```
      DOUBLE PRECISION FUNCTION SIN(X, E)
      DOUBLE PRECISION E, TERM, SUM
      REAL X
C
      SIN = X
      TERM = X
      DO 20 I = 3, 100, 2
         IF (DABS(TERM) .LT. E) GOTO 30
            TERM = -TERM * X**2 / FLOAT(I*(I-1))
            SIN = SIN + TERM
   20 CONTINUE
   30 RETURN
      END
```

In this case, the pseudo-code **WHILE** becomes a DO followed by an IF. The Fortran DO neatly summarizes the initialization, incrementing, and testing of I, and keeps the loop control separate from the computation. It is a useful statement. The important thing is to recognize its shortcomings and plan loops in terms of the more general **WHILE**.

Exercise: determine if we *now* have a working sine routine.

Don't stop at one bug.

Sometimes there are several initialization errors, as in this code:

```
C     CURRENT COMPUTING PROGRAM
C     INPUT VALUES FOR RESISTANCE,FREQUENCY AND INDUCTANCE
      READ(5,20) R,F,L
   20 FORMAT(3F10.4)
C     PRINT VALUES OF RESISTANCE,FREQUENCY AND INDUCTANCE
      WRITE(6,30) R,F,L
   30 FORMAT(3H1R=,F14.4,4H  F=,F14.4,4H  L=,F14.4)
C     INPUT STARTING AND TERMINATING VALUES OF CAPACITANCE,AND INCREMENT
      READ(5,40) SC,TC,CI
   40 FORMAT(3F10.6)
C     SET CAPACITANCE TO STARTING VALUE
      C=SC
C     SET VOLTAGE TO STARTING VALUE
      V=1.0
C     PRINT VALUE OF VOLTAGE
   50 WRITE(6,60) V
   60 FORMAT(3H0V=,F5.0)
C     COMPUTE CURRENT AI
   70 AI = E / SQRT(R**2 + (6.2832*F*L - 1.0/(6.2832*F*C))**2)
C     PRINT VALUES OF CAPACITANCE AND CURRENT
      WRITE(6,80) C,AI
   80 FORMAT(3H0C=,F7.5,4H  I=,F7.5)
C     INCREASE VALUE OF CAPACITANCE
      C = C + CI
      IF (C .LE. TC) GO TO 70
C     INCREASE VALUE OF VOLTAGE
      V = V + 1.0
C     STOP IF VOLTAGE IS GREATER THAN 3.0
      IF (V .LE. 3.0) GO TO 50
      STOP
      END
```