```
SORT: PROCEDURE OPTIONS (MAIN);
   DECLARE (NAMES(50), SPARE) CHARACTER(10);
   DECLARE SWITCH BIT(1);
   DECLARE YES BIT(1) INITIAL ('1'B), NO BIT(1) INITIAL ('0'B);
   DECLARE (I, N) FIXED BINARY;

   GET LIST (NAMES);
   SWITCH = YES;
   DO N = 50 TO 2 BY -1 WHILE (SWITCH = YES);
      SWITCH = NO;
      DO I = 1 TO N-1;
         IF NAMES(I) > NAMES(I+1) THEN DO;
            SWITCH = YES;
            SPARE = NAMES(I);
            NAMES(I) = NAMES(I+1);
            NAMES(I+1) = SPARE;
         END;
      END;
   END;
   PUT LIST (NAMES);
END;
```

The original version used a label and an IF-GOTO to build the outer loop, and a DO for the inner; it was hard to see at a glance that there are truly two loops or where each begins. Now the two loops are explicitly marked as such.

We have also used the variables YES and NO instead of the literals '1'B and '0'B, to make the code read a bit more clearly.

As much as possible, a program should be written so the control flow structures lead the reader quickly and directly to an understanding of what the program does. For example, in

```
A:  IF COUNT(RANK) < 4 THEN
    BEGIN;
      PUT LIST(RECONVERT(RANK));
      COUNT(RANK) = COUNT(RANK) + 1;
      GOTO A;
    END;
```

the construction

```
    IF ... THEN BEGIN ... END
```

is a clear signal that the group of statements between BEGIN and END is to be done exactly once if the condition is true, or not at all if it is false; then, in either case, execution will resume after the END. But look carefully, and you will find that the last statement of the group is a branch back to the test. Although the code claims to be merely an IF, that is a lie — it is actually a loop.

A DO-WHILE provides an honest way to say what the code does:

```
    DO WHILE (COUNT(RANK) < 4);
       PUT LIST(RECONVERT(RANK));
       COUNT(RANK) = COUNT(RANK) + 1;
    END;
```

The advantage is not that the second version is smaller, but that it is explicit. The DO-WHILE says "This is a loop," and is that much easier to understand. The first version forces the reader to ferret out the control flow.