CHAPTER 2: EXPRESSION

Writing a computer program eventually boils down to writing a sequence of statements in the language at hand. How each of those statements is expressed determines in large measure the intelligibility of the whole; no amount of commenting, formatting, or supplementary documentation can entirely replace well expressed statements. After all, they determine what the program actually *does*.

It is easy to mistake a sequence of overly-simple expressions for profundity. An extreme example of this is

```
      IF(X .LT. Y) GO TO 30
      IF (Y .LT. Z) GO TO 50
      SMALL = Z
      GO TO 70
30    IF (X .LT. Z) GO TO 60
      SMALL = Z
      GO TO 70
50    SMALL = Y
      GO TO 70
60    SMALL = X
70    ...
```

Ten lines, with four statement numbers and six GOTO's; surely *something* is happening. Before reading further, test yourself. What does this program do?

The mnemonic SMALL is a giveaway — the sequence sets SMALL to the smallest of X, Y, and Z.

There are a number of ways to do this computation. If our purpose is to teach how to compute the minimum, we write

```
      SMALL = X
      IF (Y .LT. SMALL) SMALL = Y
      IF (Z .LT. SMALL) SMALL = Z
```

which is direct and to the point. Labels and GOTO's are not needed. And the generalization to computing the minimum of many elements is obvious.

Say what you mean, simply and directly.

But if we are just trying to get the job done, we use the Fortran built-in function AMIN1, which computes the minimum of two or more floating point numbers: