Equality Operators

Although the relational operators are denoted by the same symbols as in many other programming languages, the ***equality operators*** have a unique appearance (Table 5.2). The "equal to" operator is two adjacent = characters, not one, since a single = character represents the assignment operator. The "not equal to" operator is also two characters: ! and =.

Table 5.2
Equality Operators

Symbol	*Meaning*
==	equal to
!=	not equal to

Like the relational operators, the equality operators are left associative and produce either 0 (false) or 1 (true) as their result. However, the equality operators have *lower* precedence than the relational operators. For example, the expression

```
i < j == j < k
```

is equivalent to

```
(i < j) == (j < k)
```

which is true if `i < j` and `j < k` are both true or both false.

Clever programmers sometimes exploit the fact that the relational and equality operators return integer values. For example, the value of the expression `(i >= j) + (i == j)` is either 0, 1, or 2, depending on whether `i` is less than, greater than, or equal to `j`, respectively. Tricky coding like this generally isn't a good idea, however; it makes programs hard to understand.

Logical Operators

More complicated logical expressions can be built from simpler ones by using the ***logical operators:*** *and*, *or*, and *not* (Table 5.3). The ! operator is unary, while && and || are binary.

Table 5.3
Logical Operators

Symbol	*Meaning*
!	logical negation
&&	logical *and*
||	logical *or*

The logical operators produce either 0 or 1 as their result. Often, the operands will have values of 0 or 1, but this isn't a requirement; the logical operators treat any nonzero operand as a true value and any zero operand as a false value.

The logical operators behave as follows:

- !*expr* has the value 1 if *expr* has the value 0.
- *expr1* && *expr2* has the value 1 if the values of *expr1* and *expr2* are both nonzero.