Q: The chapter says that a shared array should be declared as follows:

```
extern int a[];
```

Since arrays and pointers are closely related, would it be legal to write

```
extern int *a;
```

instead? [p. 356]

A: No. When used in expressions, arrays "decay" into pointers. (We've noticed this behavior when an array name is used as an argument in a function call.) In variable declarations, however, arrays and pointers are distinct types.

Q: Does it hurt if a source file includes headers that it doesn't really need?

A: Not unless the header has a declaration or definition that conflicts with one in the source file. Otherwise, the worst that can happen is a minor increase in the time it takes to compile the source file.

Q: I needed to call a function in the file `foo.c`, so I included the matching header file, `foo.h`. My program compiled, but it won't link. Why?

A: Compilation and linking are completely separate in C. Header files exist to provide information to the compiler, not the linker. If you want to call a function in `foo.c`, then you have to make sure that `foo.c` is compiled and that the linker is aware that it must search the object file for `foo.c` to find the function. Usually this means naming `foo.c` in the program's makefile or project file.

Q: If my program calls a function in `<stdio.h>`, does that mean that all functions in `<stdio.h>` will be linked with the program?

A: No. Including `<stdio.h>` (or any other header) has no effect on linking. In any event, most linkers will link only functions that your program actually needs.

Q: Where can I get the `make` utility? [p. 367]

A: `make` is a standard UNIX utility. The GNU version, known as GNU Make, is included in most Linux distributions. It's also available directly from the Free Software Foundation (*www.gnu.org/software/make/*).

Exercises

Section 15.1

1. Section 15.1 listed several advantages of dividing a program into multiple source files.
   (a) Describe several other advantages.
   (b) Describe some disadvantages.

Section 15.2

Ⓦ 2. Which of the following should *not* be put in a header file? Why not?
   (a) Function prototypes
   (b) Function definitions

(c) Macro definitions
(d) Type definitions

3. We saw that writing `#include <file>` instead of `#include "file"` may not work if *file* is one that we've written. Would there be any problem with writing `#include "file"` instead of `#include <file>` if *file* is a system header?

4. Assume that `debug.h` is a header file with the following contents:

```
#ifdef DEBUG
#define PRINT_DEBUG(n) printf("Value of " #n ": %d\n", n)
#else
#define PRINT_DEBUG(n)
#endif
```

Let `testdebug.c` be the following source file:

```
#include <stdio.h>

#define DEBUG
#include "debug.h"

int main(void)
{
  int i = 1, j = 2, k = 3;

#ifdef DEBUG
  printf("Output if DEBUG is defined:\n");
#else
  printf("Output if DEBUG is not defined:\n");
#endif

  PRINT_DEBUG(i);
  PRINT_DEBUG(j);
  PRINT_DEBUG(k);
  PRINT_DEBUG(i + j);
  PRINT_DEBUG(2 * i + j - k);

  return 0;
}
```

(a) What is the output when the program is executed?
(b) What is the output if the `#define` directive is removed from `testdebug.c`?
(c) Explain why the output is different in parts (a) and (b).
(d) Is it necessary for the `DEBUG` macro to be defined *before* `debug.h` is included in order for `PRINT_DEBUG` to have the desired effect? Justify your answer.

Section 15.4

5. Suppose that a program consists of three source files—`main.c`, `f1.c`, and `f2.c`—plus two header files, `f1.h` and `f2.h`. All three source files include `f1.h`, but only `f1.c` and `f2.c` include `f2.h`. Write a makefile for this program, assuming that the compiler is `gcc` and that the executable file is to be named `demo`.

Ⓦ 6. The following questions refer to the program described in Exercise 5.
(a) Which files need to be compiled when the program is built for the first time?
(b) If `f1.c` is changed after the program has been built, which files need to be recompiled?
(c) If `f1.h` is changed after the program has been built, which files need to be recompiled?
(d) If `f2.h` is changed after the program has been built, which files need to be recompiled?