The first declaration states that `height` is a variable of type `int`, meaning that `height` can store an integer value. The second declaration says that `profit` is a variable of type `float`.

If several variables have the same type, their declarations can be combined:

```
int height, length, width, volume;
float profit, loss;
```

Notice that each complete declaration ends with a semicolon.

Our first template for `main` didn't include declarations. When `main` contains declarations, these must precede statements:

```
int main(void)
{
  declarations
  statements
}
```

blocks ➤ 10.3

As we'll see in Chapter 9, this is true of functions in general, as well as blocks (statements that contain embedded declarations). As a matter of style, it's a good idea to leave a blank line between the declarations and the statements.

C99

In C99, declarations don't have to come before statements. For example, `main` might contain a declaration, then a statement, and then another declaration. For compatibility with older compilers, the programs in this book don't take advantage of this rule. However, it's common in C++ and Java programs not to declare variables until they're first needed, so this practice can be expected to become popular in C99 programs as well.

Assignment

A variable can be given a value by means of *assignment*. For example, the statements

```
height = 8;
length = 12;
width = 10;
```

assign values to `height`, `length`, and `width`. The numbers `8`, `12`, and `10` are said to be *constants*.

Before a variable can be assigned a value—or used in any other way, for that matter—it must first be declared. Thus, we could write

```
int height;
height = 8;
```

but not

```
height = 8;    /*** WRONG ***/
int height;
```