CHAPTER 8: DOCUMENTATION

The best documentation for a computer program is a clean structure. It also helps if the code is well formatted, with good mnemonic identifiers and labels (if any are needed), and a smattering of enlightening comments. Flowcharts and program descriptions are of secondary importance; the only reliable documentation of a computer program is the code itself. The reason is simple — whenever there are multiple representations of a program, the chance for discrepancy exists. If the code is in error, artistic flowcharts and detailed comments are to no avail. Only by reading the code can the programmer know for sure what the program does.

This is not to say that programmers should never write documentation. Quite the contrary. In a project of any size it is vital to maintain readable descriptions of what each program is supposed to do, how it is used, how it interacts with other parts of the system, and on what principles it is based. These form useful guides to the code. What is *not* useful is a narrative description of what a given routine actually does on a line-by-line basis. Anything that contributes no new information, but merely echoes the code, is superfluous.

If you write your code first in a pseudo-language, as we suggested in Chapter 3, then you already have an excellent "readable description of what each program is supposed to do." Keep the original around to refresh your memory when you have to alter the code — that way you won't have to "decompile" the actual program each time you want to figure out why you did something a certain way. If you include your pseudo-code as comments in your source code, you will in fact be helping everyone who must later read it.

Although comments have no effect on code, of course, they are still physically a part of it, and thus provide most of program documentation. We will devote much of our attention to style in commenting.

One thing we will *not* do is make pronouncements about how many comments a program should have. We have already seen examples that contain none and others with more comments than code. The right amount usually lies between these extremes, but an arbitrary rule, like "one comment for every three lines," is absurd.

A comment is of zero (or negative) value if it is wrong. For example, in