27. What functions can access a global variable that appears in the same file with them?
28. What functions can access a local variable?
29. A static local variable is used to

    a. make a variable visible to several functions.

    b. make a variable visible to only one function.

    c. conserve memory when a function is not executing.

    d. retain a value when a function is not executing.
30. In what unusual place can you use a function call when a function returns a value by reference?

Exercises

Answers to the starred exercises can be found in Appendix G.

*1. Refer to the CIRCAREA program in Chapter 2, "C++ Programming Basics." Write a function called `circarea()` that finds the area of a circle in a similar way. It should take an argument of type `float` and return an argument of the same type. Write a `main()` function that gets a radius value from the user, calls `circarea()`, and displays the result.

*2. Raising a number `n` to a power `p` is the same as multiplying `n` by itself `p` times. Write a function called `power()` that takes a `double` value for `n` and an `int` value for `p`, and returns the result as a `double` value. Use a default argument of 2 for `p`, so that if this argument is omitted, the number `n` will be squared. Write a `main()` function that gets values from the user to test this function.

*3. Write a function called `zeroSmaller()` that is passed two `int` arguments by reference and then sets the smaller of the two numbers to 0. Write a `main()` program to exercise this function.

*4. Write a function that takes two `Distance` values as arguments and returns the larger one. Include a `main()` program that accepts two `Distance` values from the user, compares them, and displays the larger. (See the RETSTRC program for hints.)

5. Write a function called `hms_to_secs()` that takes three `int` values—for hours, minutes, and seconds—as arguments, and returns the equivalent time in seconds (type `long`). Create a program that exercises this function by repeatedly obtaining a time value in hours, minutes, and seconds from the user (format 12:59:59), calling the function, and displaying the value of seconds it returns.

6. Start with the program from Exercise 11 in Chapter 4, "Structures," which adds two `struct time` values. Keep the same functionality, but modify the program so that it uses two functions. The first, `time_to_secs()`, takes as its only argument a structure of type

time, and returns the equivalent in seconds (type long). The second function, secs_to_time(), takes as its only argument a time in seconds (type long), and returns a structure of type time.

7. Start with the power() function of Exercise 2, which works only with type double. Create a series of overloaded functions with the same name that, in addition to double, also work with types char, int, long, and float. Write a main() program that exercises these overloaded functions with all argument types.
8. Write a function called swap() that interchanges two int values passed to it by the calling program. (Note that this function swaps the values of the variables in the calling program, not those in the function.) You'll need to decide how to pass the arguments. Create a main() program to exercise the function.
9. Repeat Exercise 8, but instead of two int variables, have the swap() function interchange two struct time values (see Exercise 6).
10. Write a function that, when you call it, displays a message telling how many times it has been called: "I have been called 3 times", for instance. Write a main() program that calls this function at least 10 times. Try implementing this function in two different ways. First, use a global variable to store the count. Second, use a local static variable. Which is more appropriate? Why can't you use a local variable?
11. Write a program, based on the sterling structure of Exercise 10 in Chapter 4, that obtains from the user two money amounts in old-style British format (£9:19:11), adds them, and displays the result, again in old-style format. Use three functions. The first should obtain a pounds-shillings-pence value from the user and return the value as a structure of type sterling. The second should take two arguments of type sterling and return a value of the same type, which is the sum of the arguments. The third should take a sterling structure as its argument and display its value.
12. Revise the four-function fraction calculator from Exercise 12, Chapter 4, so that it uses functions for each of the four arithmetic operations. They can be called fadd(), fsub(), fmul(), and fdiv(). Each of these functions should take two arguments of type struct fraction, and return an argument of the same type.