5.1 Logical Expressions

Several of C's statements, including the `if` statement, must test the value of an expression to see if it is "true" or "false." For example, an `if` statement might need to test the expression `i < j`; a true value would indicate that `i` is less than `j`. In many programming languages, an expression such as `i < j` would have a special "Boolean" or "logical" type. Such a type would have only two values, *false* and *true*. In C, however, a comparison such as `i < j` yields an integer: either 0 (false) or 1 (true). With this in mind, let's look at the operators that are used to build logical expressions.

Relational Operators

C's ***relational operators*** (Table 5.1) correspond to the <, >, ≤, and ≥ operators of mathematics, except that they produce 0 (false) or 1 (true) when used in expressions. For example, the value of `10 < 11` is 1; the value of `11 < 10` is 0.

Table 5.1
Relational Operators

Symbol	*Meaning*
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

The relational operators can be used to compare integers and floating-point numbers, with operands of mixed types allowed. Thus, `1 < 2.5` has the value 1, while `5.6 < 4` has the value 0.

The precedence of the relational operators is lower than that of the arithmetic operators; for example, `i + j < k - 1` means `(i + j) < (k - 1)`. The relational operators are left associative.

⚠ The expression

```
i < j < k
```

is legal in C, but doesn't have the meaning that you might expect. Since the < operator is left associative, this expression is equivalent to

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

In other words, the expression first tests whether `i` is less than `j`; the 1 or 0 produced by this comparison is then compared to `k`. The expression does *not* test whether `j` lies between `i` and `k`. (We'll see later in this section that the correct expression would be `i < j && j < k`.)