We also deleted the inaccessible STOP statement and the explicit indexing in the READ statements, indented the code, and numbered the statements systematically.

Don't use conditional branches as a substitute for a logical expression.

As an aside, the dating program provides a simple example of how an appropriate data representation can make programming easier. With INTEGER variables instead of LOGICAL, we can make the desired comparison directly:

```
      INTEGER FEM(8), MALE(8)
      READ (5,10) IGIRL, FEM
10       FORMAT (I5, 8I1)
20    READ (5,10) IBOY, MALE
      DO 30 I = 1, 8
         IF (FEM(I) .NE. MALE(I)) GOTO 20
30    CONTINUE
      ...
```

The data will also have to be changed, from T's and F's to ones and zeros, but this is a simple mechanical operation. We will discuss data structure at more length in Chapter 3.

The expression in parentheses in a logical IF statement is of type LOGICAL; its value is either .TRUE. or .FALSE.. Most of the time we use just a relational operator, such as .LE. or .EQ., to determine the truth value of the condition. But we can, if we wish, use the Boolean operators .AND., .OR., and .NOT. to make arbitrarily complex logical expressions. Boolean algebra is not used nearly as widely as ordinary arithmetic, so we must write logical expressions more carefully lest we confuse the reader.

Consider the sequence

```
 6 IF(X1.GE.ARRAY(I)) GO TO 2
   IF(ARRAY(I).LT.X2) ICOUNT=ICOUNT+1
 2 ...
```

It takes a while to realize that ICOUNT is incremented only if ARRAY(I) lies between X1 and X2. Inversions and GOTO's slow down reading comprehension and should be avoided. Rewriting gives:

```
 6 IF (ARRAY(I).GT.X1 .AND. ARRAY(I).LT.X2) ICOUNT = ICOUNT + 1
```

It is much easier to tell at a glance what the logic implies.

Logical conditions can often be combined *if* they are all related, and if they are combined with only a single type of operator. For example,