each other. The statement following the control section is called the body of the loop, and it is executed as long as the test expression remains true:

```cpp
for (initialization; test-expression; update-expression)
    body
```

C++ syntax counts a complete for statement as a single statement, even though it can incorporate one or more statements in the body portion. (Having more than one statement requires using a compound statement, or block, as discussed later in this chapter.)

The loop performs initialization just once. Typically, programs use this expression to set a variable to a starting value and then use the variable to count loop cycles.

The test-expression determines whether the loop body gets executed. Typically, this expression is a relational expression, that is, one that compares two values. Our example compares the value of i to 5, checking to see if i is less than 5. If the comparison is true, the program executes the loop body. Actually, C++ doesn't limit test-expression to true-false comparisons. You can use any expression, and C++ will typecast it to type bool. Thus, an expression with a value of 0 is converted to the bool value false, and the loop terminates. If the expression evaluates to nonzero, it is typecast to the bool value true, and the loop continues. Listing 5.2 demonstrates this by using the expression i as the test condition. (In the update section, i-- is similar to i++ except that it decreases the value of i by 1 each time it's used.)

**Listing 5.2 num_test.cpp**

```cpp
// num_test.cpp -- use numeric test in for loop
#include <iostream>
using namespace std;
int main()
{
    cout << "Enter the starting countdown value: ";
    int limit;
    cin >> limit;
    int i;
    for (i = limit; i; i--)
        // quits when i is 0
        cout << "i = " << i << "\n";
```