I'm a goto kind of guy.

1 use(s)

The second new area is using a reference as a return value. Normally, the return mechanism copies the returned value to a temporary storage area, which the calling program then accesses. Returning a reference, however, means that the calling program accesses the return value directly without there being a copy. Typically, the reference refers to a reference passed to the function in the first place, so the calling function actually winds up directly accessing one of its own variables. Here, for example, sysopref is a reference to looper, so the return value is the original looper variable in main().

Because use() returns a type sysop reference, it can be used as an argument to any function expecting either a sysop argument or a reference-to-sysop argument, such as use() itself. Thus, the next function call in Listing 8.6 is really two function calls, with one function's return value serving as the argument for the second:

use(use(looper));

The inner function call prints the name and quote members and increments the used member to 2. The function returns sysopref, reducing what's left to the following:

use(sysopref);

Because sysopref is a reference to looper, this function call is equivalent to the following:

use(looper);

So use() displays the two string members again and increments the used member to 3.

Remember

A function that returns a reference is actually an alias for the referred-to variable.

The third new area the program explores is that you can assign a value to a function if that function has a reference type return value: