The integer variables are declared on one line, and the floating-point (non-integer) variables on another.

Keep variables that are defined with a value on a line by themselves. To wit:

```
int first=1;
int the_rest;
```

Constants and Variables

In addition to the variable, the C language has something called a *constant*. It's used like a variable, though its value never changes.

Suppose that one day someone dupes you into writing a trigonometry program. In this type of program, you have to use the dreaded value π (pi). That's equal to 3.1415926 (and on and on). Because it never changes, you can create a constant named `pi` that is equal to that value.

Another example of a constant is a quoted string of text:

```
printf("%s","This is a string of text");
```

The text `"This is a string of text"` is a constant used with this `printf()` function. A variable can go there, though a string constant — a literal, quoted string — is used instead.

- A constant is used just like a variable, though its value never changes.
- A numeric constant is a number value, like π, that remains the same throughout your program.
- π is pronounced "pie." It's the Greek letter *p*. We pronounce the English letter *p* as *"pee."*
- A string constant is a bit of text that never changes, though that's really true of most text in a C language program. This chapter, therefore, concentrates primarily on numeric constants.

Dreaming up and defining constants

All this constant nonsense can seem silly. Then one day, you're faced with a program like SPEED.C — except that the program is much longer — and you truly come to realize the value of a C language constant and the nifty `#define` directive you read about later in this chapter: