expression instead of a relational expression for the test part of a for loop, you still produce valid code. That's because you can use any valid C++ expression for a for loop test condition. Remember, nonzero values test as true and zero tests as false. An expression that assigns 4 to musicians has the value 4 and is treated as true. If you come from a language, such as Pascal or BASIC, that uses = to test for equality, you might be particularly prone to this slip.

Listing 5.10 shows a situation in which you can make this sort of error. The program attempts to examine an array of quiz scores and stop when it reaches the first score that's not a 20. It shows a loop that correctly uses comparison and then one that mistakenly uses assignment in the test condition. The program also has another egregious design error that you'll see how to fix later. (You learn from your mistakes, and Listing 5.10 is happy to help in that respect.)

Listing 5.10 equal.cpp

```
// equal.cpp -- equality vs assignment
#include <iostream>
using namespace std;
int main()
{
    int quizscores[10] =
    { 20, 20, 20, 20, 20, 19, 20, 18, 20, 20};

    cout << "Doing it right:\n";        
    int i;
    for (i = 0; quizscores[i] == 20; i++)
        cout << "quiz " << i << " is a 20\n";

    cout << "Doing it dangerously wrong:\n";        
    for (i = 0; quizscores[i] = 20; i++)
        cout << "quiz " << i << " is a 20\n";

    return 0;
}
```

Because this program has a serious problem, you might prefer reading about it to actually