```
IF PRICE(J) <= LOT THEN DO;
   LOT = PRICE(J);
   LOCATION = J;
END;
```

Fortran provides nothing analogous to DO-END; this is one of its major failings. There is no way to treat a block of statements as a group (after an IF, for instance), except by putting them into a subroutine or branching around them. This leads to tortuous code indeed if the program is at all complicated. Even so, some usages are clearer than others.

```
      IF (TABLE (NO) .GT. HICOM) GO TO 50
      GO TO 20
50    HICOM = TABLE (NO)
      NUMBER = NO
20    CONTINUE
```

This should be replaced by

```
      IF (TABLE(NO) .LE. HICOM) GOTO 20
         HICOM = TABLE(NO)
         NUMBER = NO
20    CONTINUE
```

Again we indent the statements that are skipped over, to show that they are controlled by the IF. Within the limitations of Fortran, this is about the best we can do.

Use DO-END *and indenting*
to delimit groups of statements.

In PL/I, an IF may be followed by an ELSE part, to express the action to be taken if the condition is not true. But consider

```
IF SWFSTCTL = '1'
   THEN GOTO CONTINUE ;
   ELSE DO ;
          DIVCTL = DIV ;       /* INITIALIZE CONTROL */
          SWFSTCTL = '1' ;
        END ;
CONTINUE :
```

The ELSE is a red herring, serving no purpose here. It should be used only when there are two distinct and mutually exclusive actions depending on one test. If there is only one action, it belongs after the THEN, so that the reason for the action can be stated directly:

```
IF SWFSTCTL ¬= '1' THEN DO;
   DIVCTL = DIV;        /* INITIALIZE CONTROL */
   SWFSTCTL = '1';
END;
```

On the other hand, when there really are two cases, an ELSE should be used: