

disabling code ► 14.4 In any event, there's a better way to disable portions of a program, as we'll see later.

Q: Where does the `float` type get its name? [p. 17]

A: `float` is short for “floating-point,” a technique for storing numbers in which the decimal point “floats.” A `float` value is usually stored in two parts: the fraction (or mantissa) and the exponent. The number 12.0 might be stored as 1.5×2^3 , for example, where 1.5 is the fraction and 3 is the exponent. Some programming languages call this type `real` instead of `float`.

Q: Why do floating-point constants need to end with the letter `f`? [p. 19]

A: For the full explanation, see Chapter 7. Here's the short answer: a constant that contains a decimal point but doesn't end with `f` has type `double` (short for “double precision”). `double` values are stored more accurately than `float` values. Moreover, `double` values can be larger than `float` values, which is why we need to add the letter `f` when assigning to a `float` variable. Without the `f`, a warning may be generated about the possibility of a number being stored into a `float` variable that exceeds the capacity of the variable.

***Q: Is it really true that there's no limit on the length of an identifier? [p. 26]**

A: Yes and no. The C89 standard says that identifiers may be arbitrarily long. However, compilers are only required to remember the first 31 characters (63 characters in C99). Thus, if two names begin with the same 31 characters, a compiler might be unable to distinguish between them.

C99

external linkage ► 18.2

To make matters even more complicated, there are special rules for identifiers with external linkage; most function names fall into this category. Since these names must be made available to the linker, and since some older linkers can handle only short names, only the first six characters are significant in C89. Moreover, the case of letters may not matter. As a result, `ABCDEFGH` and `abcdefgh` might be treated as the same name. (In C99, the first 31 characters are significant, and the case of letters is taken into account.)

C99

Most compilers and linkers are more generous than the standard, so these rules aren't a problem in practice. Don't worry about making identifiers too long—worry about making them too short.

Q: How many spaces should I use for indentation? [p. 28]

A: That's a tough question. Leave too little space, and the eye has trouble detecting indentation. Leave too much, and lines run off the screen (or page). Many C programmers indent nested statements eight spaces (one tab stop), which is probably too much. Studies have shown that the optimum amount of indentation is three spaces, but many programmers feel uncomfortable with numbers that aren't a power of two. Although I normally prefer to indent three or four spaces, I'll use two spaces in this book so that my programs will fit within the margins.