

```

      REAL X(200), Y(200)
      READ 21, N
21  FORMAT(I5)
      IF (1.LE.N .AND. N.LE.200) GOTO 30
      PRINT 23, N
23  FORMAT(1X, 'BAD INPUT COUNT:', I10)
      STOP
30  READ 31, (X(I), Y(I), I=1,N)
31  FORMAT(2F10.5)
      READ 31, A
      IF (A.GE.X(1) .AND. A.LE.X(N)) GOTO 40
      PRINT 33, A, X(1), X(N)
33  FORMAT(1X, F10.5, ' IS OUT OF TABLE RANGE', 2F10.5)
      STOP
40  LOW = 1
      IHIGH = N
50  IF (IHIGH-LOW .LE. 1) GOTO 60
      MID = (IHIGH+LOW)/2
      IF (A.LE.X(MID)) IHIGH = MID
      IF (A.GE.X(MID)) LOW = MID
      GOTO 50
60  IF (A.EQ.X(LOW)) IHIGH = LOW
      IF (A.EQ.X(IHIGH)) LOW = IHIGH
      PRINT 61, X(LOW), Y(LOW), A, X(IHIGH), Y(IHIGH)
61  FORMAT(1X, 5F10.5)
      STOP
      END

```

Our search loop is a DO-WHILE (while  $IHIGH-LOW > 1$ ) that can, under some circumstances, be performed *zero* times. Degenerate cases frequently arise where a piece of code has nothing to do — in this instance, when  $N$  is one or two, no search is necessary. In such cases it is important to “do nothing” gracefully; the DO-WHILE has this useful property.

---

*Make sure your code “does nothing” gracefully.*

---

Fortran programmers should remember that with most Fortran compilers the DO loop is always done once, regardless of its limits; an explicit test is necessary to “do” it zero times. For example, the subroutine INSERT inserts VALUE in the array V at position J. The current size of V is N; INSERT increments this after inserting VALUE.

```

      SUBROUTINE INSERT (V, N, J, VALUE)
      DIMENSION V(80)
      DO 10 I = J, N
          K = N + J - I
          V(K + 1) = V(K)
10  CONTINUE
      V(J) = VALUE
      N = N + 1
      RETURN
      END

```