thinking *for* is a function. It also prevents you from naming a function *for*.

**Tip**

Common C++ style is to place a space between *for* and the following parentheses and to omit space between a function name and the following parentheses:

```c++
for (int i = 6; i < 10; i++)
    smart_function(i);
```

Other control statements, such as *if* and *while*, are treated similarly to *for*. This serves to reinforce visually the distinction between a control statement and a function call. Also, common practice is to indent the body of a *for* statement to make it stand out visually.

### Expressions and Statements

A *for* control section uses three expressions. Within its self-imposed limits of syntax, C++ is a very expressive language. Any value or any valid combination of values and operators constitute an expression. For example, 10 is an expression with the value 10 (no surprise), and 28 * 20 is an expression with the value 560. In C++, every expression has a value. Often the value is obvious. For example, the expression

```
22 + 27
```

is formed from two values and the addition operator, and it has the value 49. Sometimes the value is less obvious. For example,

```
x = 20
```

is an expression because it's formed from two values and the assignment operator. C++ defines the value of an assignment expression to be the value of the member on the left, so the expression has the value 20. The fact that assignment expressions have values permits statements such as the following: