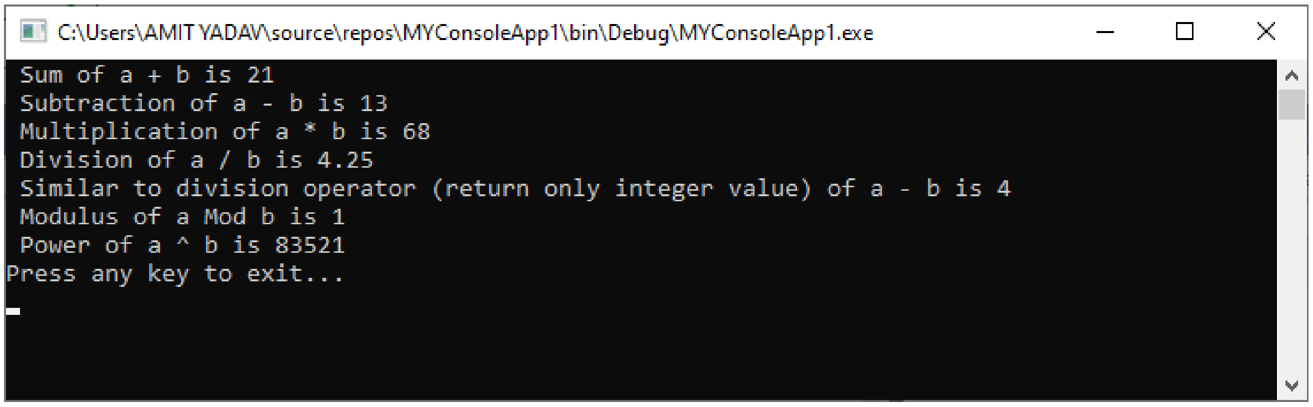
Example of **Arithmetic Operators in VB.NET:**

Arithmetic_Operator.vb

```
Module Arithmetic_Operator
   Sub Main()
      Dim a, b, c As Integer
      Dim d As Single
      a = 17
      b = 4
      c = a + b
```

```
        Console.WriteLine(" Sum of a + b is {0}", c)
        c = a - b
        Console.WriteLine(" Subtraction of a - b is {0}", c)
         c = a * b
        Console.WriteLine(" Multiplication of a * b is {0}", c)
        d = a / b
        Console.WriteLine(" Division of a / b is {0}", d)
         c = a \ b
        Console.WriteLine(" Similar to division Operator (return only integer value) of a -
b is {0}", c)
        c = a Mod b
        Console.WriteLine(" Modulus of a Mod b is {0}", c)
         c = a ^ b
        Console.WriteLine(" Power of a ^ b is {0}", c)
        Console.WriteLine("Press any key to exit...")
        Console.ReadKey()
    End Sub
End Module
```

OUTPUT



C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe

```
 Sum of a + b is 21
 Subtraction of a - b is 13
 Multiplication of a * b is 68
 Division of a / b is 4.25
 Similar to division operator (return only integer value) of a - b is 4
 Modulus of a Mod b is 1
 Power of a ^ b is 83521
Press any key to exit...
```

Comparison Operators

- As the name suggests, the Comparison Operator is used to compare the value of two variables or operands for the various condition such as greater, less than or equal, etc. and returns a Boolean value either true or false based on the condition.

Operator	Description	Example
=	It checks whether the value of the two operands is equal; If yes, it returns a true value, otherwise it shows False.	(A = B)
<>	It is a Non-Equality Operator that checks whether the value of the two operands is not equal; it returns true; otherwise, it shows false.	(A <> B), check Non-Equality
>	A greater than symbol or Operator is used to determine whether the value of the left operand is greater than the value of the right operand; If the condition is true, it returns TRUE; otherwise, it shows FALSE value.	(A > B); if yes, TRUE, Else FALSE
<	It is a less than symbol which checks whether the value of the left operand is less than the value of the right operand; If the condition is true, it returns TRUE; otherwise, it shows FALSE value.	(A < B); if the condition is true, returns TRUE else FALSE
>=	It is greater than equal to which checks two conditions whether the first operand is greater than or equal to the second operand; if yes, it returns TRUE; otherwise, it shows False.	A >= B
<=	This symbol represents less than equal to	A <= B

	which determines the first operand is less than or equal to the second operand, and if the condition is true, it returns TRUE; otherwise, it shows FALSE.	
Is	The Is Operator is used to validate whether the two objects reference the same variable or object; If the test is true, it returns True; otherwise, the result is False. In short, it checks the equality of the objects. An Is Operator is also used to determine whether the object refers to a valid object.	result = obj1 Is obj2
IsNot	The IsNot Operator is similar to Is Operator, except that the two object references the different object; if yes, the result is True; otherwise, the result is False.	Result = obj1 IsNot obj2
Like	The Like Operator is used to check the pattern expression of string variable; And if the pattern matched, the result is True; otherwise, it returns False.	result = string Like the pattern, the pattern represents the series of characters used by Like Operator.

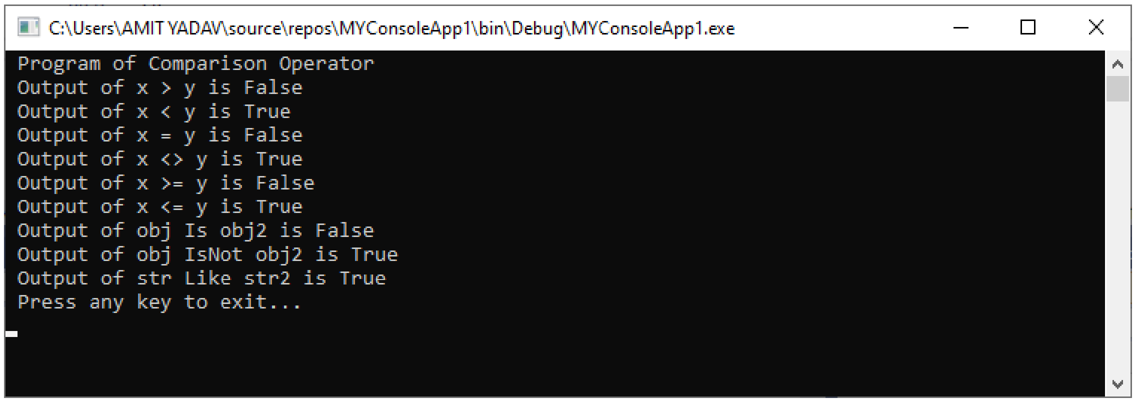
Example of Comparison Operators in VB.NET

Comparison_Operator.vb

```
Module Comparison_Operator
    Sub Main()
        Dim x As Integer = 5
        Dim y As Integer = 10
        Dim Result, obj, obj2 As Object
        Dim str, str2 As String
        str = "Apple12345"
        str2 = "Apple12345"
```

```
  obj = 10
  obj2 = 20
  Console.WriteLine(" Program of Comparison Operator")
 Console.WriteLine(" Output of x > y is {0}", x > y)
 Console.WriteLine(" Output of x < y is {0}", x < y)
 Console.WriteLine(" Output of x = y is {0}", x = y)
 Console.WriteLine(" Output of x <> y is {0}", x <> y)
  Console.WriteLine(" Output of x >= y is {0}", x >= y)
  Console.WriteLine(" Output of x <= y is {0}", x <= y)
  Result = obj Is obj2
  Console.WriteLine(" Output of obj Is obj2 is {0}", Result)
   Result = obj IsNot obj2
  Console.WriteLine(" Output of obj IsNot obj2 is {0}", Result)
   Result = str Like str2
  Console.WriteLine(" Output of str Like str2 is {0}", Result)
   Console.WriteLine(" Press any key to exit...")
  Console.ReadKey()
    End Sub
End Module
```

OUTPUT



C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe

```
Program of Comparison Operator
Output of x > y is False
Output of x < y is True
Output of x = y is False
Output of x <> y is True
Output of x >= y is False
Output of x <= y is True
Output of obj Is obj2 is False
Output of obj IsNot obj2 is True
Output of str Like str2 is True
Press any key to exit...
```

Logical and Bitwise Operators

- The logical and bitwise Operators work with Boolean (true or false) conditions, and if the conditions become true, it returns a Boolean value.
- The following are the logical and bitwise Operators used to perform the various logical operations such as And, Or, Not, etc. on the operands (variables).
- Suppose there are two operand A and B, where A is True, and B is False.

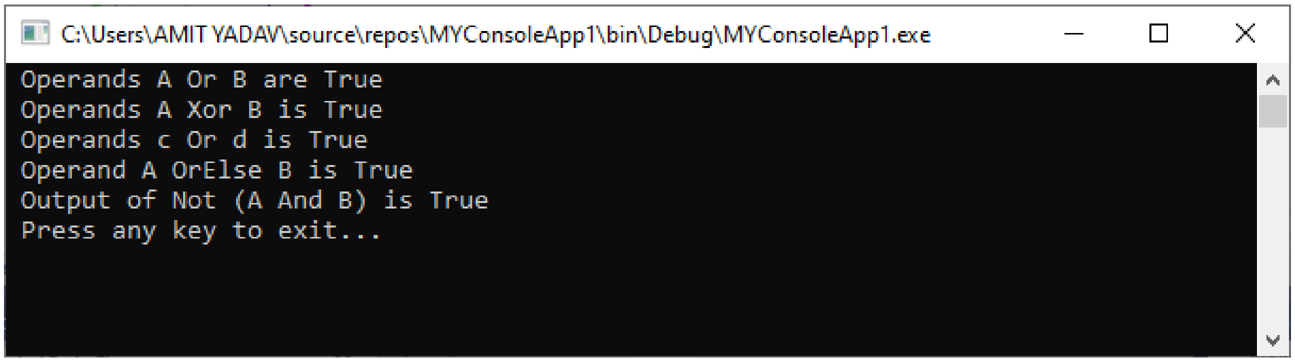
Operator	Description	Example
And	The And Operator represents, whether both the operands are true; the result is True.	(A And B), result = False
Or	It is an Or Operator that returns a true value; if anyone operand is true from both the operands.	(A Or B), result = True
Not	The Not Operator is used to reverse the logical condition. For example, if the operand's logic is True, it reveres the condition and makes it False.	Not A Or Not(A And B) is True
Xor	It is an Exclusive OR Operator that represents, whether both the expression is true or false, the result is True; otherwise, the result is False.	A Xor B is True
AndAlso	It is a logical AND Operator that performs short-circuit operation on the variables, and if both the operands are true, the result is True else the result is False.	A AndAlso B = False
OrElse	It is a logical OR Operator that perform short-circuit operation on Boolean data. If anyone of the operand is true, the result is True else the result is False.	A OrElse B = True
IsFalse	The IsFalse Operator is used to determine whether an expression is False.	
IsTrue	The IsTrue Operator is used to determine whether an expression is True.	

Example of Logical and Bitwise Operator:

Logic_Bitwise.vb

```
Module Logic_Bitwise
    Sub Main()
        Dim A As Boolean = True
        Dim B As Boolean = False
        Dim c, d As Integer
        c = 10
        d = 20
        If A And B Then
            Console.WriteLine(" Operands A And B are True")
        End If
        If A Or B Then
            Console.WriteLine(" Operands A Or B are True")
        End If
        If A Xor B Then
            Console.WriteLine(" Operands A Xor B is True")
        End If
        If c And d Then
            Console.WriteLine(" Operands c And d is True")
        End If
        If c Or d Then
            Console.WriteLine(" Operands c Or d is True")
        End If
         If A AndAlso B Then
            Console.WriteLine(" Operand A AndAlso B is True")
        End If
         If A OrElse B Then
            Console.WriteLine(" Operand A OrElse B is True")
        End If
         If Not (A And B) Then
            Console.WriteLine(" Output of Not (A And B) is True")
        End If
        Console.WriteLine(" Press any key to exit?")
        Console.ReadKey()
    End Sub
End Module
```

OUTPUT



```
C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe
Operands A Or B are True
Operands A Xor B is True
Operands c Or d is True
Operand A OrElse B is True
Output of Not (A And B) is True
Press any key to exit...
```

Bit Shift Operators

- The Bit Shit Operators are used to perform the bit shift operations on binary values either to the right or to the left.

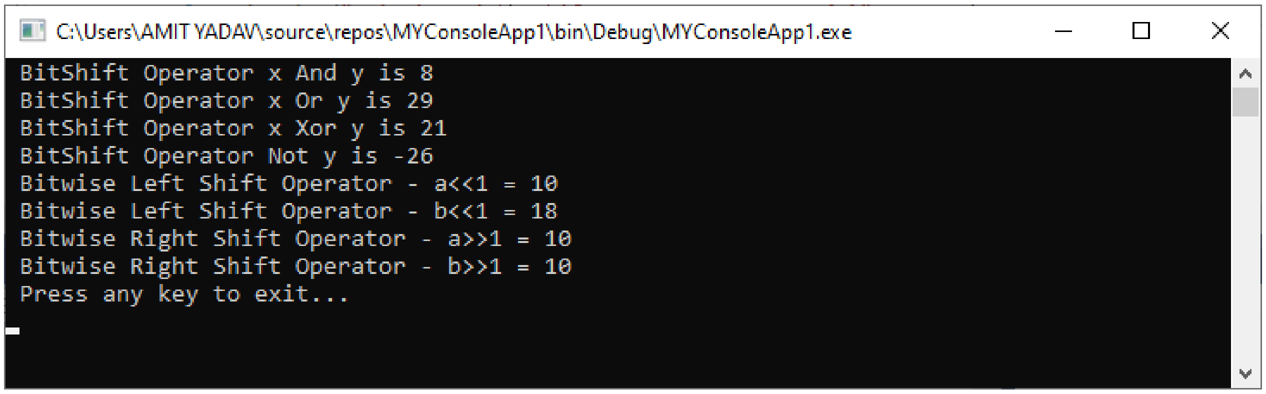
Operator	Description
AND	The Binary AND Operator are used to copy the common binary bit in the result if the bit exists in both operands.
OR	The Binary OR Operator is used to copy a common binary bit in the result if the bit found in either operand.
XOR	The Binary XOR Operator in VB.NET, used to determine whether a bit is available to copy in one operand instead of both.
Not	The binary NOT Operator is also known as the binary Ones' Compliment Operator, which is used to flip binary bits. This means it converts the bits from 0 to 1 or 1 to 0 binary bits.
<<	The Binary Left Shift Operator is used to shift the bit to the left side.
>>	The Binary Right Shift Operator is used to shift the bit to the right side.

Example of Bit Shift Operator in VB.NET:

BitShift_Operator.vb

```
Module Bitshift_Operator
    Sub Main()
        Dim x, y, z As Integer
        x = 12
        y = 25
        Dim a, b As Double
        a = 5 ' a = 5(00000101)
        b = 9 ' b = 9(00001001)
        ' Use of And Operator
        z = x And y
        Console.WriteLine(" BitShift Operator x And y is {0}", z)
        'Use of Or Operator
        z = x Or y
        Console.WriteLine(" BitShift Operator x Or y is {0}", z)
        z = x Xor y
        Console.WriteLine(" BitShift Operator x Xor y is {0}", z)
         z = Not y
        Console.WriteLine(" BitShift Operator Not y is {0}", z)
        'Use of << Left-Shift Operator
        ' Output is 00001010
        Console.WriteLine(" Bitwise Left Shift Operator - a<<1 = {0}", a << 1)
         'Output is 00010010
        Console.WriteLine(" Bitwise Left Shift Operator - b<<1 = {0}", b << 1)
         'Use of >> Right-Shift Operator
        'Output is 00000010
        Console.WriteLine(" Bitwise Right Shift Operator - a>>1 = {0}", a << 1)
        'Output is 00000100
        Console.WriteLine(" Bitwise Right Shift Operator - b>>1 = {0}", a << 1)
        Console.WriteLine(" Press any key to exit...")
        Console.ReadKey()
    End Sub
End Module
```

OUTPUT



C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe

```
BitShift Operator x And y is 8
BitShift Operator x Or y is 29
BitShift Operator x Xor y is 21
BitShift Operator Not y is -26
Bitwise Left Shift Operator - a<<1 = 10
Bitwise Left Shift Operator - b<<1 = 18
Bitwise Right Shift Operator - a>>1 = 10
Bitwise Right Shift Operator - b>>1 = 10
Press any key to exit...
```

Assignment Operators

- The Assignment Operators are used to assign the value to variables in VB.NET.

Operator	Description	Example
=	It is a simple assignment Operator used to assign a right-side operand or value to a left side operand.	X = 5, X assign a value 5 X = P + Q, (P + Q) variables or value assign to X.
+=	An Add AND assignment Operator is used to add the value of the right operand to the left operand. And the result is assigned to the left operand.	X += 5, which means X= X+5 (5 will add and assign to X and then result saved to Left X operand)
-=	It is a Subtract AND assignment Operator, which subtracts the right operand or value from the left operand. And then, the result will be assigned to the left operand.	X -= P, which is same as X = X - P
***=**	It is a Multiply AND assignment Operator, which	X *= P, which is same

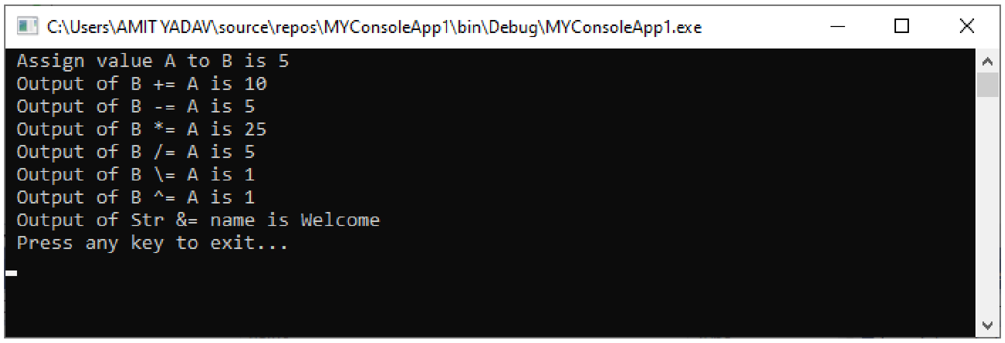
	multiplies the right operand or value with the left operand. And then, the result will be assigned to the left operand.	as X = X - P
/=	It is a Divide AND assignment Operator, which divides the left operand or value with the right operand. And then, the result will be assigned to the left operand (in floating-point).	X /= P, which is same as X = X - P
\=	It is a Divide AND assignment Operator, which divides the left operand or value with the right operand. And then, the result will be assigned to the left operand (in integer-point division).	X \= P, which is same as X = X - P
^=	It is an expression AND assignment Operator, which raises the left operand or value to the right operand's power. And then, the result will be assigned to the left operand.	X ^= P, which is same as X = X ^ P
&=	It is a concatenate string assignment Operator used to bind the right-hand string or variable with the left-hand string or variable. And then, the result will be assigned to the left operand.	Str &= name, which is same as Str = Str & name

Example of Assignment Operator in VB.NET:

Assign_Operator.vb

```
Module Assign_Operator
    Sub Main()
        Dim A As Integer = 5
        Dim B As Integer
        Dim Str, name As String
        name = "come"
        Str = "Wel"
          B = A
        Console.WriteLine(" Assign value A to B is {0}", B)
        B += A
        Console.WriteLine(" Output of B += A is {0}", B)
        B -= A
        Console.WriteLine(" Output of B -= A is {0}", B)
        B *= A
        Console.WriteLine(" Output of B *= A is {0}", B)
        B /= A
        Console.WriteLine(" Output of B /= A is {0}", B)
         B \= A
        Console.WriteLine(" Output of B \= A is {0}", B)
         B ^= A
        Console.WriteLine(" Output of B ^= A is {0}", B)
         Str &= name
        Console.WriteLine(" Output of Str &= name is {0}", Str)
         Console.WriteLine(" Press any key to exit...")
        Console.ReadKey()
    End Sub
End Module
```

OUTPUT



C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe

```
Assign value A to B is 5
Output of B += A is 10
Output of B -= A is 5
Output of B *= A is 25
Output of B /= A is 5
Output of B \= A is 1
Output of B ^= A is 1
Output of Str &= name is Welcome
Press any key to exit...
```

Concatenation Operators

- In VB.NET, there are two concatenation Operators to bind the operands:

Operator	Description	Example
&	It is an ampersand symbol that is used to bind two or more operand together.	Result = Wel & come, Result = Welcome

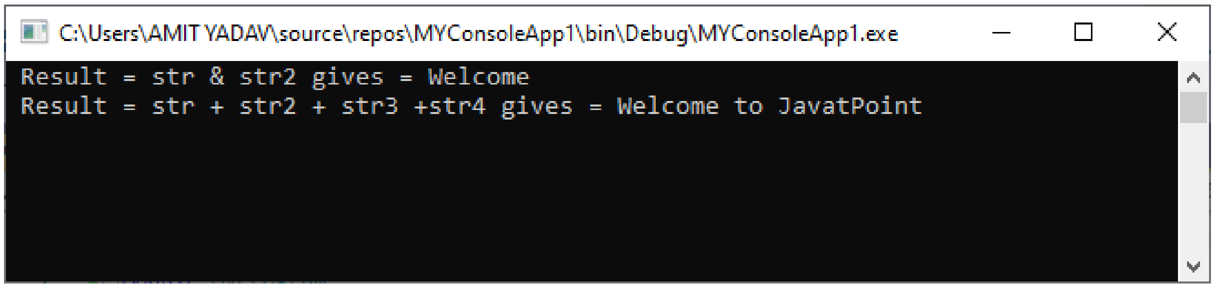
	Furthermore, a nonstring operand can also be concatenated with a string variable (but in that case, Option Strict is on).	
+	It is also used to add or concatenate two number or string.	Result = Wel + come, Result = Welcome

Example of Concatenation Operators in VB.NET.

MyProgram.vb

```
Module MyProgram
  Sub Main()
    Dim str As String = "Wel"
    Dim str2 As String = "come"
    Dim str3 As String = " "
    Dim str4 As String = "to JavatPoint"
    Dim result As String
    Dim result2 As String
    result = str & str2
    Console.WriteLine(" Result = str & str2 gives = {0}", result)
    result2 = str + str2 + str3 + str4
    Console.WriteLine(" Result = str + str2 + str3 +str4 gives = {0}", result2.ToString)
 Console.ReadLine()
  End Sub
End Module
```

OUTPUT



C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe

```
Result = str & str2 gives = Welcome
Result = str + str2 + str3 +str4 gives = Welcome to JavatPoint
```

Operator Precedence in VB.NET

- Operator precedence is used to determine the order in which different Operators in a complex expression are evaluated.
- There are distinct levels of precedence, and an Operator may belong to one of the levels.
- The Operators at a higher level of precedence are evaluated first. Operators of similar precedents are evaluated at either the left-to-right or the right-to-left level.

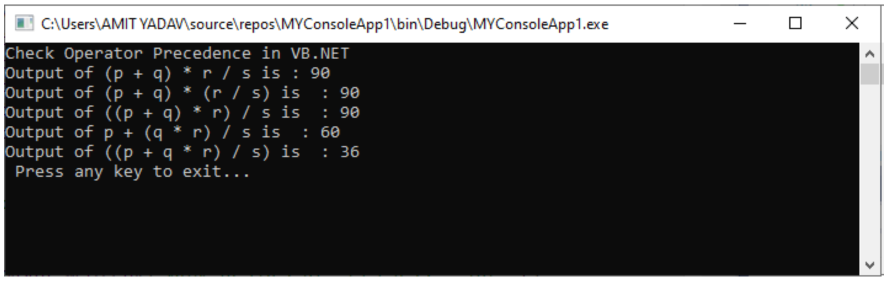
Operations	**Operators**	**Precedence**
Await		Highest
Exponential	**^**	
Unary identity and negation	**+, -**	
Multiplication and floating-point division	***, /**	
Integer division	\	
Modulus arithmetic	**Mod**	
Addition and Subtraction	**+, -**	
Arithmetic bit shift	**<<, >>**	
All comparison Operators	**=, <>, <, <=, >, >=, Is, IsNot, Like, TypeOf …is**	
Negation	**Not**	
Conjunction	**And, AndAlso**	
Inclusive disjunction	**Or, Else**	
Exclusive disjunction	**Xor**	Lowest

Example of Operator Precedence in VB.NET.

Operator_Precedence.vb

```
Module Operator_Precedence
    Sub Main()
        Dim p As Integer = 30
        Dim q As Integer = 15
        Dim r As Integer = 10
        Dim s As Integer = 5
        Dim result As Integer
         Console.WriteLine("Check Operator Precedence in VB.NET")
        result = (p + q) * r / s
        Console.WriteLine("Output of (p + q) * r / s is : {0}", result)
        result = (p + q) * (r / s)
        Console.WriteLine("Output of (p + q) * (r / s) is  : {0}", result)
        result = ((p + q) * r) / s
        Console.WriteLine("Output of ((p + q) * r) / s is  : {0}", result)
        result = p + (q * r) / s
        Console.WriteLine("Output of p + (q * r) / s is  : {0}", result)
        result = ((p + q * r) / s)
        Console.WriteLine("Output of ((p + q * r) / s) is  : {0}", result)
        Console.WriteLine(" Press any key to exit...")
        Console.ReadKey()
    End Sub
End Module
```

OUTPUT



```
Check Operator Precedence in VB.NET
Output of (p + q) * r / s is : 90
Output of (p + q) * (r / s) is  : 90
Output of ((p + q) * r) / s is  : 90
Output of p + (q * r) / s is  : 60
Output of ((p + q * r) / s) is  : 36
 Press any key to exit...
```

Question

1) What is .NET Framework?
2) Explain Net Framework Features & Architecture.
3) Write short note on CLR (common language runtime)
4) Write short note on CTS (Common Type System)
5) Write short note on BCL (Base Class Library)

6) Write short note on CLS (Common language Specification)
7) Write short note on Microsoft .NET Assemblies
8) Write short note on FCL (Framework Class Library)
9) Write short note on Solution Explorer
10) Write short note on Toolbox
11) Write short note on Properties Window
12) Write short note on Form Designer
13) Write short note on Output Window
14) Write short note on Object Browser.
15) Explain different Data Types,
16) What is Variable? Explain Scope Of Variable
17) Explain Type Casting
18) What is Constants? Explain with example.
19) Write short note on Arithmetic Operators
20) Write short note on Comparison Operators
21) Write short note on Logical and Bitwise Operators
22) Write short note on Bit Shift Operators
23) Write short note on Assignment Operators
24) Write short note on Concatenation Operators
25) Write short note on Operator Precedence in VB.NET