**Operators and Expressions in Visual Basic**

An operator is a code element that performs an operation on one or more code elements that hold values. Value elements include variables, constants, literals.

An expression is a series of value elements combined with operators, which yields a new value. The operators act on the value elements by performing calculations, comparisons, or other operations.

**Types of Operators**

Visual Basic provides the following types of operators:

Arithmetic Operators

Comparison Operators

Logical (or Relational) Operators

Concatenation Operators

**The Arithmetic Operators**

Following arithmetic operators are supported by VB.

Assume variable A holds 5 and variable B holds 10, then −



Operator Description Example

+ Adds the two operands A + B will give 15

- Subtracts the second operand from the first A - B will give -5

\* Multiplies both the operands A \* B will give 50

/ Divides the numerator by the denominator B / A will give 2

% Modulus operator and the remainder after an integer division B % A will give 0

^ Exponentiation operator B ^ A will give 100000

Integer division is carried out using the \ Operator (Visual Basic). Integer division returns the quotient, that is, the integer that represents the number of times the divisor can divide into the dividend without consideration of any remainder.

Dim k As Integer

k = 23 \ 5

' The preceding statement sets k to 4.

**The Comparison Operators**

Assume variable A holds 10 and variable B holds 20, then

Operator Description Example

= Checks if the value of the two operands are equal or not.

If yes, then the condition is true. (A = B) is False.

<> Checks if the value of the two operands are equal or not.

If the values are not equal, then the condition is true. (A <> B) is True.

> Checks if the value of the left operand is greater than the

value of the right operand. If yes, then the condition is true. (A > B) is False.

< Checks if the value of the left operand is less than the value

of the right operand. If yes, then the condition is true. (A < B) is True.

>= Checks if the value of the left operand is greater than or equal to

the value of the right operand. If yes, then the condition is true. (A >= B) is False.

<= Checks if the value of the left operand is less than or equal to

the value of the right operand. If yes, then the condition is true. (A <= B) is True.

**The Logical Operators**

Following logical operators are supported by VB. Assume variable A holds 10 and variable B holds 0, then

Operator Description Example

AND Called Logical AND operator. If both the conditions are True,

then the Expression is true. a<>0 AND b<>0 is False.

OR Called Logical OR Operator. If any of the two conditions are

True, then the condition is true. a<>0 OR b<>0 is true.

NOT Called Logical NOT Operator. Used to reverse the logical state of its

operand. If a condition is true, then Logical NOT operator

will make false. NOT(a<>0 OR b<>0) is false.

**The Concatenation Operators**

Following Concatenation operators are supported by VB.

Assume variable A holds 5 and variable B holds 10 then

Operator Description Example

+ Adds two Values as Variable. Values are Numeric A + B will give 15

& Concatenates two Values A & B will give 510



Assume variable A = "Microsoft" and variable B = "VBScript", then

Operator Description Example

+ Concatenates two Values A + B will give MicrosoftVBScript

& Concatenates two Values A & B will give MicrosoftVBScript

Concatenation Operators can be used for both numbers and strings. The output depends on the context, if the variables hold numeric value or string value.

**Making decisions with if statement**

The ability to make decisions is known as branching. We can repeat a section of code a specified number of times or while a certain condition applies. This process is known as looping.

**Making Decisions**

**The If Statement**

The simplest way to make a decision in a Visual Basic .NET program is to use the If….Then statement.

Example

‘define a value for n…

Dim n As Integer

n=27

‘here is where we take a decision,

‘and tell the user what happened…

If n=27 Then

MessageBox.Show(“’n ’ is, indeed, 27!”)



End If

**The Else Statement**

What do we do if we want to run one piece of code if the condition is true and another piece if the condition is false? We use the Else Statement.

‘define a value for n…

Dim n As Integer

‘here is where we take a decision,

‘and tell the user what happened…

If n=1000 Then

MessageBox.Show(“’n ’ is, indeed, 1000!”)

Else

MessageBox.Show(“’n’ is not 1000!”)

End If

**Multiple Alternatives with ElseIf**

If we want to test for more than one condition, we need to make use of the ElseIf statement.

‘define a value for n…

Dim n As Integer

n=27

‘here is where we take a decision,

‘and tell the user what happened…

If n=1000 Then

MessageBox.Show(“’n ’ is, indeed, 1000!”)

ElseIf n=27 Then

MessageBox.Show(“’n’ is 27!”)

Else

MessageBox.Show(“’n’ is neither 1000 nor 27!”)

End If

**Nested If Statements**

It is possible to nest an If statement inside another. For Example

If n=3 Then

MessageBox.Show(“n=3”)

If x=6 Then

MessageBox.Show(“x=6”)

End If

End If