```
IF A > B
   THEN S = 1;
   ELSE IF A = B
      THEN IF C > D
        THEN S = 2;
        ELSE S = 3;
      ELSE IF C > D
        THEN S = 4;
        ELSE IF C = D
          THEN S = 5;
          ELSE S = 6;
```

Again this is neatly indented to display the structure, but it doesn't help the reader to *understand.* Under what circumstances will S be assigned the value three? It is not easy to tell.

This code is almost in the form of a CASE statement. There is one violation: the case A=B has an IF-ELSE in its THEN clause. It is no surprise that this is the hardest part of the code to comprehend. So let us transform it accordingly:

```
IF A > B THEN
   S = 1;
ELSE IF A = B & C > D THEN
   S = 2;
ELSE IF A = B THEN
   S = 3;
ELSE IF C > D THEN
   S = 4;
ELSE IF C = D THEN
   S = 5;
ELSE
   S = 6;
```

At the cost of one additional comparison, we have obtained a familiar structure. Now we know for certain that one, and only one, case will be executed. Reading from the top down until the proper condition is met tells us which one.

The reader has undoubtedly noticed by now that our personal stylistic conventions for layout, comments, and the like are not absolutely uniform. Nonetheless, we have tried to be reasonably consistent, for unless we are consistent, you will not be able to count on what our formatting is trying to tell you about the programs. Good formatting is a part of good programming.

We conclude with a larger example of documentation. This program solves a set of N linear equations in N unknowns, using Gauss-Seidel iteration. Originally from a textbook, it appeared in an article entitled "How to Write a Readable Fortran Program," in *Datamation*, October, 1972.