Extra for Experts

Earlier in this chapter you learned that 0 represents false and 1 represents true. But when it comes to making decisions based on a control expression, C++ considers anything other than 0 to be true. Therefore, any non-zero integer can represent true.

Why is this useful? Suppose you are using an integer to hold a value that represents an error condition. Zero (0) represents an error-free condition. When a value other than zero is present, however, an error exists and a message like the one below can be printed regardless of what error has occurred.

```
if (ErrorCode)
 { cout << "An error has occurred.\n"; }
```

USING if/else

The *if/else structure* is sometimes called a *two-way selection structure*. Using if/else, one block of code is executed if the control expression is true and another block is executed if the control expression is false. Consider the code fragment below.

```
if (i < 0)
  {cout << "The number is negative.\n"; }
else
  {cout << "The number is zero or positive.\n";}
```

The *else* portion of the structure is executed if the control expression is false. Figure 7-7 shows a flowchart for a two-way selection structure.

The code shown in Figure 7-8 adds an else clause to the if structure in the program in Exercise 7-4. Output is improved by providing information on whether the city's population qualifies it as one of the 100 largest U.S. cities. If

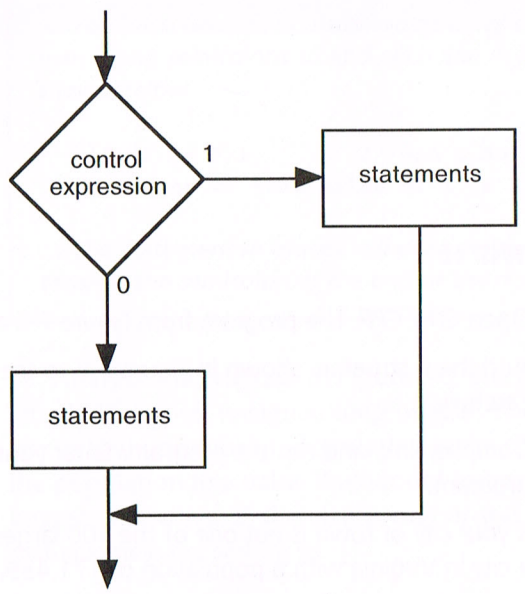

FIGURE 7-7
The if/else structure is a two-way selection structure.


```
if (population >= 171439)
  {
    cout << "According to the 1990 census, " << city_name
         << " is one of\nthe 100 largest U.S. cities.\n";
  }
 else
  {
    cout << "According to the 1990 census, " << city_name
         << " is not one of\nthe 100 largest U.S. cities.\n";
  }
```

FIGURE 7-8
With an if/else structure, one of the two blocks of code will *always* be executed.

the population is 171,439 or more, the first output statement is executed; otherwise the second output statement is executed. In every case, either one or the other output statement is executed.

PITFALLS

Many programmers make the mistake of using > or < when they really need >= or <=. In the code segment in Figure 7-8, using > rather than >= would cause Newport News, the 100th largest city, to be excluded because its population is 171,439, not greater than 171,439.

EXERCISE 7-5 USING if/else

1. Add the else clause shown in Figure 7-8 to the if structure in the program on your screen. Save the new program as *CITYELSE.CPP*.
2. Compile, link, and run the program.
3. Enter the city of Gary, Indiana (population 116,646).
4. Run the program again using Raleigh, North Carolina (population 212,050).
5. Close the source code file.

NESTED if STRUCTURES

You can place if structures within other if structures. When an if or if/else structure is placed within another if or if/else structure, the structures are said to be ***nested***. The flowchart in Figure 7-9 decides whether a student is exempt from a final exam based on grade average and days absent.

To be exempt from the final, a student must have a 90 average or better and cannot have missed more than three days of class. The algorithm first determines if the student's average is greater than or equal to 90. If the result is false, the student must take the final exam. If the result is true, the number of days absent is checked to determine if the other exemption requirement is met. Figure 7-10 shows the algorithm as a C++ code segment.