putting the two statements that set STRING(1) and STRING(80) before the loop, although it might be better to widen the output to 82 columns instead.

The second bug is more serious. We will leave it as an exercise for the reader to verify that even-length input fields are not centered, but are placed one position too far to the right. When the length is 80, this overwrites STRING(81). This operation is illegal, of course, but few implementations bother to catch it, preferring instead to let you deduce for yourself how some other variable mysteriously changes its value in the middle of a computation.

Rather than trying to patch the errors, let us re-examine the data structure. A moment of study reveals that two arrays, PHRASE and STRING, are used to hold a single piece of data, the original input card. We can eliminate this double representation by creating an array BORDER of 80 blanks with a period on each end, then writing out PHRASE with enough of BORDER on each side to provide the necessary periods and blanks.

```
      INTEGER BLANK, PHRASE(80), BORDER(82), LEFT, LNB, RIGHT, RNB
      DATA BLANK /' '/, BORDER(1) /'.'/, BORDER(82) /'.'/
      DO 10 I = 2, 81
         BORDER(I) = BLANK
   10 CONTINUE
C
   20 READ(5,21) (PHRASE(I), I = 1, 80)
   21    FORMAT(80A1)
C FIND LNB=LEFTMOST NON-BLANK, RNB=RIGHTMOST
      LNB = 0
      DO 30 I = 1, 80
         IF (PHRASE(I) .NE. BLANK) RNB = I
         IF (PHRASE(I) .NE. BLANK .AND. LNB .EQ. 0) LNB = I
   30 CONTINUE
      IF (LNB .EQ. 0) GOTO 90
         LEFT = 1 + (80-(RNB-LNB+1))/2
         RIGHT = 82 - (80-(RNB-LNB+1)+1)/2
         WRITE(6,41) (BORDER(I), I=1,LEFT), (PHRASE(I), I=LNB,RNB),
     $               (BORDER(I), I=RIGHT,82)
   41       FORMAT(1X, 82('.'), /,
     $           3(1X, '.', 80X, '.', /),
     $             1X, 82A1, /,
     $           3(1X, '.', 80X, '.', /),
     $             1X, 82('.') )
         GOTO 20
   90 WRITE(6,91)
   91    FORMAT(1X, 'BLANK CARD READ IN. EXIT FROM CENTER.')
      STOP
      END
```

The actual centering part is now smaller by a factor of three, and that much simpler. As a fringe benefit, it is correct. (This problem is easier in PL/I because variables are permitted in FORMAT statements and because loops can be done zero times. Try it.)

Choosing a better data structure is often an art, which we cannot teach. Often you must write a preliminary draft of the code before you can determine what changes in the data structure will help simplify control. The place to begin such improvements is by asking, "How can I organize the data so the computation becomes as easy as possible?"