A switch statement is often easier to read than a cascaded if statement. Moreover, switch statements are often faster than if statements, especially when there are more than a handful of cases.

Q&A

In its most common form, the switch statement has the form

switch statement

```
switch ( expression ) {
  case constant-expression : statements
  ...
  case constant-expression : statements
  default : statements
}
```

The switch statement is fairly complex; let's look at its components one by one:

characters ➤7.3

- ***Controlling expression.*** The word switch must be followed by an integer expression in parentheses. Characters are treated as integers in C and thus can be tested in switch statements. Floating-point numbers and strings don't qualify, however.
- ***Case labels.*** Each case begins with a label of the form

  case *constant-expression* :

  A ***constant expression*** is much like an ordinary expression except that it can't contain variables or function calls. Thus, 5 is a constant expression, and 5 + 10 is a constant expression, but n + 10 isn't a constant expression (unless n is a macro that represents a constant). The constant expression in a case label must evaluate to an integer (characters are also acceptable).
- ***Statements.*** After each case label comes any number of statements. No braces are required around the statements. (Enjoy it—this is one of the few places in C where braces aren't required.) The last statement in each group is normally break.

Duplicate case labels aren't allowed. The order of the cases doesn't matter; in particular, the default case doesn't need to come last.

Only one constant expression may follow the word case; however, several case labels may precede the same group of statements:

```
switch (grade) {
  case 4:
  case 3:
  case 2:
  case 1:  printf("Passing");
           break;
  case 0:  printf("Failing");
           break;
  default: printf("Illegal grade");
           break;
}
```