When the computer makes a comparison, the comparison results in a value of zero or one. If the resulting value is zero, it means the comparison proved false. If the result is one, the comparison proved true. So in C++, false is represented by the integer 0 and true is represented by the integer 1.

Extra for Experts

Fuzzy Logic

Fuzzy logic is a system that allows more than simply true or false. Fuzzy logic allows for some gray area. For example, instead of simply having a 0 for false and a 1 for true, fuzzy logic might allow a 0.9 as a way of saying "it's probably true."

Practical applications of fuzzy logic include things like the thermostat on your home's air conditioner. A standard thermostat turns the air conditioner on when the temperature goes above the desired comfort level. This causes the temperature in the house to rise and fall above and below the thermostat setting. A fuzzy logic thermostat could sense that the temperature is rising, and turn on the air conditioner before the temperature rises above the desired level. The result is a more stable room temperature and conservation of energy.

On the Net

For more information about fuzzy logic, see http://www.ProgramCPP.com. See topic 7.1.1.

RELATIONAL OPERATORS

To make comparisons, C++ provides a set of ***relational operators,*** shown in Table 7-1. They are similar to the symbols you have used in math when working with equations and inequalities.

RELATIONAL OPERATOR	MEANING	EXAMPLE
==	equal to	i == 1
>	greater than	i > 2
<	less than	i < 0
>=	greater than or equal to	i >= 6
<=	less than or equal to	i <= 10
!=	not equal to	i != 12

TABLE 7-1

The relational operators are used to create expressions like the examples in Table 7-1. The result of the expression is one (true) if the data meets the requirements of the comparison. Otherwise, the result of the expression is zero (false). For example, the result of 2 > 1 is one (true), and the result of 2 < 1 is zero (false).

The program in Figure 7-3 demonstrates how expressions are made from relational operators. The result of the expressions is to be displayed as either a one or zero.

Note

Be careful when using the >= and <= operators. The order of the symbols is critical. Switching the symbols will result in an error.

WARNING

Do not confuse the relational operator (==) with the assignment operator (=). Use == for comparisons and = for assignments.


```
#include<iostream.h>

main()
{
 int i = 2;
 int j = 3;
 int true_false;

 cout << (i == 2) << '\n'; // displays a 1 (true)
 cout << (i == 1) << '\n'; // displays a 0 (false)
 cout << (j > i) << '\n';
 cout << (j < i) << '\n';  // Can you predict the
 cout << (j <= 3) << '\n'; // output of the rest of
 cout << (j >= i) << '\n'; // these statements?
 cout << (j != i) << '\n';

 true_false = (j < 4); // The result can be stored to an integer
 cout << true_false << '\n';
 return 0;
}
```

FIGURE 7-3
Expressions created using relational operators return either a 1 or a 0.

EXERCISE 7-1 RELATIONAL OPERATORS

1. Load the program *RELATE.CPP*. The program from Figure 7-3 will appear. Can you predict its output?
2. Compile, link, and run the program.
3. After you have analyzed the output, close the source code file.

LOGICAL OPERATORS

Sometimes it takes more than two comparisons to obtain the desired result. For example, if you want to test to see if an integer is in the range 1 to 10, you must do two comparisons. In order for the integer to fall within the range, it must be greater than 0 *and* less than 11.

C++ provides three ***logical operators*** for multiple comparisons. Table 7-2 shows the three logical operators and their meaning.

Figure 7-4 shows three diagrams called ***truth tables.*** They will help you understand the result of comparisons with the logical operators *and*, *or*, and *not*.

Note

The key used to enter the or operator (||) is usually located near the Enter or Return key. It is usually on the same key with the backslash (\).

LOGICAL OPERATOR	MEANING	EXAMPLE
&&	and	(j == 1 && k == 2)
||	or	(j == 1 || k == 2)
!	not	result = !(j == 1 && k == 2)

TABLE 7-2