AND

A	B	A && B
false (0)	false (0)	false (0)
false (0)	true (1)	false (0)
true (1)	false (0)	false (0)
true (1)	true (1)	true (1)

OR

A	B	A || B
false (0)	false (0)	false (0)
false (0)	true (1)	true (1)
true (1)	false (0)	true (1)
true (1)	true (1)	true (1)

NOT

A	!A
false (0)	true (1)
true (1)	false (0)

FIGURE 7-4
Truth tables illustrate the results of logical operators.

Consider the following C++ statement.

```
in_range = (i > 0 && i < 11);
```

The variable **in_range** is assigned the value 1 if the value of **i** falls into the defined range, and 0 if the value of **i** does not fall into the defined range.

The not operator (!) turns true to false and false to true. For example, suppose you have a program that catalogs old movies. Your program uses an integer variable named **InColor** that has the value zero if the movie was filmed in black and white and the value one if the movie was filmed in color. In the statement below, the variable **Black_and_White** is set to one (true) if the movie is *not* in color. Therefore, if the movie is in color, **Black_and_White** is set to zero (false).

```
Black_and_White = !InColor;
```

EXERCISE 7-2 LOGICAL OPERATORS

1. Enter the following program into a blank editor screen. Save the source code as *LOGICAL.CPP*.

```
#include<iostream.h>

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

  true_false = (i < 3 && j > 3);
  cout << "The result of (i < 3 && j > 3) is " << true_false << '\n';

  true_false = (i < 3 && j >= 3);
  cout << "The result of (i < 3 && j >= 3) is " << true_false << '\n';

  cout << "The result of (i == 1 || i == 2) is "
       << (i == 1 || i == 2) << '\n';

  true_false = (j < 4);
  cout << "The result of (j < 4) is " << true_false << '\n';

  true_false = !true_false;
  cout << "The result of !true_false is " << !true_false << '\n';
  return 0;
}
```


2. Compile and run the program to see the output.
3. After you have analyzed the output, close the source code file.

COMBINING MORE THAN TWO COMPARISONS

You can use logical operators to combine more than two comparisons. Consider the statement below that decides whether it is okay for a person to ride a roller coaster.

```
ok_to_ride = (height_in_inches > 45 && !back_trouble
              && !heart_trouble);
```

In the statement above, **back_trouble** and **heart_trouble** hold the value 0 or 1 depending on whether the person being considered has the problem. For example, if the person has back trouble, the value of **back_trouble** is set to 1. The not operator (!) is used because it is okay to ride if the person does *not* have back trouble and does *not* have heart trouble. The entire statement says that it is okay to ride if the person's height is greater than 45 inches *and* the person has no back trouble *and* no heart trouble.

ORDER OF LOGICAL OPERATIONS

You can mix logical operators in statements as long as you understand the order in which the logical operators will be applied. The *not* operator (!) is applied first, then the *and* operator (&&), and finally the *or* operator (||). Consider the statement below.

```
dog_acceptable = (white || black && friendly);
```

The example above illustrates why it is important to know the order in which logical operators are applied. At first glance it may appear that the statement above would consider a dog to be acceptable if the dog is either white or black and also friendly. But in reality, the statement above considers a white dog that wants to chew your leg off to be an acceptable dog. Why? Because the *and* operator is evaluated first and then the result of the *and* operation is used for the *or* operation. The statement can be corrected with some additional parentheses, as shown below.

```
dog_acceptable = ((white || black) && friendly);
```

C++ evaluates operations in parentheses first just like in arithmetic statements.

EXERCISE 7-3 MIXING LOGICAL OPERATORS

1. Open *LOGICAL2.CPP*.
2. Compile, link, and run the program to see the effect of the parentheses.
3. Close the source code file.

SHORT-CIRCUIT EVALUATION

Suppose you have decided you want to go to a particular concert. You can only go, however, if you can get tickets and if you can get off work. Before you check whether you can get off work, you find out that the concert is sold out and