Operator(s)	Operation(s)	Order of evaluation (precedence)
`( )`	Parentheses	Evaluated first. If the parentheses are *nested*, such as in the expression `a * ( b + c / d + e ) )`, the expression in the *innermost* pair is evaluated first. [*Caution:* If you have an expression such as `(a + b) * (c - d)` in which two sets of parentheses are not nested, but appear "on the same level," the C++ Standard does *not* specify the order in which these parenthesized subexpressions will be evaluated.]

`*`
`/`
`%`
	Multiplication
	Division
	Modulus
		Evaluated second. If there are several, they're evaluated left to right.

`+` `-`	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

Fig. 2.10 | Precedence of arithmetic operators.

Sample Algebraic and C++ Expressions

Now consider several expressions in light of the rules of operator precedence. Each example lists an algebraic expression and its C++ equivalent. The following is an example of an arithmetic mean (average) of five terms:

Algebra: $m = \frac{a + b + c + d + e}{5}$

C++: `m = ( a + b + c + d + e ) / 5;`

The parentheses are required because division has *higher* precedence than addition. The *entire* quantity `( a + b + c + d + e )` is to be divided by `5`. If the parentheses are erroneously omitted, we obtain `a + b + c + d + e / 5`, which evaluates incorrectly as

$$a + b + c + d + \frac{e}{5}$$

The following is an example of the equation of a straight line:

Algebra: $y = mx + b$

C++: `y = m * x + b;`

No parentheses are required. The multiplication is applied first because multiplication has a *higher* precedence than addition.

The following example contains modulus (`%`), multiplication, division, addition, subtraction and assignment operations:

Algebra: $z = pr\%q + w/x - y$

C++:

```
z  =  p  *  r  %  q  +  w  /  x  -  y;
   6     1     2     4     3     5
```