Visual Basic Language Reference

Operator Precedence in Visual Basic

When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called *operator precedence*.

When expressions contain operators from more than one category, they are evaluated according to the following rules:

- The arithmetic and concatenation operators have the order of precedence described below, and all have higher precedence than the comparison, logical, and bitwise operators.
- Comparison operators have equal precedence, and all have higher precedence than the logical and bitwise operators, but lower precedence than the arithmetic and concatenation operators.
- Logical and bitwise operators have the order of precedence described below, and all have lower precedence than the arithmetic, concatenation, and comparison operators.
- Operators with equal precedence are evaluated left to right in the order in which they appear in the expression.

Operators are evaluated in the following order of precedence:

Arithmetic and Concatenation Operators

Exponentiation (**^**)

Unary negation (**–**)

Multiplication and division (*****, **/**)

Integer division (\)

Modulus arithmetic (**Mod**)

Addition and subtraction (**+**, **–**), string concatenation (**+**)

String concatenation (**&**)

Arithmetic bit shift (**<<**, **>>**)

Comparison Operators

All comparison operators (**=**, **<>**, **<**, **<=**, **>**, **>=**, **Like**, **Is**, **TypeOf...Is**)

Logical and Bitwise Operators

Negation (**Not**)

Conjunction (**And**, **AndAlso**)

Disjunction (**Or**, **OrElse**, **Xor**)

The string concatenation operator (**&**) is not an arithmetic operator, but in precedence it is grouped with the arithmetic operators.

The **Is** operator is an object reference comparison operator. It does not compare objects or their values; it checks only to determine if two object variables refer to the same object.

When operators of equal precedence occur together in an expression, for example multiplication and division, each operation is evaluated as it occurs from left to right. Parentheses can be used to override the order of precedence and to force some parts of an expression to be evaluated before others. Operations within parentheses are always performed before those outside. Within parentheses, however, operator precedence is maintained. The following example illustrates this:

```
Dim A, B, C, D, E, F, G As Double
A = 3.0
B = 6.0
C = 4.0
D = 2.0
E = 1.0
F = A + B - C / D * E
' The previous line sets F to 7.0. Because of natural operator
' precedence, it is exactly equivalent to the following line:
F = (A + B) - ((C / D) * E)
' The following line overrides the natural operator precedence:
G = A + (B - C) / (D * E)
' The previous line sets G to 4.0.
```

See Also

= Operator | Is Operator | Like Operator | Operators Listed by Functionality | Operators