

tells the compiler that `f` is a function that returns a `float` value and has one argument, also of type `float`.

In general, a declaration has the following appearance:

declaration *declaration-specifiers declarators ;*

Declaration specifiers describe the properties of the variables or functions being declared. *Declarators* give their names and may provide additional information about their properties.

Declaration specifiers fall into three categories:

- *Storage classes.* There are four storage classes: `auto`, `static`, `extern`, and `register`. At most one storage class may appear in a declaration; if present, it should come first.
- *Type qualifiers.* In C89, there are only two type qualifiers: `const` and `volatile`. C99 has a third type qualifier, `restrict`. A declaration may contain zero or more type qualifiers.
- *Type specifiers.* The keywords `void`, `char`, `short`, `int`, `long`, `float`, `double`, `signed`, and `unsigned` are all type specifiers. These words may be combined as described in Chapter 7; the order in which they appear doesn't matter (`int unsigned long` is the same as `long unsigned int`). Type specifiers also include specifications of structures, unions, and enumerations (for example, `struct point { int x, y; }`, `struct { int x, y; }`, or `struct point`). Type names created using `typedef` are type specifiers as well.

C99 (C99 has a fourth kind of declaration specifier, the *function specifier*, which is used only in function declarations. This category has just one member, the keyword `inline`.) Type qualifiers and type specifiers should follow the storage class, but there are no other restrictions on their order. As a matter of style, I'll put type qualifiers before type specifiers.

Declarators include identifiers (names of simple variables), identifiers followed by `[]` (array names), identifiers preceded by `*` (pointer names), and identifiers followed by `()` (function names). Declarators are separated by commas. A declarator that represents a variable may be followed by an initializer.

Let's look at a few examples that illustrate these rules. Here's a declaration with a storage class and three declarators:

```

      storage class      declarators
        |               |   |   |
        v               v   v   v
static float x, y, *p;
              ^
              |
            type specifier
  
```

The following declaration has a type qualifier but no storage class. It also has an initializer: