C allows us to insert any amount of space—blanks, tabs, and new-line characters—between tokens. This rule has several important consequences for program layout:

- *Statements can be divided* over any number of lines. The following statement, for example, is so long that it would be hard to squeeze it onto a single line:

```
printf("Dimensional weight (pounds): %d\n",
  (volume + INCHES_PER_POUND - 1) / INCHES_PER_POUND);
```

- *Space between tokens* makes it easier for the eye to separate them. For this reason, I usually put a space before and after each operator:

```
volume = height * length * width;
```

  I also put a space after each comma. Some programmers go even further, putting spaces around parentheses and other punctuation.

- *Indentation* can make nesting easier to spot. For example, we should indent declarations and statements to make it clear that they're nested inside `main`.

Q&A

- *Blank lines* can divide a program into logical units, making it easier for the reader to discern the program's structure. A program with no blank lines is as hard to read as a book with no chapters.

The `celsius.c` program of Section 2.6 illustrates several of these guidelines. Let's take a closer look at the `main` function in that program:

```
int main(void)
{
  float fahrenheit, celsius;

  printf("Enter Fahrenheit temperature: ");
  scanf("%f", &fahrenheit);

  celsius = (fahrenheit - FREEZING_PT) * SCALE_FACTOR;

  printf("Celsius equivalent: %.1f\n", celsius);

  return 0;
}
```

First, observe how the space around =, -, and * makes these operators stand out. Second, notice how the indentation of declarations and statements makes it obvious that they all belong to `main`. Finally, note how blank lines divide `main` into five parts: (1) declaring the `fahrenheit` and `celsius` variables; (2) obtaining the Fahrenheit temperature; (3) calculating the value of `celsius`; (4) printing the Celsius temperature; and (5) returning to the operating system.

While we're on the subject of program layout, notice how I've placed the { token underneath `main()` and put the matching } on a separate line, aligned with {. Putting } on a separate line lets us insert or delete statements at the end of the function; aligning it with { makes it easy to spot the end of `main`.

A final note: Although extra spaces can be added *between* tokens, it's not pos-