- *expr1* `||` *expr2* has the value 1 if either *expr1* or *expr2* (or both) has a nonzero value.

In all other cases, these operators produce the value 0.

Both `&&` and `||` perform "short-circuit" evaluation of their operands. That is, these operators first evaluate the left operand, then the right operand. If the value of the expression can be deduced from the value of the left operand alone, then the right operand isn't evaluated. Consider the following expression:

```
(i != 0) && (j / i > 0)
```

To find the value of this expression, we must first evaluate `(i != 0)`. If `i` isn't equal to 0, then we'll need to evaluate `(j / i > 0)` to determine whether the entire expression is true or false. However, if `i` is equal to 0, then the entire expression must be false, so there's no need to evaluate `(j / i > 0)`. The advantage of short-circuit evaluation is apparent—without it, evaluating the expression would have caused a division by zero.

⚠ Be wary of side effects in logical expressions. Thanks to the short-circuit nature of the `&&` and `||` operators, side effects in operands may not always occur. Consider the following expression:

```
i > 0 && ++j > 0
```

Although `j` is apparently incremented as a side effect of evaluating the expression, that isn't always the case. If `i > 0` is false, then `++j > 0` is not evaluated, so `j` isn't incremented. The problem can be fixed by changing the condition to `++j > 0 && i > 0` or, even better, by incrementing `j` separately.

The `!` operator has the same precedence as the unary plus and minus operators. The precedence of `&&` and `||` is lower than that of the relational and equality operators; for example, `i < j && k == m` means `(i < j) && (k == m)`. The `!` operator is right associative; `&&` and `||` are left associative.

5.2 The `if` Statement

The `if` statement allows a program to choose between two alternatives by testing the value of an expression. In its simplest form, the `if` statement has the form

`if` statement

`if (` *expression* `)` *statement*

Notice that the parentheses around the expression are mandatory; they're part of the `if` statement, not part of the expression. Also note that the word `then` doesn't come after the parentheses, as it would in some programming languages.