Note, too, that the field(s) specified as the KEY field(s) for sorting purposes must be defined *as part of the sort record format*. In the following batch program excerpt, the field to be sorted is S-DEPT-NO within the SD file called SORT-FILE:

```
 DATA DIVISION.
 FILE SECTION.
 FD  UNSORTED-MASTER-FILE.
 01  UNSORTED-REC                  PIC X(80).
************************************************
 SD  SORT-FILE.  <------------------------------- Note that SORT-FILE is defined with
 01  SORT-REC.                                    an SD and has no LABEL RECORDS
     05  S-DEPT-NO                 PIC XX.        clause
     05                            PIC X(78).
************************************************
 FD  SORTED-MASTER-FILE.
 01  SORTED-REC                    PIC X(80).
```

The SORT procedure would then be coded as follows:

```
SORT  SORT-FILE
    ON ASCENDING KEY S-DEPT-NO  <----Defined within the SD file
    USING UNSORTED-MASTER-FILE
    GIVING SORTED-MASTER-FILE
STOP RUN.
```

The only field descriptions required in the SORT record format are the ones used for sorting purposes. In this instance, only the S-DEPT-NO must be defined as part of the SD, since that is the only key field to be used for sorting.

In summary, the SORTED-MASTER-FILE would contain records with the same format as UNSORTED-MASTER-FILE, but the records would be placed in the sorted master file in department number sequence.

A SORT procedure can also *precede* an update or control break procedure *within the same program*. That is, where a file must be in a specific sequence, we can sort it first and then proceed with the required processing. In this case, the file defined in the GIVING clause would be opened as input, after it has been created as a sorted file:

```
                     PROCEDURE DIVISION.
                     100-MAIN-MODULE.
                         SORT SORT-FILE
                              ON ASCENDING KEY TERR
                                 USING UNSORTED-MASTER-FILE
                                 GIVING SORTED-MASTER-FILE
                    {    OPEN INPUT SORTED-MASTER-FILE
                    |         OUTPUT CONTROL-REPORT
                    |    PERFORM UNTIL ARE-THERE-MORE-RECORDS = 'NO '
                    |        READ SORTED-MASTER-FILE
                    |            AT END
Standard processing {                MOVE 'NO ' TO ARE-THERE-MORE-RECORDS
                    |            NOT AT END
                    |                PERFORM 200-PROCESS-RTN
                    |        END-READ
                    |    END-PERFORM
                    |        .
                    {        .
```

SELF-TEST

1. Suppose we want EMPLOYEE-FILE records in alphabetic order by NAME within DISTRICT within TERRITORY, all in ascending sequence. The output file is called SORTED-EMPLOYEE-FILE. Complete the following SORT statement:

   ```
   SORT  WORK-FILE ...
   ```

2. How many files are required in a simple SORT routine? Describe these files.
3. The work or sort file is defined as an ________ in the DATA DIVISION.
4. Suppose we have an FD called NET-FILE-IN, an SD called NET-FILE, and an FD called NET-FILE-OUT. We want NET-FILE-OUT sorted into ascending DEPT-NO sequence. Code the PROCEDURE DIVISION entry.
5. In Question 4, DEPT-NO must be a field defined within the (SD/FD) file.

blank fields, (3) remove unneeded fields from the input records, and (4) count input records.

Example 1 We will code a SORT routine that eliminates records with a quantity field equal to zero *before sorting*. The test for zero quantity will be performed in an INPUT PROCEDURE. Consider the first three DIVISIONs of the COBOL program:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SORT-IT.
*
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT IN-FILE ASSIGN TO 'C:\CHAPTER14\DISK1.DAT'.
     SELECT SORT-FILE ASSIGN TO 'C:\CHAPTER14\WORK1.DAT'.
     SELECT SORTED-MSTR ASSIGN TO 'C:\CHAPTER14\DISK2.DAT'.
*
 DATA DIVISION.
 FILE SECTION.
 FD  IN-FILE.
 01  IN-REC.
     05               PIC X(25).
     05   QTY         PIC 9(5).  ◄——Needed for INPUT PROCEDURE section
     05               PIC X(70).
 SD  SORT-FILE.
 01  SORT-REC.
     05   TERR        PIC X(5).  ◄——Needed for ASCENDING KEY clause
     05               PIC X(95).
 FD  SORTED-MSTR.
 01  SORTED-MSTR-REC PIC X(100).
```

With the newest version of COBOL, procedure-names used with INPUT PROCEDURE can be regular paragraphs. Thus, we can code:

```
100-MAIN-MODULE.
    SORT  SORT-FILE
          ON ASCENDING KEY TERR
             INPUT PROCEDURE 200-TEST-IT
             GIVING SORTED-MSTR
    STOP RUN.
200-TEST-IT.
    OPEN INPUT IN-FILE
    PERFORM UNTIL ARE-THERE-MORE-RECORDS = 'NO '
        READ IN-FILE
            AT END
                MOVE 'NO ' TO ARE-THERE-MORE-RECORDS
            NOT AT END
                PERFORM 300-PROCESS-RTN
        END-READ
    END-PERFORM
    CLOSE IN-FILE.
300-PROCESS-RTN.
    IF  QTY = ZEROS
        CONTINUE
    ELSE
        MOVE IN-REC TO SORT-REC
        RELEASE SORT-REC ◄———Writes the record onto the sort file
    END-IF.
```

The 200-TEST-IT paragraph must:

1. Open the input file. (With a USING option instead of the INPUT PROCEDURE, the input file is automatically opened by the SORT verb.)
2. Perform some processing of input records until there is no more data.
3. Close the input file.