

VB.NET Operators

In **VB.NET** programming, the **Operator** is a symbol that is used to perform various operations on variables. **VB.NET** has different types of Operators that help in performing logical and mathematical operations on data values. The **Operator precedence** is used to determine the execution order of different Operators in the VB.NET programming language.

What is VB.NET Operator?

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
- Miscellaneous 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:

Arithmetic Operators in VB.NET

Operators	Description	Example
^	It is an exponentiation Operator that is used to raises one operand to the power of another operand.	$Y \wedge X$ (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.	$X \backslash Y$
Mod	It is a modulo (Modulus) Operator, which is used to divide two operands and returns only a remainder.	$X \text{ Mod } Y$

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

Arithmetic_Operator.vb

```
Imports System
Module Arithmetic_Operator
    Sub Main()
        'Declare a, b And c as integer Data Type()
        Dim a, b, c As Integer
        Dim d As Single
        a = 17
        b = 4
        ' Use of + Operator
```

```
c = a + b
Console.WriteLine(" Sum of a + b is {0}", c)
```

```
'Use of - Operator
c = a - b
Console.WriteLine(" Subtraction of a - b is {0}", c)
```

```
'Use of * Operator
c = a * b
Console.WriteLine(" Multiplication of a * b is {0}", c)
```

```
'Use of / Operator
d = a / b
Console.WriteLine(" Division of a / b is {0}", d)
```

```
'Use of \ Operator
c = a \ b
Console.WriteLine(" Similar to division Operator (return only integer value) of a -
b is {0}", c)
```

```
'Use of Mod Operator
c = a Mod b
Console.WriteLine(" Modulus of a Mod b is {0}", c)
```

```
'Use of ^ Operator
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

Now compile and execute the above program, by pressing the F5 button or Start button from the Visual Studio; then it shows the following result:

```
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	A >= B

	shows False.	
<=	This symbol represents less than equal to 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.	A <= B
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

Imports System

Module Comparison_Operator

Sub Main()

'declaration of Integer, Object and String Data Type variables

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")
```

```
'Use of > Operator
```

```
Console.WriteLine(" Output of x > y is {0}", x > y)
```

```
'Use of < Operator
```

```
Console.WriteLine(" Output of x < y is {0}", x < y)
```

```
'Use of = Operator
```

```
Console.WriteLine(" Output of x = y is {0}", x = y)
```

```
'Use of <> Operator
```

```
Console.WriteLine(" Output of x <> y is {0}", x <> y)
```

```
'Use of >= Operator
```

```
Console.WriteLine(" Output of x >= y is {0}", x >= y)
```

```
'Use of <= Operator
```

```
Console.WriteLine(" Output of x <= y is {0}", x <= y)
```

```
'Use of Is Operator
```

```
Result = obj Is obj2
```

```
Console.WriteLine(" Output of obj Is obj2 is {0}", Result)
```

```
'Use of Is Operator
```

```
Result = obj IsNot obj2
```

```
Console.WriteLine(" Output of obj IsNot obj2 is {0}", Result)
```

```
'Use of Like Operator
```

```
Result = str Like str2
```

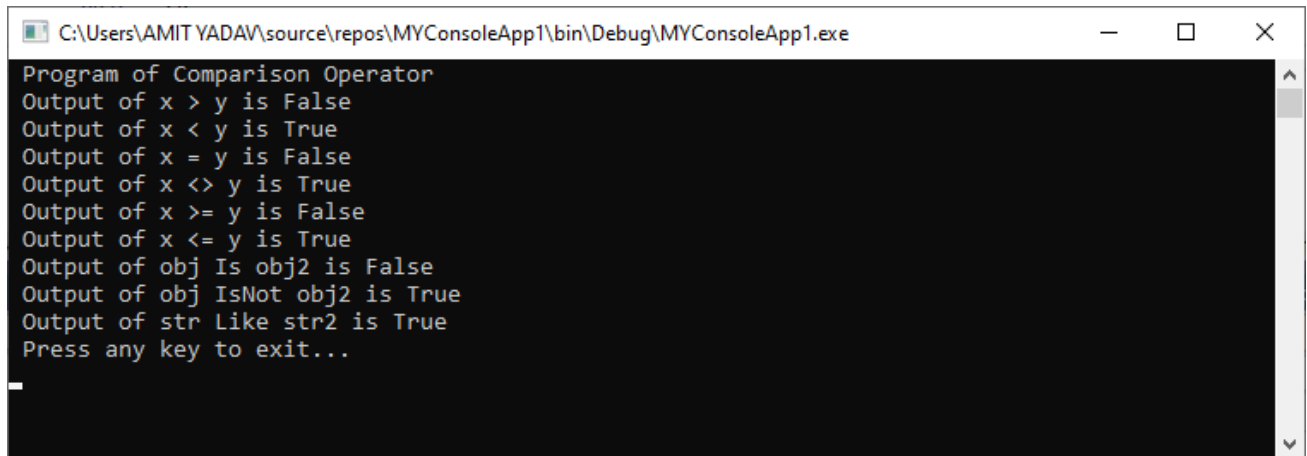
```
Console.WriteLine(" Output of str Like str2 is {0}", Result)
```

```
Console.WriteLine(" Press any key to exit...")
```

```
Console.ReadKey()
```

End Sub
End Module

Now compile and execute the above code by pressing the F5 button or Start button in Visual studio, it returns the following 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

```
Imports System
Module Logic_Bitwise
    Sub Main()
        Dim A As Boolean = True
        Dim B As Boolean = False
        Dim c, d As Integer
        c = 10
        d = 20

        'Use of And Operator
        If A And B Then
            Console.WriteLine(" Operands A And B are True")
        End If
    End Sub
End Module
```


End If

'Use of Or Operator

If A Or B Then

 Console.WriteLine(" Operands A Or B are True")

End If

'Use of Xor Operator

If A Xor B Then

 Console.WriteLine(" Operands A Xor B is True")

End If

'Use of And Operator

If c And d Then

 Console.WriteLine(" Operands c And d is True")

End If

'Use of Or Operator

If c Or d Then

 Console.WriteLine(" Operands c Or d is True")

End If

'Use of AndAlso Operator

If A AndAlso B Then

 Console.WriteLine(" Operand A AndAlso B is True")

End If

'Use of OrElse Operator

If A OrElse B Then

 Console.WriteLine(" Operand A OrElse B is True")

End If

'Use of Not Operator

If Not (A And B) Then

 Console.WriteLine(" Output of Not (A And B) is True")

End If

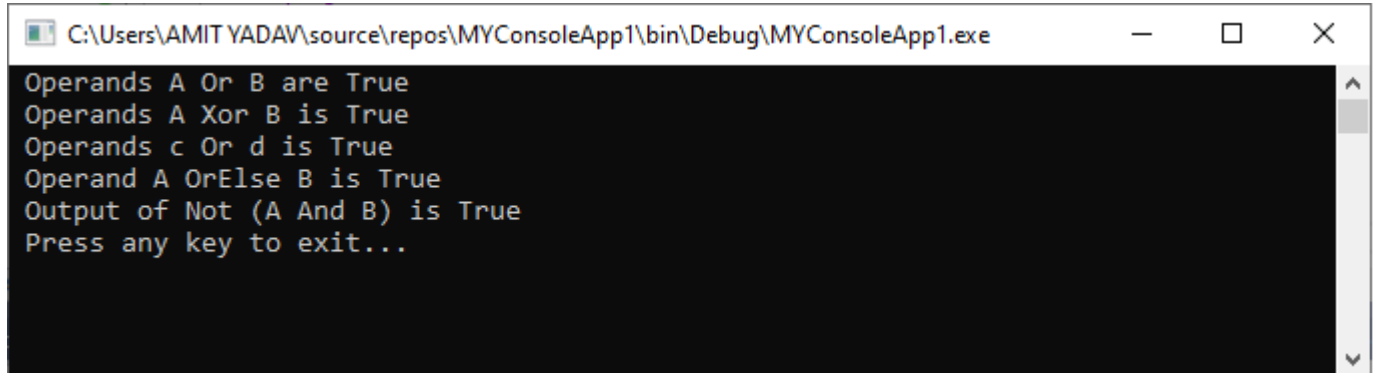
Console.WriteLine(" Press any key to exit?")

Console.ReadKey()

End Sub

End Module

Now compile and execute the above code by pressing the F5 button or Start button in Visual studio, it returns the following 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 Shift Operators are used to perform the bit shift operations on binary values either to the right or to the left.

Bit Shift operations in VB.NET

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

Imports System

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)
```

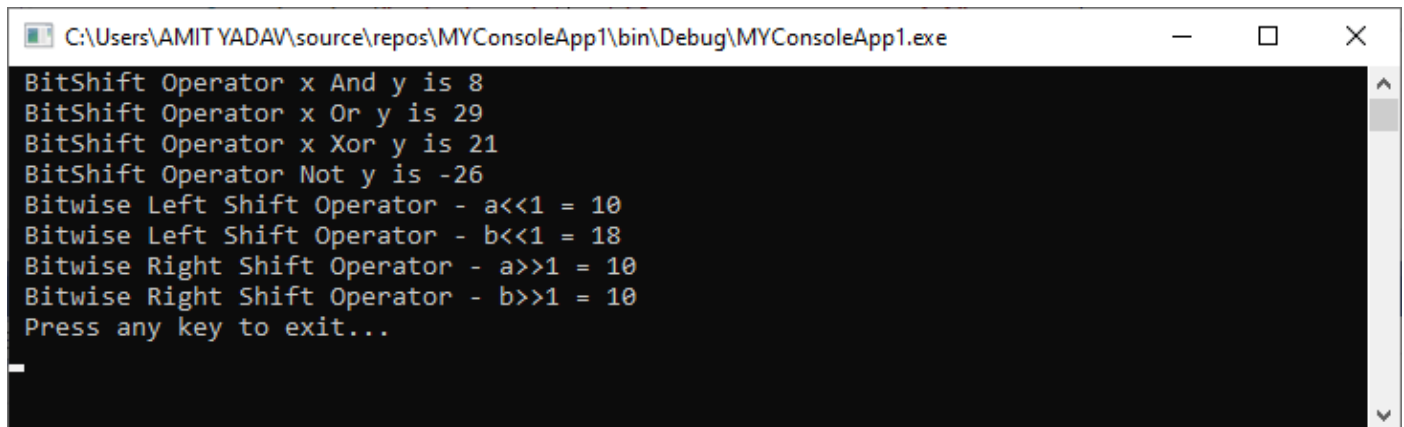
```
Console.WriteLine(" Press any key to exit...")
```

```
Console.ReadKey()
```

```
End Sub
```

```
End Module
```

Now compile and execute the above code by pressing the F5 button or Start button in Visual studio, it returns the following output:

A screenshot of a Windows console window titled "C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe". The console displays the following output:

```
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.

Assignment Operators 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 multiplies the right operand or value with 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 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).	$X \setminus = P$, which is same as $X = X \setminus P$
^=	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

Imports System

Module Assign_Operator

Sub Main()

'Declare variable and b As Integer

Dim A As Integer = 5

Dim B As Integer

Dim Str, name As String

name = "come"

Str = "Wel"

'Use of = Operator

B = A

Console.WriteLine(" Assign value A to B is {0}", B)

'Use of += Operator

B += A

Console.WriteLine(" Output of B += A is {0}", B)

'Use of -= Operator

B -= A

Console.WriteLine(" Output of B -= A is {0}", B)

'Use of *= Operator

B *= A

Console.WriteLine(" Output of B *= A is {0}", B)

'Use of /= Operator

B /= A

Console.WriteLine(" Output of B /= A is {0}", B)

'Use of = Operator

B \= A

Console.WriteLine(" Output of B \= A is {0}", B)

'Use of ^= Operator

B ^= A

```
Console.WriteLine(" Output of B ^= A is {0}", B)
```

```
'Use of &= Operator
```

```
Str &= name
```

```
Console.WriteLine(" Output of Str &= name is {0}", Str)
```

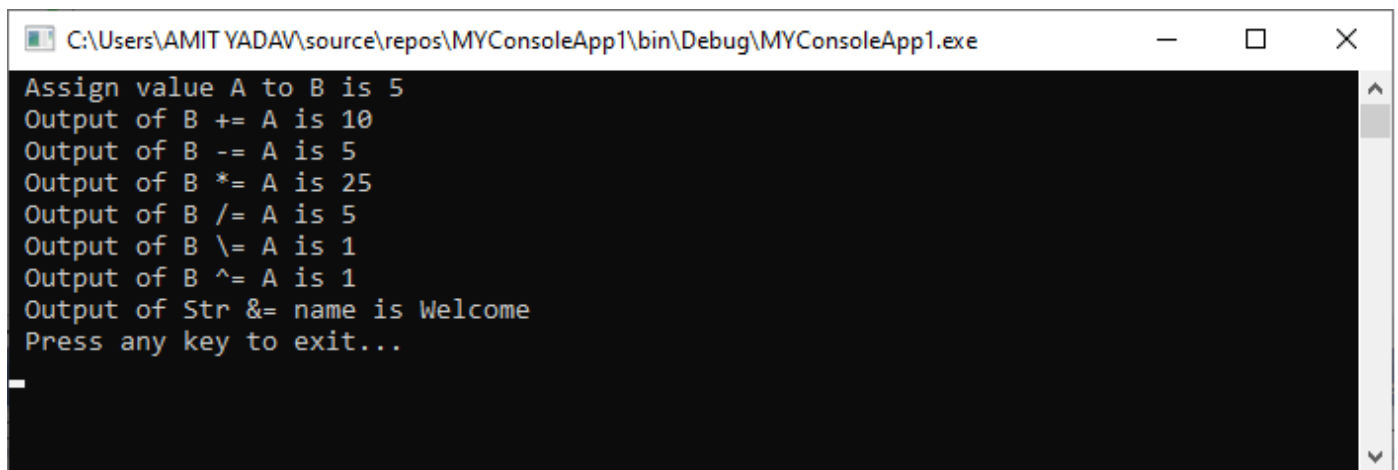
```
Console.WriteLine(" Press any key to exit...")
```

```
Console.ReadKey()
```

```
End Sub
```

```
End Module
```

Now compile and execute the above code by pressing the F5 button or Start button in Visual studio, it returns the following output:



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

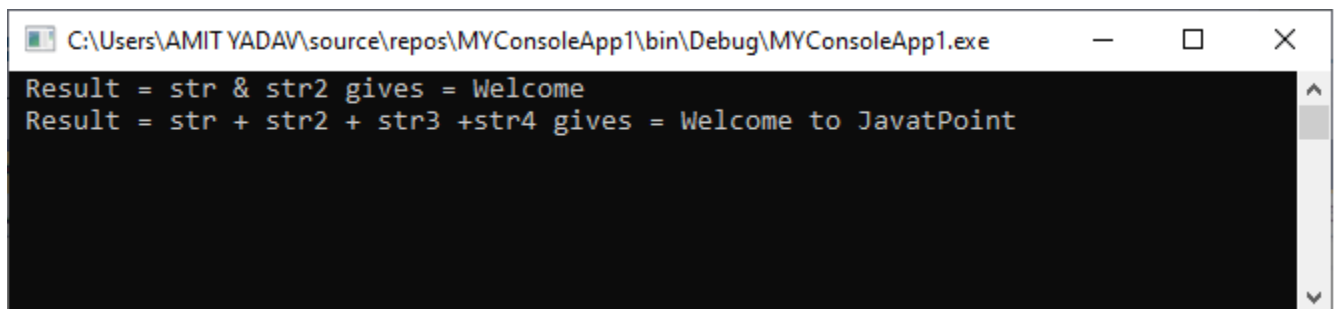
	string.	Wel + come, Result = Welcome
--	---------	---------------------------------------

Example of Concatenation Operators in VB.NET.

MyProgram.vb

```
Imports System
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
```

Now compile and execute the above code by pressing the F5 button or Start button in Visual studio, it returns the following output:



The screenshot shows a console window titled "C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe". The output of the program is displayed on two lines:

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


Miscellaneous Operators

There are some important Operator in VB.NET

Operator	Description	Example
Await	An Await Operator is used in an operand to suspend the execution of an asynchronous method or lambda expression until the awaited task completes.	<pre>Dim output as out = Await AsyncMethodThatReturnsResult() Await AsyncMethod()</pre>
AddressOf	The AddressOf Operator is used to provide a reference to the address of a procedure.	<pre>AddHandler Button2.Click, AddressOf Button2_Click</pre>
GetType	A GetType Operator is used to retrieve the type of the specified object. In addition, the retrieved object type provides various information such as methods, properties, and events.	<pre>MsgBox(GetType(String).ToString())</pre>
Function Expression	It defines the lambda expression, which declares the parameter and code. A Lambda expression is a function that is used to calculate and return value without defining the name.	<pre>Dim mul2 = Function(num As Integer) num * 4 Console.WriteLine(mul2(4))</pre>
If	The If Operator using short circuit evaluation to	<pre>Dim a = -4 Console.WriteLine(If (a >= 0,</pre>

	conditionally return a single object value from two defined object values. The If Operator can be used with two or three defined arguments.	"Positive", "Negative"))
--	---	--------------------------

Example of **Miscellaneous Operators** in VB.NET.

Misc_Operator.vb

Imports System

Module Misc_Operator

Sub Main()

' Initialize a variable

Dim a As Integer = 50

' GetType of the Defined Type

Console.WriteLine(GetType(Double).ToString())

Console.WriteLine(GetType(Integer).ToString())

Console.WriteLine(GetType(String).ToString())

Console.WriteLine(GetType(Single).ToString())

Console.WriteLine(GetType(Decimal).ToString())

'Use of Function()

Dim multiplywith10 = Function(sum As Integer) sum * 10

Console.WriteLine(multiplywith10(10))

Console.WriteLine(If(a >= 0, "Negative", "Positive"))

Console.WriteLine(" Press any key to exit...")

Console.ReadLine()

End Sub

End Module

Now compile and execute the above code by pressing the F5 button or Start button in Visual studio, it returns the following output:

```
C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe
System.Double
System.Int32
System.String
System.Single
System.Decimal
100
Negative
Press any key to exit...
```

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.

The Following table shows the operations, Operators and their precedence -

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

Imports System

Module Operator_Precedence

Sub Main()

'Declare and Initialize p, q, r, s variables

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")

'Check Operator Precedence

result = (p + q) * r / s ' 45 * 10 / 5

Console.WriteLine("Output of (p + q) * r / s is : {0}", result)

result = (p + q) * (r / s) ' (45) * (10/5)

Console.WriteLine("Output of (p + q) * (r / s) is : {0}", result)

result = ((p + q) * r) / s ' (45 * 10) / 5

Console.WriteLine("Output of ((p + q) * r) / s is : {0}", result)

result = p + (q * r) / s ' 30 + (150/5)

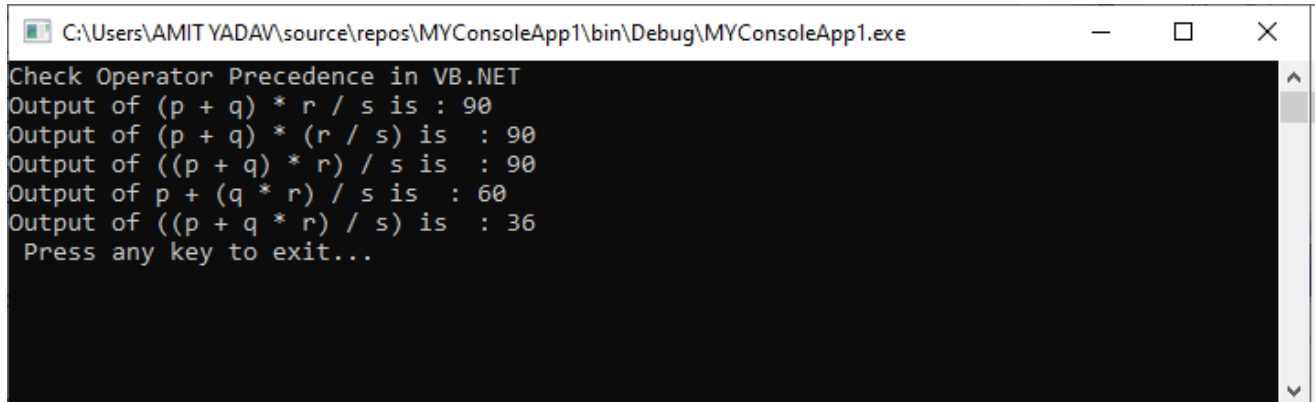
Console.WriteLine("Output of p + (q * r) / s is : {0}", result)

result = ((p + q * r) / s) ' ((30 + 150) / 5)

Console.WriteLine("Output of ((p + q * r) / s) is : {0}", result)

```
    Console.WriteLine(" Press any key to exit...")
    Console.ReadKey()
End Sub
End Module
```

Now compile and execute the above code by pressing the F5 button or Start button in Visual studio, it returns the following output:

A screenshot of a Windows console window titled "C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe". The window has a black background with white text. The text displays the output of a VB.NET program, showing five different arithmetic expressions and their results, followed by a prompt to press any key to exit. The output is as follows:

```
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...
```