

Arithmetic Operations

Arithmetic operations, or calculations, are important parts of programming. Almost every program includes at least a few arithmetic operations. The last chapter did several calculations.

Assigning Values to a Variable

A variable can be assigned a value with the assignment statement. The format is:

`<variable> = <newvalue>`

Examples:

The new value is a constant: `X = 5`

The new value is another variable: `X = Y`

The new value is an expression: `X = Y/3`

It can even use its current value in the expression: `X = X + 1`

The last statement assigns a new value to X that is one more than its current value!

Note: The statement `X = 5` is an assignment statement, this is *not* the same as **If** `X = 5`, which is a Boolean expression to *compare* the two values.

A sequence of statements is shown below. The last value shown in the column is the current value, the current value is used each time a statement is executed. Keeping track of variables this way is called *tracing*:

Statement:	X
<code>X = 5</code>	5
<code>X = X + 1</code>	6
<code>X = X + 1</code>	7

It is important to remember that the variable that is assigned a value is on the left; the equation, or new value, is on the right. When the computer executes the statement `X = Y`, it looks up the current value of `Y` and stores that value as the new value of `X`:

Statement:	X	Y
<code>X = 5</code>	5	0
<code>Y = 12</code>	5	12
<code>X = Y</code>	12	12

The statement `Y = X` is *not* the same:

	X	Y
<code>X = 5</code>	5	0
<code>Y = 12</code>	5	12
<code>Y = X</code>	5	5

You can check the result by adding the code to a procedure, and then add `Me.Text=X` at the end.

Arithmetic Operators

Visual Basic has the following operators:

Operator	Purpose:	Example:	X
+	Addition	X= 5+3	8
-	Minus	X= 8-2	6
	(using most recent value of x)	X= X-1	5
*	Multiplication	X= 6*2	12
\	Integer division	X= 17\5	3
/	Division	X= 17/5	3.5
		X= 15/5	3
Mod	Remainder	X= 17 MOD 5	2
^	Raise to a power: 4^2 is 4 ² , 2^3 is 2 ³	X= 4^2	16
		X= 2^3	8

An actual program would not use a statement such as X=5+3, it would save time to simply use X=8. An actual program would be more likely to use variables: X=Y+Z, for example.

Note that there is a *times*, or *multiplication*, operator: *. In algebra, variables are always a single letter, XY in algebra means X times Y. In programming, variables can be several letters, and we could not be sure whether XY meant X times Y or a variable called XY.

Mod

The Mod (modulus) operator is used to find the remainder. Before children learn about decimal numbers, they may give the answer to division problems as:

“17 divided by 5 is 3 with a **remainder of 2**”

$$\begin{array}{r}
 3 \\
 5 \overline{)17} \\
 \underline{15} \\
 2
 \end{array}
 \leftarrow 17 \bmod 5$$

Note that 17/5 results in 3.5, 17\5 results in 3, while 17 mod 5 results in 2.

Study tip: After studying the examples, try evaluating the expressions without looking at the answers, then do the exercises. You can check your answers using the immediate window.

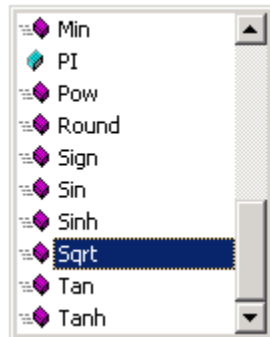
Exercise: Show the value of z after executing each statement, check your answers by executing the statements.

Statement:	X	Y	Z
X = 8	8		
Y = 5		5	
Z = X+Y			
Z = X^2			
Z = X*2			
Z = X/2			
Z = X MOD 2			
Z = X-Y			
Z = Y MOD 3			

Functions

In addition to the operators shown above, Visual Basic has functions to find square root; trigonometric functions such as sin and cosine, and many others. You can see a list of the functions and display them by typing line in any procedure

```
Me.Text = System.Math.
```



Finish typing the line as

```
Me.Text = System.Math.Sqrt(16)
```

When you run the program, this procedure will display 4 in the text. Functions receive values and return a result. In the example above Sqrt receives 16 and returns 4.

Order of Operations

A statement can perform more than one arithmetic operation. If there is more than one arithmetic operation in a statement, they are executed according to the level of the operators: Parenthesis take precedence over all other operators; multiplication and division take precedence over mod, mod takes precedence over addition and subtraction.

Evaluate an expression in the following order:

1. Anything inside parenthesis, including functions, is done first: working from the inner-most parenthesis out;
2. Raise to a power: ^ is performed from left to right;
3. Multiplication and division (*, /, \) are performed from left to right;
4. Mod is performed from left to right;
5. Addition and subtraction (+ and -), are performed from left to right.

Examples: The order of evaluation is shown by underlining the next operation, then replacing it with the result (in bold) on the next line.

$$\begin{array}{r} \text{A.} \quad 3 + 5 * 2 \\ \quad \quad 3 + \underline{10} \\ \quad \quad \underline{13} \end{array}$$

** takes precedence over +*

*5*2=10: the 10 replaces 5*2 on this line*

3+10=13: the 13 replaces 3+10 on this line: Done!

$$\begin{array}{r} \text{B.} \quad (\underline{3 + 5}) * 2 \\ \quad \quad \underline{8} * 2 \\ \quad \quad \underline{16} \end{array}$$

parenthesis first

Notice the difference between the first and second problems!

- C.
$$\begin{array}{r} 3 * (5 - 3 * 8 + 4) \\ 3 * (5 - \mathbf{24} + 4) \\ 3 * (\mathbf{-19} + 4) \\ 3 * \mathbf{-15} \\ \hline \mathbf{-45} \end{array}$$
 *parenthesis first, within parenthesis, * before + or -
if same level (+ and -), work from left to right*
- D.
$$\begin{array}{r} 3 * (5 - 3) * (8 + 4) \\ 3 * \mathbf{2} * (8 + 4) \\ 3 * 2 * \mathbf{12} \\ \hline \mathbf{6} * \mathbf{12} \\ \hline \mathbf{72} \end{array}$$
 *parenthesis first, left to right
parenthesis first
if same level(*, /), work from left to right*
- E.
$$\begin{array}{r} -3 * (6 * (3 - 8) + 10) \\ -3 * (6 * \mathbf{-5} + 10) \\ -3 * (\mathbf{-30} + 10) \\ -3 * \mathbf{-20} \\ \hline \mathbf{60} \end{array}$$
 *inner-most parenthesis first
if same level (+ and -), work from left to right
parenthesis first
the minus in front of 3: -3, is a sign, not an operator*

Variables and Functions

If there are variables and functions in an expression replace the variables with their current values (shown below); evaluate any functions; and then evaluate using the rules for the order of operation given above:

Examples:

X	Y	Z
4	5	6

- A.
$$\begin{array}{r} Z - X \text{ Mod } Y \\ 6 - 4 \text{ Mod } 5 \\ 6 - \mathbf{4} \\ \hline \mathbf{2} \end{array}$$
 *replace variables with current values
mod takes precedence over -, 4/5=0 with 4 remainder!
4 mod 5=4, the 4 replaces 4 mod 5 on this line
6-4=2, the 2 replaces 6-4 on this line: Done!*
- B.
$$\begin{array}{r} Z - \text{System.Math.Sqrt} (X) \\ 6 - \text{System.Math.Sqrt} (4) \\ 6 - \mathbf{2} \\ \hline \mathbf{4} \end{array}$$
 *replace variables with current values
The square root of 4 is 2
The 2 replaces Sqrt(4) on this line
6-2=4, the 4 replaces 6-2 on this line: Done!*
- C.
$$\begin{array}{r} z - \text{System.Math.Sqrt} (X+Y) \\ 6 - \text{System.Math.Sqrt} (4+5) \\ 6 - \text{System.Math.Sqrt} (\mathbf{9}) \\ 6 - \mathbf{3} \\ \hline \mathbf{3} \end{array}$$
 *replace variables with current values
Add 4+5 before evaluating the function
The square root of 9 is 3
The 3 replaces sqrt(9) on this line
6-3=3, the 3 replaces 6-3 on this line: Done!*

$$\begin{array}{r}
 \text{D. } (X + Y + Z) / 3 \\
 (4 + 5 + 6) / 3 \\
 (\quad \mathbf{9} \quad + 6) / 3 \\
 \hline
 \mathbf{15} / 3 \\
 \hline
 \mathbf{5}
 \end{array}$$

replace variables with current values

parenthesis first: + from left to right

parenthesis first

Notice that this is the average of the 3 numbers:

What is the result if the parenthesis are left out?

Algebra

Many times a programmer is given an algebraic formula, and must write an assignment statement that will correctly calculate the answer. After writing a statement, check it by hand, using sample values and the method shown above to verify that the equation is correct. Use the following guidelines to convert algebraic expressions to programming statements:

Algebra Program Explanation:

xy	$X*Y$	<i>Algebra uses single letters for variables, and can omit the *, programming languages always require an operator.</i>
$x \cdot y$	$X*Y$	<i>Algebra sometimes uses a dot for multiplication.</i>
x^2	X^2	<i>Programming languages use only one line for equations: they can not write anything above or below the line.</i>
x^z	X^Z	<i>The ^ operator is used for any power</i>
\sqrt{x}	$\text{Sqrt}(X)$	<i>The System.Math.Sqrt function returns the square root</i>
$\frac{a+b}{c}$	$(A+B)/C$	<i>Compound numerators and divisors must be enclosed in parenthesis: Notice the difference between this example and the one below:</i>
$a + \frac{b}{c}$	$A+B/C$	<i>Notice the difference between this example and the one above.</i>

Remember: Algebra uses single letters for variables: programmers should use good variable names wherever possible!

The "Grade" program calculates the average of the midterm and final with the equation:

$$\text{Average} = (\text{Midterm} + \text{Final}) / 2$$

Assume that the midterm grade is 80 and the final grade is 90. Average will be calculated as shown below:

$$\begin{array}{r}
 (80 + 90) / 2 \\
 \hline
 \mathbf{170} / 2 \\
 \hline
 \mathbf{85}
 \end{array}$$

parenthesis first

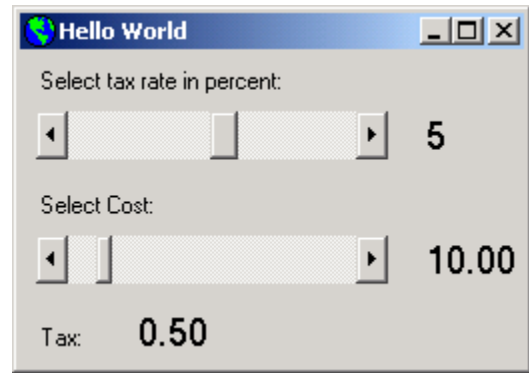
What will the average be if you leave off the parenthesis?

Experiment: Write a program to calculate grade. Use scroll bars for midterm and final. An extra challenge is to calculate the letter grade as well as the average.

Taxes

Start a new windows application, name it Taxes. A scroll bar can only select integers, but this program divides the amount on the scroll bar by 100 to allow the user to select the cost of an item to 2 decimal places. Another scroll bar selects the sales tax percent to one decimal place.

Build the form as shown in the illustration according to the table below.



Object	Property	Value
A lblRateHdr	Text	Select tax rate in percent
▣ scrRate		<i>no properties changed</i>
A lblRate	Text	0.0 'this is the initial value
	AutoSize	True
	Font	Select a large size
<i>Copy lblRate and paste to create lblCost and lblTax with the same properties.</i>		
▣ scrCost	Max	10000 'this will let us select 100.00
	LargeChange	1
A lblCostHdr	Text	Select cost
A lblTaxHdr	Text	Tax:

Write the code for scrRate and then modify the first line to handle scrCost also.

```
Private Sub scrRate_Scroll(ByVal sender As System.Object, _
    ByVal e As System.Windows.Forms.ScrollEventArgs) _
    Handles scrRate.Scroll, scrCost.Scroll
    Dim Tax, Rate, Cost As Double
    Rate = Me.scrRate.Value / 10
    Cost = Me.scrCost.Value / 100
    Tax = Cost * Rate / 100
    Me.lblRate.Text = Rate
    Me.lblCost.Text = Format(Cost, "0.00")
    Me.lblTax.Text = Format(Tax, "0.00")
End Sub 'scrRate_Scroll
```

Note: The format function takes 2 arguments, the value to be formatted and a string that shows the format. The first zero in the string "0.00" forces at least 1 digit before the zero, but 5 more may appear. The 2 zeroes at the end specify an exact number. If we didn't use the format function, \$1.50 would display as 1.5. The format can also contain a \$ if you would like to include it: `Me.lblTax.Text = Format(Tax, "$0.00")`

Experiment: Double click the word **Format** in the code, then press **F1** to show context sensitive help. The example in help shows how you can print the day of the week, the time and the current date.