19 Program Design

Wherever there is modularity there is the potential for misunderstanding:
Hiding information implies a need to check communication.

It's obvious that real-world programs are larger than the examples in this book, but you may not realize just how much larger. Faster CPUs and larger main memories have made it possible to write programs that would have been impractical just a few years ago. The popularity of graphical user interfaces has added greatly to the average length of a program. Most full-featured programs today are at least 100,000 lines long. Million-line programs are commonplace, and it's not unheard-of for a program to have 10 million lines or more.

Q&A

Although C wasn't designed for writing large programs, many large programs have in fact been written in C. It's tricky, and it requires a great deal of care, but it can be done. In this chapter, I'll discuss techniques that have proved to be helpful for writing large programs and show which C features (the `static` storage class, for example) are especially useful.

Writing large programs (often called "programming-in-the-large") is quite different from writing small ones—it's like the difference between writing a term paper (10 pages double-spaced, of course) and a 1000-page book. A large program requires more attention to style, since many people will be working on it. It requires careful documentation. It requires planning for maintenance, since it will likely be modified many times.

Above all, a large program requires careful design and much more planning than a small program. As Alan Kay, the designer of the Smalltalk programming language, puts it, "You can build a doghouse out of anything." A doghouse can be built without any particular design, using whatever materials are at hand. A house for humans, on the other hand, is too complex to just throw together.

Chapter 15 discussed writing large programs in C, but it concentrated on language details. In this chapter, we'll revisit the topic, this time focusing on techniques for good program design. A complete discussion of program design issues is obviously beyond the scope of this book. However, I'll try to cover—briefly—