Taken together, this set of constructions is generally adequate for comfortably expressing any sequencing operations in a program. The term "structured programming" is sometimes used (at least in a narrow sense) to refer to the process of programming with nothing but proper nests of these basic operations.

The advantage of this discipline is that since there are no GOTO statements, it is generally easier to follow the flow of control; for the most part such a program reads directly from top to bottom, so the reader doesn't have to follow paths with his fingers all over the listing. And no GOTO's means no labels — there is only one way to reach each statement.

On the other hand, structured programming in this limited sense certainly will not solve all your programming problems. We will see in the rest of this chapter plenty of code that contains only the basic constructions in properly nested combinations, yet which is hard to understand and even incorrect.

Use the fundamental control flow constructs.

Bare Fortran doesn't have any of these fundamental structures. What can you do to cope? We have several suggestions. For the long term, the 1977 Fortran Standard provides an ELSE and a way to group statements after IF and ELSE; it looks like:

```
IF (condition) THEN
    statements
ELSE
    statements
ENDIF
```

These can be nested, and there can be an ELSE IF. Regrettably, Fortran 77 does not have a WHILE statement. There is also a version of the debugging compiler WATFIV, called WATFIV-S, which supports statement grouping, ELSE, and WHILE.

A second possibility, which may be more accessible in the short run, is to use one of the host of Fortran preprocessors which have been developed in the past few years. A preprocessor is a program which translates a Fortran dialect with adequate control flow statements into pure Fortran; ideally you never need to look at the generated Fortran. (The "pseudo-code" that we will present in the next sections is based on Ratfor, a language implemented by one such Fortran preprocessor. It is described in *Software Tools,* by Brian W. Kernighan and P. J. Plauger, Addison-Wesley, 1976.)

A third possibility is to think out your code in a decent language, then translate into Fortran when it comes time to start transcribing the code into machine-readable form. This requires no software, just discipline. To see how it works in practice, consider the following quadratic equation solver, in which IF statements come so thick and fast as to baffle the reader.