For example, validating input that has been read and reading lists of data terminated by a sentinel value are good applications of the `while` loop.

- **The `do-while` loop.** The `do-while` loop is also a conditional loop. Unlike the `while` loop, however, `do-while` is a posttest loop. It is ideal in situations where you always want the loop to iterate at least once. The `do-while` loop is a good choice for repeating a menu.
- **The `for` loop.** The `for` loop is a pretest loop that has built-in expressions for initializing, testing, and updating. These expressions make it very convenient to use a counter variable to control the number of iterations that the loop performs. The initialization expression can initialize the counter variable to a starting value, the test expression can test the counter variable to determine whether it holds the maximum value, and the update expression can increment the counter variable. The `for` loop is ideal in situations where the exact number of iterations is known.

5.10 Nested Loops

CONCEPT: A loop that is inside another loop is called a *nested loop*.

A nested loop is a loop that appears inside another loop. A clock is a good example of something that works like a nested loop. The second hand, minute hand, and hour hand all spin around the face of the clock. Each time the hour hand increments, the minute hand increments 60 times. Each time the minute hand increments, the second hand increments 60 times.

Here is a program segment with a `for` loop that partially simulates a digital clock. It displays the seconds from 0 to 59:

```
cout << fixed << right;
cout.fill('0');
for (int seconds = 0; seconds < 60; seconds++)
     cout << setw(2) << seconds << endl;
```

NOTE: The `fill` member function of `cout` changes the fill character, which is a space by default. In the program segment above, the `fill` function causes a zero to be printed in front of all single digit numbers.

We can add a `minutes` variable and nest the loop above inside another loop that cycles through 60 minutes:

```
cout << fixed << right;
cout.fill('0');
for (int minutes = 0; minutes < 60; minutes++)
{
     for (int seconds = 0; seconds < 60; seconds++)
     {
          cout << setw(2) << minutes << ":";
          cout << setw(2) << seconds << endl;
     }
}
```

To make the simulated clock complete, another variable and loop can be added to count the hours:

```
cout << fixed << right;
cout.fill('0');
for (int hours = 0; hours < 24; hours++)
{
     for (int minutes = 0; minutes < 60; minutes++)
     {
          for (int seconds = 0; seconds < 60; seconds++)
          {
               cout << setw(2) << hours << ":";
               cout << setw(2) << minutes << ":";
               cout << setw(2) << seconds << endl;
          }
     }
}
```

The output of the previous program segment follows:

```
00:00:00
00:00:01
00:00:02
   .      (The program will count through each second of 24 hours.)
   .
   .
23:59:59
```

The innermost loop will iterate 60 times for each iteration of the middle loop. The middle loop will iterate 60 times for each iteration of the outermost loop. When the outermost loop has iterated 24 times, the middle loop will have iterated 1,440 times and the innermost loop will have iterated 86,400 times!

The simulated clock example brings up a few points about nested loops:

- An inner loop goes through all of its iterations for each iteration of an outer loop.
- Inner loops complete their iterations faster than outer loops.
- To get the total number of iterations of a nested loop, multiply the number of iterations of all the loops.

Program 5-14 is another test-averaging program. It asks the user for the number of students and the number of test scores per student. A nested inner loop, in lines 26 through 33, asks for all the test scores for one student, iterating once for each test score. The outer loop in lines 23 through 37 iterates once for each student.

Program 5-14

```
1  // This program averages test scores. It asks the user for the
2  // number of students and the number of test scores per student.
3  #include <iostream>
4  #include <iomanip>
5  using namespace std;
```

(program continues)

Program 5-14 *(continued)*

```
 6
 7  int main()
 8  {
 9       int numStudents,   // Number of students
10           numTests;      // Number of tests per student
11       double total,      // Accumulator for total scores
12              average;    // Average test score
13
14       // Set up numeric output formatting.
15       cout << fixed << showpoint << setprecision(1);
16
17       // Get the number of students.
18       cout << "This program averages test scores.\n";
19       cout << "For how many students do you have scores? ";
20       cin >> numStudents;
21
22       // Get the number of test scores per student.
23       cout << "How many test scores does each student have? ";
24       cin >> numTests;
25
26       // Determine each student's average score.
27       for (int student = 1; student <= numStudents; student++)
28       {
29            total = 0;     // Initialize the accumulator.
30            for (int test = 1; test <= numTests; test++)
31            {
32                 double score;
33                 cout << "Enter score " << test << " for ";
34                 cout << "student " << student << ": ";
35                 cin >> score;
36                 total += score;
37            }
38            average = total / numTests;
39            cout << "The average score for student " << student;
40            cout << " is " << average << ".\n\n";
41       }
42       return 0;
43  }
```

Program Output with Example Input Shown in Bold

```
This program averages test scores.
For how many students do you have scores? 2 [Enter]
How many test scores does each student have? 3 [Enter]
Enter score 1 for student 1: 84 [Enter]
Enter score 2 for student 1: 79 [Enter]
Enter score 3 for student 1: 97 [Enter]
The average score for student 1 is 86.7.

Enter score 1 for student 2: 92 [Enter]
Enter score 2 for student 2: 88 [Enter]
Enter score 3 for student 2: 94 [Enter]
The average score for student 2 is 91.3.
```