```
DO M = 1 TO N-1;
   DO J = 1 TO N-M;
      IF ARAY(J) < ARAY(J+1) THEN DO;
         SAVE = ARAY(J);
         ARAY(J) = ARAY(J+1);
         ARAY(J+1) = SAVE;
      END;
   END;
END;
```

Use the good features of a language;
avoid the bad ones.

A failure to state clearly the underlying logic can lead to tangled control flow, as in this program for a rudimentary computer dating service:

```
   LOGICAL FEM(8),MALE(8)
   READ(5,6)IGIRL,(FEM(I),I=1,8)
 9 READ(5,6)IBOY,(MALE(I),I=1,8)
   DO 8I= 1,8
   IF(FEM(I)) GO TO 7
   IF(.NOT.MALE(I)) GO TO 8
   GO TO 9
 7 IF(.NOT.MALE(I)) GO TO 9
 8 CONTINUE
   WRITE(2,10) IBOY
 6 FORMAT (I5,8L1)
10 FORMAT (10X,I5)
   GO TO 9
   STOP
   END
```

We have to look long and hard at this jungle of IF's and GOTO's before the light dawns. The program is supposed to write IBOY only if each of the MALE(I) has the same truth value as the corresponding FEM(I). Standard Fortran does not allow us to ask directly if two LOGICAL variables are equal or not, but we can still improve readability by using .AND. and .OR.:

```
      LOGICAL FEM(8), MALE(8)
      READ (5,10) IGIRL, FEM
10       FORMAT (I5, 8L1)
20    READ (5,10) IBOY, MALE
      DO 30 I = 1, 8
         IF ((FEM(I) .AND. .NOT.MALE(I)) .OR.
     $        (MALE(I) .AND. .NOT.FEM(I))) GOTO 20
30    CONTINUE
      WRITE (2,40) IBOY
40       FORMAT (10X, I5)
      GOTO 20
      END
```

This tells us directly that the program will go on to read the next input line, without printing IBOY, if any one of the FEM(I) differs from its corresponding MALE(I).