Q & A

Q: I notice that you use the term "standard header" rather than "standard header file." Is there any reason for not using the word "file"?

A: Yes. According to the C standard, a "standard header" need not be a file. Although most compilers do indeed store standard headers as files, the headers could in fact be built into the compiler itself.

Q: Section 14.3 described some disadvantages of using parameterized macros in place of functions. In light of these problems, isn't it dangerous to provide a macro substitute for a standard library function? [p. 531]

A: According to the C standard, a parameterized macro that substitutes for a library function must be "fully protected" by parentheses and must evaluate its arguments exactly once. These rules avoid most of the problems mentioned in Section 14.3.

Exercises

Section 21.1

1. Locate where header files are kept on your system. Find the nonstandard headers and determine the purpose of each.

2. Having located the header files on your system (see Exercise 1), find a standard header in which a macro hides a function.

3. When a macro hides a function, which must come first in the header file: the macro definition or the function prototype? Justify your answer.

4. Make a list of all reserved identifiers in the "future library directions" section of the C99 standard. Distinguish between identifiers that are reserved for use only when a specific header is included versus identifiers that are reserved for use as external names.

*5. The `islower` function, which belongs to `<ctype.h>`, tests whether a character is a lower-case letter. Why would the following macro version of `islower` not be legal, according to the C standard? (You may assume that the character set is ASCII.)

```
#define islower(c) ((c) >= 'a' && (c) <= 'z')
```

6. The `<ctype.h>` header usually defines most of its functions as macros as well. These macros rely on a static array that's declared in `<ctype.h>` but defined in a separate file. A portion of a typical `<ctype.h>` header appears below. Use this sample to answer the following questions.

   (a) Why do the names of the "bit" macros (such as `_UPPER`) and the `_ctype` array begin with an underscore?

   (b) Explain what the `_ctype` array will contain. Assuming that the character set is ASCII, show the values of the array elements at positions 9 (the horizontal tab character), 32 (the space character), 65 (the letter `A`), and 94 (the `^` character). See Section 23.5 for a description of what each macro should return.