Expressions and operators

Mathematical operators and expressions

The five basic binary mathematical operators are

Operator	Operation	Example
+	Addition	a = b + c
-	Subtraction	a = b – c
*	Multiplication	a = b * c
/	Division	a = b / c
%	Remainder	a = b%c = Remainder of (b / c)

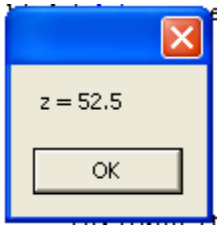
The operators "+" and "-" also have unary versions that change the sign of a variable. Mathematical expressions are constructed using these operators and the assignment operator "=".

As in the last section start a new application, add a button to it, double click on the button to get the method that is run when the button Click event is raised. Insert the following example of a mathematical expression:

```
double x = 2.5;
double y = 20;
double z = x*y+x;

MessageBox.Show("z = "+z.ToString());
```

You should get the following after running the application and clicking the button:



It is obvious in this example that first "x" was multiplied by "y" and then the result was added to "x" to get "z." This brings up the issue of how the compiler selects the order of operations. Why did the compiler not calculate "y+x" first and then multiply it by "x." This is because operators have an order of precedence which the compiler used to figure out which operations are done first. For mathematical expressions the order of precedence are:

Unary + -, then * / %, and then + -

Parenthesis are used to instruct the compiler on the order of operations. The compiler first calculates the contents of an "()" as one mathematical expression. Therefore if one replaces "x*y+x" in the above example by "x*(y+x)" the result of running the application would be:

Note, in this case the compiler first calculated "(y+x)" to get 22.5 and then multiplied it by "x."

There are several increment and decrement operators that can be used to increment by one unit the variable. These are:

Example	Meaning
A = ++B	First increment B by one and then assign it to A
A = - -B	First subtract one from B and then assign it to A
A = B++	First assign B to A and then increment B by one
A = B- -	First assign B to A and then subtract one from B

Operator precedence is as follows where operators of equal precedence are evaluated from left to right.

Precedence order	Operators
Highest	++ (prefix), - - (prefix), unary +, unary -
	*, /, %
	+, -
	=, *=, /=, %=, +=, -=
Lowest	++ (suffix), -- (suffix)

Mathematical assignment operator

There are several assignment operators that can be used

Name	Example	Meaning
=	A = B	
+=	A += B	A = A+B
-=	A -= B	A = A-B
*=	A *= B	A = A*B
/=	A /= B	A = A/B

%=	A %=B	A = Remainder(A/B)

Boolean expressions and operators

Boolean expressions are expressions that evaluate to “true” or to “false.” The Boolean comparison operators in C# can be used with a variety of data types and variables and given in the following table:

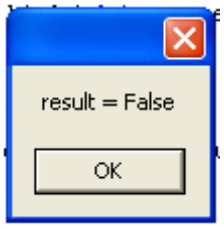
Operator	Name	Example
= =	Equal	A = = B
!=	Not equal	A !=B
<	Less than	A < B
>	Grater than	A > B
<=	Less than or equal	A <= B
>=	Grater than or equal	A >= B

In each case the example expression results in a Boolean value of “true” if the expression is true and “false” if the expression is false. For example, try the following code in your application:

```
bool result = 3<2;

MessageBox.Show("result = "+result.ToString());
```

Once you run your application, you should get:



There are a number of Boolean operators that are specifically for use with Boolean variables. These are given as:

Operator	Name	Example
!	NOT	!A
^	XOR (exclusive or)	A^B
&&	AND	A&&B
||	OR	A||B

The following rules exist for the examples shown in the table. The logical AND provides a true value only when both A and B are true. The logical OR provides a true value when either A or B is true. The XOR provides a value of true when either A or B are true, but

not when both are true. The NOT simply changes the value of A from true to false, or from false to true depending on if A is true or false.

The "&&" and "||" have two additional forms "&" and "|" which are similar but less safe when used with reference types since for these cases they check the address not the actual value.

Boolean assignment operators

As the arithmetic operators, the Boolean operators have several assignment operators

Operator	Example	Meaning
=	A = B	
&=	A &= B	A = A & B
|=	A |= B	A = A | B
^=	A ^= B	A = A ^ B

As an example enter the following code into your application:

```
bool A = false;
A &= true;

MessageBox.Show("A = "+A.ToString());
```

The result of running your application should be:

Boolean operator precedence is given in the following table:

Precedence order	Operator
Highest precedence	!
	<, >, <=, >=
	= =, !=
	&
	^
	|
	&&
	||
Lowest precedence	&=, ^=, |=

As in arithmetic expressions, it is best to use parenthesis to control the order of operation.

Math functions

There are a number of mathematical functions and constants that can be used in developing mathematical expressions. These mathematical functions can be invoked by using the "Math" namespace. For example, insert the following code in your application to get the value of $\cos(\pi)$:

```
double a = Math.Cos(Math.PI);

MessageBox.Show("a = "+a.ToString());
```

The result of running your application should be

The existing mathematical functions and constants are:

Function	Return type	Meaning
Abs(double x)	double	|x|
Acos(double x)	double	arccos(x)
Asin(double x)	double	arcsin(x)
Atan(double x)	double	arctan(x)
Atan2(double x, double y)	double	Angle that tangent is quotient of two numbers
BigMul(int m, int n)	long	Full product of two 32-bit numbers
Ceiling(double x)	double	Smallest hole number >= x
Cos(double x)	double	cos(x), x in radians
Cosh(double x)	double	cosh(x)
DivRem(int m, int n, out int result)	int	Quotient of two integers m and n
E	double	Natural logarithm base e
Equals(object a, object b)	bool	True if a and b are equal
Exp(double x)	double	Exponent of x
Floor(double x)	double	Largest whole number <= x
IEEERemainder(double x, double y)	double	Remainder of x/y
Log(double x, double base)	double	Logarithm of x in given base
Log(double x)	double	Natural logarithm of x (base e)
Log10(double x)	double	Log base 10 or x

Max(x,y), x and y are any data types	Type of x,y	Larger of x and y
Min(x,y), x and y are any data types	Type of x,y	Smaller of x and y
PI	double	π
Pow(double x, double y)	double	x to the power y
ReferenceEquals(object x, object y)	bool	Checks to see if x and y reference same object
Round(double x, int decimals)	double	Nearest number to x with precision decimals (many variations)
Sign(double x)	int	Returns an integer indicating sign of x
Sin(double x)	double	sin(x)
Sinh(double x)	double	sinh(x)
Sqrt(double x)	double	Square root of x
Tan(double x)	double	tan(x)
Tanh(double x)	double	tanh(x)

Some of these have variations not included in the table. Visual Studio can be used to find all variations.