Declaring a Structure Tag

A *structure tag* is a name used to identify a particular kind of structure. The following example declares a structure tag named `part`:

```
struct part {
  int number;
  char name[NAME_LEN+1];
  int on_hand;
};
```

Notice the semicolon that follows the right brace—it must be present to terminate the declaration.

Accidentally omitting the semicolon at the end of a structure declaration can cause surprising errors. Consider the following example:

```
struct part {
  int number;
  char name[NAME_LEN+1];
  int on_hand;
}               /*** WRONG: semicolon missing ***/

f(void)
{
  ...
  return 0;     /* error detected at this line */
}
```

The programmer failed to specify the return type of the function `f` (a bit of sloppy programming). Since the preceding structure declaration wasn't terminated properly, the compiler assumes that `f` returns a value of type `struct part`. The error won't be detected until the compiler reaches the first `return` statement in the function. The result: a cryptic error message.

Once we've created the `part` tag, we can use it to declare variables:

```
struct part part1, part2;
```

Unfortunately, we can't abbreviate this declaration by dropping the word `struct`:

```
part part1, part2;   /*** WRONG ***/
```

`part` isn't a type name; without the word `struct`, it is meaningless.

Since structure tags aren't recognized unless preceded by the word `struct`, they don't conflict with other names used in a program. It would be perfectly legal (although more than a little confusing) to have a variable named `part`.

Incidentally, the declaration of a structure *tag* can be combined with the declaration of structure *variables*: