UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

A general purpose ANSI FORTRAN-IV subroutine system for sorting data

by

Gerald Ian Evenden

Open-File Report 76-306

1976

This report is preliminary and has not been edited or reviewed for conformity with U.S. Geological Survey standards.

Contents

Abstract

A system of ANSI FORTRAN subroutines provides FORTRAN programmers with a transportable data sorting system. Flexibility is provided through user written input, output and comparison routines.

Disclaimer.

Although these subroutines have been subjected to many tests and considerable usage, a warranty on accuracy or proper functioning is neither implied nor expressed.

Introduction.

The lack of a standard, general purpose, intrinsic sorting function is a recurrent problem in FORTRAN programming. Although some manufacturers and computer centers supply FORTRAN callable sorting routines, they frequently lack transportability because of their proprietary status or their use of machine-dependent features. This problem has created an unnecessary burden on programmers faced with trying to create transportable software and involved with conversion of software from one computer to another.

The sort-system presented here is coded in ANSI FORTRAN-IV and represents a transportable, yet flexible method of providing the sorting function to FORTRAN programs. The principle deficiency of the routine is that it obviously cannot (or, at least, should not) perform as well as well coded, machine taylored software.

General description of the sorting system.

Basic versatility of this sorting system is provided by the user preparing the input, output and comparison routines and passing them as arguments to the sorting system. Complexity of input and output editing, data types and sort key structure is limited only by the user's ability to define his problem. The sorting system's function is limited to controlling appropriate execution of these routines and auxiliary scratch files. It should be noted that "input" and "output" only refer to transferal of the data between the user and the sort system and need not involve input or output with peripheral devices. The only other requirements of the user is to provide the sorting system with work storage and data record length. This method is similar to that employed by COBOL (except for sort key specification and working storage) and Burroughs' (1968) extended ALGOL sort procedure.

Subroutine SORT (user entry point) is the main control segment of the sorting system. User parameters are checked, the work storage is divided into working units (through internal variables IAS, IAE, IOS, and IOE) and the sorting and merging phases are executed by respective calls to routines SORTA and SORTB. Description of the SORT input parameters is contained in the comments of subroutine SORT listed in Appendix 2 and an example calling program is provided in Appendix 1.

Subroutine SORTA inputs the data to the sorting system by repeatedly calling the user input routine until the user informs the system that the end of data has occurred. If the work area is filled prior to an end of data condition, the data is sorted in memory by a call to SORTD and output as a logical sorted block to the scratch files and the system is informed that a merging phase must follow the sorting phase. If, however, the end of data condition occurs prior to scratch file output the sorted data is returned directly to the user's output routine and the sort operation is terminated without the scratch files and merge phase being used.

Innumerable methods are available for memory sorting. My timing trials of some of the methods analyzed and translated to FORTRAN by Loeser (1974) indicated that QUICKERSORT by Scowen (1965) was a reasonable choice. This decision was also tempered by QUICKERSORT's lack of non-intrinsic function requirements (such as a random number generator) and the non-objective reason of minimum code. SORTD (Loeser's routine SHORT) and KSORT were necessarily modified to eliminate Loeser's diagnostic counters and to include the usage of the user-comparison routine and employment of the record index array.

Of the several merging methods described in Sorting Techniques (IBM, 1965) the balanced merge (in subroutine SORTB) was chosen on the grounds of overall simplicity.

Balanced merging requires 4 (NBC=2) or more scratch files with a current limitation of 14 (NBC=7). The upper limit can easily be changed by altering the dimensions of vector IA in routine SORTB, vectors IAS, IAE, NOUT, NIN, NINB in the common area SORTC, and parameter NBMAX data statement value in routine SORT to a value of one-half the maximum number of files allowed. Users selection of the number of scratch files is a choice between increased memory requirements of a large NBC (for I/O buffer areas) and a decreased number of merge passes.

The subroutines SORTE, SORTF, SORTG and SORTH provide for scratch file I/O handling. In general, this is the weakest part of the sorting system. Because I/O control provided by ANSI FORTRAN is primitive, improvements can often be made in these routine when manufacturer's extended FORTRAN I/O statements are available. Subroutine SORTE handles merge phase switching of input/output status of each scratch file. Subroutines SORTF and SORTG are employed for scratch input and SORTH for scratch output.

Conclusion.

This sorting system has performed well with small to medium volumes of data and has reduced the problem of sorting in one application to a trivial portion of the job. Although this implementation of sorting in FORTRAN is not the state of the art of sorting, the method of execution is considered an extremely viable technique for overall flexibility. In addition, the system provides complete transportability between computers with ANSI FORTRAN-IV compilers.

It is hoped that groups involved with language specifications will recognize that sorting is a requirement of all types of computer programming and that an effort will be made in the future to make sort statements an intrinsic part of all languages.

References.

Burroughs, 1968, Extended ALGOL reference manual: Burroughs Corporation, p. 12-1 - 12-5.

IBM, 1965, Sorting techniques: International Business Machines Corporation, 100 p.

Loeser, Rudolph, 1974, Some performance tests of "QUICKERSORT" and descendants: Comm. ACM, v. 17, no. 3, pp. 143 - 152.

Scowen, R. S., 1965, Algorith 271, QUICKERSORT: Comm. ACM, v. 8, no. 11, p.669-670.

Appendix 1.

Example program using the sorting system.

The following program and subroutines provides a simple example of execution of the sorting system. Two word records of sequence numbers and random numbers is created in function IN. This sequence is ordered by ascending value of the random numbers as a primary key and descending sequence numbers as the secondary key. The results of the sort program is printed by the output routine OUT.

```
C
C  EXAMPLE OF SORT EXECUTION.
      DIMENSION WORK(30)
      EXTERNAL IN,OUT,ICOMP
      CALL SORT(IN,OUT,ICOMP,WORK,30,2,2,IER)
      PRINT 10,IER
   10 FORMAT(' END IER=',I10)
      STOP
      END
C
      FUNCTION IN(IREC)
      DIMENSION IREC(2)
      DATA N/1/
      IF (N.GT.20) GO TO 20
      IREC(1)=N
      IREC(2)=RAN(J)*10.
      N=N+1
      IN=1
      PRINT 10,IREC
   10 FORMAT(2I10)
      RETURN
   20 IN=0
      PRINT 30
   30 FORMAT(' END OF INPUT')
      RETURN
      END
C
      SUBROUTINE OUT(IREC,K)
      DIMENSION IREC(2)
      IF (K) 20,30,20
   10 FORMAT(2I10)
   20 PRINT 10,IREC
      GO TO 50
```

```
   30 PRINT 40
   40 FORMAT(' END OUTPUT')
   50 RETURN
      END
C
      FUNCTION ICOMP(I,J)
      DIMENSION I(2),J(2)
      II=I(2)-J(2)
      IF (II) 20,10,20
   10 II=J(1)-I(1)
   20 ICOMP=II
      RETURN
      END
```

The results of the print statements follows:

```
          1          1
          2          7
          3          6
          4          3
          5          1
          6          1
          7          0
          8          7
          9          6
         10          0
         11          6
         12          3
         13          3
         14          5
         15          8
         16          4
         17          7
         18          4
         19          0
         20          1
 END OF INPUT
         19          0
         10          0
          7          0
         20          1
          6          1
          5          1
          1          1
         13          3
         12          3
          4          3
         18          4
         16          4
         14          5
         11          6
          9          6
          3          6
         17          7
          8          7
          2          7
         15          8
  END OUTPUT
  END IER=           0
```

Appendix 2.

Listing of sorting system.

```
C
C-----------------------------------------------------------------
C
C  SUBROUTINE SORT
C
C  PURPOSE..
C    TO PROVIDE GENERAL, HIGHLY TRANSPORTABLE SORTING
C    CAPABILITY FOR FORTRAN PROGRAMS.
C
C  USAGE..
C    CALL SORT(INPUT,OUTPT,ICOMP,W,NW,NBC,NWPRC,IER)
C
C  PARAMETER DESCRIPTION..
C    INPUT  - USER SUPPLIED INTEGER FUNCTION WHICH RETURNS
C             INPUT RECORD ARRAY IN THE DUMMY ARGUMENT.
C             FUNCTIONAL VALUE IS NON-ZERO WHEN VALID DATA
C             RECORD RETURNED AND ZERO WHEN DATA
C             ABSENT AND NO MORE RECORDS TO BE INPUT.
C    OUTPT  - USER SUPPLIED SUBROUTINE TO ACCEPT SORTED
C             OUTPUT RECORDS IN THE FIRST ARRAY ARGUMENT.
C             THE SECOND INTEGER ARGUMENT CONTAINS A ONE
C             WHEN DATA PRESENT AND A ZERO WHEN DATA
C             ABSENT AND PREVIOUS CALL CONTAINS LAST RECORD.
C    ICOMP  - USER SUPPLIED INTEGER FUNCTION WHICH
C             DETERMINES THE PROPER ORDER OF
C             TWO INPUT DATA RECORD ARRAY ARGUMENTS.
C             ICOMP.LT.0 WHEN 1ST ARG SHOULD PRECEED 2ND.
C             ICOMP.EQ.0 FOR NO DIFFERENCE IN ORDER.
C             ICOMP.GT.0 WHEN 2ND ARG SHOULD PRECEED 1ST.
C    W      - WORK ARRAY OF NW WORDS.
C    NW     - LENGTH OF WORK ARRAY.
C    NBC    - NUMBER OF WORK OUTPUT FILES TO BE EMPLOYED.
C             NOTE.. 2*NBC FILES WILL BE USED FOR MERGING.
C             RESTRICTION.. 2.LE.NBC.LE.7
C    NWPRC  - NUMBER OF WORDS IN DATA RECORDS.
C    IER    - ERROR RETURN CODE.
C             =0 NO ERRORS.
C             .GT.0 PATHOLOGIC PROBLEMS.
C             =-1 NBC OUT OF RANGE.
C             =-2 NWPRC.LT.1
C             =-3 NW TOO SMALL, MORE WORK AREA REQUIRED.
C
C  REMARKS..
C    THIS SYSTEM WILL EMPLOY FORTRAN LOGICAL UNIT NUMBERS
C    15 THROUGH 15+2*NB-1.  THE USER MUST NOT USE THESE
C    LOGICAL UNIT NUMBERS DURING EXECUTION OF THIS SYSTEM.
C
C  SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED..
C    SORTA,SORTB,SORTD,SORTE,SORTF,SORTG,SORTH,KSORT,
```

```
C     AND NAMED COMMON AREA SORTC.
C
C   CODED BY..
C     GERALD IAN EVENDEN
C     U. S. GEOLOGICAL SURVEY
C     DENVER FEDERAL CENTER
C     DENVER, COLORADO  80225
C
C------------------------------------------------------------
C
      SUBROUTINE SORT(INPUT,OUTPT,ICOMP,W,NW,
     1               NBC,NWPRC,IER)
C
C   NOTE THAT SINCE INPUT, OUTPT AND COMP ARE
C   NOT EMPLOYED IN SORT THEY ARE DECLARED AS ARRAYS SO THAT
C   THEY WILL BE PASSED BY NAME (ADDRESS) TO THE
C   THE SUBROUTINES SORTA AND SORTB.
C
      DIMENSION INPUT(1),OUTPT(1),ICOMP(1),W(1)
C
      COMMON /SORTC/
     1 IAS(7),IAE(7),NWPR,NWPR1,NB,NRPS,IOS,IOE,LASTP,
     2  NWPB,NOUT(7),NIN(7),NINB(7),KOUT,KOUTL,KIN,
     3  NBPCO,NBPCI,INLB,IOB,IOUT,NWOL,NWLB
C
      EQUIVALENCE (N,NBPCO)
      DATA NBMAX/7/
C
C   CHECK SOME INPUT VARIABLES.
      NB=NBC
      IF (NB.GT.1.AND.NB.LE.NBMAX) GO TO 10
      IER=-1
      GO TO 70
   10 IF (NWPRC.GT.0) GO TO 20
      IER=-2
      GO TO 70
   20 NWPR=NWPRC
      NWPB=NW/(NB+1)
      NRPB=NWPB/NWPR
      IF (NRPB.GT.0) GO TO 30
      IER=-3
      GO TO 70
   30 IER=0
      NWPB=NRPB*NWPR
C
C   LAYOUT MEMORY USAGE
      IOS=NW-NWPB+1
      IOE=NW
      IAS(1)=1
      IAE(1)=NWPB
      DO 40 I=2,NB
        IAS(I)=IAS(I-1)+NWPB
        IAE(I)=IAE(I-1)+NWPB
   40   CONTINUE
```

```
      N=NB-1
      I=N*NRPB
   50 IF (IOS-IAE(N).GT.I) GO TO 60
      N=N-1
      I=I-NRPB
      GO TO 50
   60 NRPS=N*NRPB
      NWPR1=NWPR-1
C
C  BLOCK SORTING PHASE
      LASTP=1
      CALL SORTA(W(IAE(N)+1),W,INPUT,OUTPT,ICOMP,IER)
      IF (IER.NE.0.OR.LASTP.LT.0) GO TO 70
C
C  MERGING PHASE REQUIRED
      CALL SORTB(W,OUTPT,ICOMP)
C
C  ALL DONE
   70 RETURN
      END
```

```
C
C  INPUT AND SORTING PHASE.
C
      SUBROUTINE SORTA(IP,W,INPUT,OUTPT,ICOMP,IER)
      DIMENSION IP(1),W(1),ICOMP(1)
C
      COMMON /SORTC/
     1 IAS(7),IAE(7),NWPR,NWPR1,NB,NRPS,IOS,IOE,LASTP,
     2  NWPB,NOUT(7),NIN(7),NINB(7),KOUT,KOUTL,KIN,
     3  NBPCO,NBPCI,INLB,IOB,IOUT,NWOL,NWLB
C
   10 IEO=1
C
C  GET INPUT RECORDS AND SET TAG ARRAY.
      J=1
      DO 40 I=1,NRPS
        IF (INPUT(W(J))) 30,20,30
   20   N=I-1
        IF (N) 80,80,50
   30   IP(I)=J
        J=J+NWPR
   40   CONTINUE
C
C  FULL INPUT
      N=NRPS
      IEO=0
C
C  SORT BLOCK(S)
   50 CALL SORTD(IP,N,ICOMP,W,W(IOS),NWPB,IER)
      IF (IER.NE.0) RETURN
C
C  CHECK IF ALL DONE BEFORE SCRATCH WRITTEN
      IF (IEO.NE.0.AND.LASTP.GT.0) GO TO 90
      IF (LASTP.GT.0) CALL SORTE
C
C  MOVE SORTED BLOCK(S) TO OUTPUT
      IO=IOS
      DO 70 I=1,N
        K1=IP(I)
        K2=K1+NWPR1
        DO 60 K=K1,K2
          W(IO)=W(K)
          IO=IO+1
   60     CONTINUE
        IF (IO.LT.IOE) GO TO 70
        CALL SORTH(W(IOS),NWPB)
        IO=IOS
   70   CONTINUE
      IF (IEO.EQ.0) GO TO 10
      IO=IO-IOS
      IF (IO.GT.0) CALL SORTH(W(IOS),IO)
   80 RETURN
C
C  ALL INPUT COMPLETED PRIOR TO SCRATCH FILE DUMP
```

```
C
C  SO JUST PASS IT BACK TO USER.
   90 DO 100 I=1,N
        CALL OUTPT(W(IP(I)),1)
  100   CONTINUE
      LASTP=-1
C
C  TELL USER OUTPUT ROUTINE WE'RE DONE
      CALL OUTPT(W,0)
      RETURN
      END
```

```
C
C   THIS ROUTINE PERFORMS BALANCED MERGING
C   OF THE SCRATCH FILE DATA.
C
      SUBROUTINE SORTB(W,OUTPT,ICOMP)
      DIMENSION W(1),IA(7)
C
      COMMON /SORTC/
     1 IAS(7),IAE(7),NWPR,NWPR1,NB,NRPS,IOS,IOE,LASTP,
     2  NWPB,NOUT(7),NIN(7),NINB(7),KOUT,KOUTL,KIN,
     3  NBPCO,NBPCI,INLB,IOB,IOUT,NWOL,NWLB
C
C   SWITCH FILES AND LOAD WORK AREA.
   10 CALL SORTE
      DO 30 I=1,NB
        CALL SORTF(I,W(IAS(I)),N)
        IF (N.GT.0) GO TO 20
        IA(I)=0
        GO TO 30
   20   IA(I)=IAS(I)
        IAE(I)=IA(I)+N-1
   30   CONTINUE
      IO=IOS
C
C   NOTE THAT IA SERVES AS BOTH AN INDEX AND A FLAG VALUE.
C     IA(N).GT.0 THEN INDEX TO CURRENT INPUT FILE RECORD.
C     IA(N).EQ.0 THEN INPUT FILE EMPTY (EOF).
C     IA(N).LT.0 THEN NO MORE DATA IN CURRENT SORT BLOCK.
C      ABS(IA(BN)) IS STARTING ADDRESS FOR NEXT BLOCK CYCLE.
C
C   FIND WINNER AMONG INPUT BLOCKS.
   40 LWINF=0
      DO 70 I=1,NB
        K=IA(I)
        IF (K) 70,70,50
   50   IF (LWINF.EQ.0) GO TO 60
        IF (ICOMP(W(K),W(LWIN))) 60,70,70
   60   LWIN=K
        LWINF=I
   70   CONTINUE
C
C   CHECK IF ANY DATA.
      IF (LWINF.LE.0) GO TO 160
C
C   YES.  GOT WINNER, PUT IN OUTPUT
      IF (LASTP) 80,90,90
C
C   LAST PHASE OUTPUT TO USER ROUTINE
   80 CALL OUTPT(W(LWIN),1)
      LWIN=LWIN+NWPR
      GO TO 110
C
C   NOT LAST PHASE, SO PUT TO SCRATCH
   90 IO2=IO+NWPR1
```

```
      DO 100 I=IO,IO2
        W(I)=W(LWIN)
  100   LWIN=LWIN+1
      IO=IO2+1
C
C  CHECK IF AREA FULL.
      IF (IO.LT.IOE) GO TO 110
      CALL SORTH(W(IOS),NWPB)
      IO=IOS
C
C  UPDATE POINTERS
  110 IF (LWIN.GT.IAE(LWINF)) GO TO 120
      IA(LWINF)=LWIN
      GO TO 40
C
C  INPUT BLOCK EMPTY
  120 I=IAS(LWINF)
      CALL SORTF(LWINF,W(I),N)
      IF (N) 140,130,150
  130 IA(LWINF)=0
      GO TO 40
  140 IA(LWINF)=-I
      IAE(LWINF)=I-N-1
      GO TO 40
  150 IA(LWINF)=I
      IAE(LWINF)=I+N-1
      GO TO 40
C
C  DONE WITH BLOCK CYCLE, CHECK FOR NEXT CYCLE
  160 IF (IA(1).EQ.0) GO TO 180
      DO 170 I=1,NB
  170 IA(I)=-IA(I)
      GO TO 40
C
C  DONE WITH PHASE
  180 IF (LASTP) 210,190,190
C
C  DUMP REMAINING RECORD, IF ANY
  190 I=IO-IOS
      IF (I) 10,10,200
  200 CALL SORTH(W(IOS),I)
      GO TO 10
C
C  TELL CALL SYSTEM THAT JOB IS DONE.
  210 CALL SORTE
      CALL OUTPT(W,0)
C
C  QUIT
      RETURN
      END
```

```
C
C-----------------------------------------------------------------
C
C  SUBROUTINE SORTD
C
C  PURPOSE-
C       TO SORT A VECTOR A IN TO A SEQUENCE DETERMINED BY
C       A USER SUPPLIED FUNCTION SUBPROGRAM COMP.  THE
C       VECTOR A MAY BE EITHER IN THE SORT OR A TAG
C       (INDEX) TO THE KEY VECTORS.
C
C  USAGE-
C       CALL SORTD(A,N,COMP,AUX,IWORK,NWORK,IER)
C
C  DESCRIPTION OF PARAMETERS-
C    A     - ON ENTRY N WORDS IN ANY ORDER.
C            ON RETURN N WORDS SORTED IN SEQUENCE DETERMINED
C            BY COMP.
C    N     - DIMENSION OF VECTOR A.
C    COMP  - SEQUENCE FUNCTION SUBROUTINE WHICH IS
C            CALLED BY KSORT AS
C             F=COMP(A(I),A(J),AUX).
C             WHEN--
C             COMP.LT.0  A(I) LOWER IN SEQUENCE THAN A(J)
C             COMP.EQ.0  A(I) IDENTICAL IN SEQUENCE TO A(J)
C             COMP.GT.0  A(I) HIGHER IN SEQUENCE THAN A(J)
C    AUX   - AUXILLIARY VECTOR PASSED TO COMP.
C    IWORK - WORK VECTOR
C    NWORK - DIMENSION OF VECTOR IWORK.  SHOULD BE NOT
C            LESS THAN 2*LOG2(N).
C    IER   - RESULTANT ERROR PARAMETER CODED AS FOLLOWS
C            IER.EQ.0  NO ERRORS
C            IER.EQ.1  DIMENSION OF IWORK TOO SMALL.
C
C  SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED-
C       KSORT - SORT PARTITIONING ROUTINE.
C         MAY BE EITHER QSORT, QUICKSORT OR QUICKERSORT.
C       COMP   - MUST BE SUPPLIED BY USER.  NOTE THAT COMP
C                 MUST BE DECLARED EXTERNAL IN CALLING
C                 PROGRAM.
C
C  METHOD/REFERENCE-
C       LOESER, RUDOLF, 1974, SOME PERFORMANCE TESTS OF
C          QUICKSORT AND DESCENDANTS, COMM. ACM, V. 17,
C          NO. 3, P. 143-152.
C    MODIFIED BY-
C       EVENDEN, G. I., 1974, U. S. GEOLOGICAL SURVEY,
C        DENVER, CO.  80225
C
C-----------------------------------------------------------------
C
      SUBROUTINE SORTD(A,N,COMP,AUX,IWORK,NWORK,IER)
C
      DIMENSION A(1),COMP(1),AUX(1),IWORK(1)
```

```
C
      IER=0
      IF (N-1) 90,90,10
   10 J=NWORK-2
      M=0
      LL1=1
      LU1=N
   20 IF (LU1-LL1) 70,70,30
   30 IF (KSORT(A,LL1,LU1,LL,LU,COMP,AUX)) 70,70,40
   40 IF (M-J) 60,60,50
   50 IER=1
      GO TO 90
   60 M=M+2
      IWORK(M-1)=LL
      IWORK(M)=LU
      GO TO 20
   70 IF (M) 90,90,80
   80 LL1=IWORK(M-1)
      LU1=IWORK(M)
      M=M-2
      GO TO 20
   90 RETURN
      END
```

```
C
C-------------------------------------------------------------------
C
C  SUBROUTINE KSORT
C
C  PURPOSE-
C       SPLITTING SUBROUTINE FOR SORTD
C       BASED ON THE QUICKERSORT ALGORITHM.
C
C  SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED-
C       COMP
C
C  REMARKS-
C       SEE SORTD AND REFERENCES FOR DETAILS
C
C  METHOD/REFERENCE-
C       SCOWEN, R. S.,1965,ALGORITHM 271-QUICKERSORT, COMM.
C       ACM, V. 8, NO. 11, P. 669-670.
C    TRANSLATED TO FORTRAN BY-
C       LOESER, RUDOLF, 1974, SOME PERFORMANCE TESTS OF
C       QUICKSORT AND DESCENDANTS, COMM. ACM, V. 17, NO. 3,
C       P. 143-152.
C    MODIFIED BY-
C       EVENDEN, G. I., 1974, U. S. GEOLOGICAL SURVEY,
C       DENVER, CO.  80225
C
C-------------------------------------------------------------------
C
      FUNCTION KSORT(A,LL1,LU1,LL,LU,COMP,AUX)
      INTEGER A,T,X,COMP
C
      DIMENSION A(1),AUX(1)
C
      IF (LU1-LL1-1) 10,30,50
   10 KSORT=0
   20 RETURN
   30 IF (COMP(AUX(A(LL1)),AUX(A(LU1)))) 10,10,40
   40 X=A(LL1)
      A(LL1)=A(LU1)
      A(LU1)=X
      GO TO 10
   50 KSORT=1
      IP=(LL1+LU1)/2
      T=A(IP)
      A(IP)=A(LL1)
      IQ=LU1
      K=LL1
   60 K=K+1
      IF (K-IQ) 70,70,120
   70 IF (COMP(AUX(A(K)),AUX(T))) 60,60,80
   80 IF (IQ-K) 120,90,90
   90 IF (COMP(AUX(A(IQ)),AUX(T))) 110,100,100
  100 IQ=IQ-1
      GO TO 80
```

```
  110 X=A(K)
      A(K)=A(IQ)
      A(IQ)=X
      IQ=IQ-1
      GO TO 60
  120 A(LL1)=A(IQ)
      A(IQ)=T
      IF ((IQ+IQ)-(LL1+LU1)) 140,140,130
  130 LL=LL1
      LU=IQ-1
      LL1=IQ+1
      GO TO 20
  140 LL=IQ+1
      LU=LU1
      LU1=IQ-1
      GO TO 20
      END
```

```
C
C   SCRATCH FILE CONTROL FOR MERGING.
C
      SUBROUTINE SORTE
C
      COMMON /SORTC/
     1 IAS(7),IAE(7),NWPR,NWPR1,NB,NRPS,IOS,IOE,LASTP,
     2  NWPB,NOUT(7),NIN(7),NINB(7),KOUT,KOUTL,KIN,
     3  NBPCO,NBPCI,INLB,IOB,IOUT,NWOL,NWLB
C
      DATA KOUTS,KD/15,14/
      IF (LASTP) 50,30,10
C
C   INITIALIZATION CALL.
   10 KOUT=KOUTS
      KIN=KOUT+NB
      KOUTL=KIN-1
      DO 20 I=1,NB
        NOUT(I)=0
   20   CONTINUE
      IOB=0
      IOUT=KOUT
      LASTP=0
      RETURN
C
C   INTERMEDIATE PHASE
   30 M=KIN
      N=KOUT
      IF (NOUT(1).LE.NBPCO) LASTP=-1
      DO 40 I=1,NB
        REWIND M
        REWIND N
        M=M+1
        N=N+1
        NINB(I)=0
        NIN(I)=NOUT(I)
        NOUT(I)=0
   40   CONTINUE
      NWLB=NWOL
      INLB=IOUT
      M=KIN
      KIN=KOUT
      KOUT=M
      IOUT=KOUT
      IOB=0
      NBPCI=NBPCO  -
      NBPCO=NBPCO*NB
      IOUT=KOUT
      KOUTL=KOUT+NB-1
      RETURN

   FINAL CLOSE OUT PHASE
   50 M=KIN
      DO 60 I=1,NB
```

```
      REWIND M
      M=M+1
 60   CONTINUE
    RETURN
C
    END
```

```
C
C  SCRATCH  FILE INPUT ROUTINE FOR MERGING
C
      SUBROUTINE SORTF(IN,W,NW)
      DIMENSION W(1)
      COMMON /SORTC/
     1 IAS(7),IAE(7),NWPR,NWPR1,NB,NRPS,IOS,IOE,LASTP,
     2  NWPB,NOUT(7),NIN(7),NINB(7),KOUT,KOUTL,KIN,
     3  NBPCO,NBPCI,INLB,IOB,IOUT,NWOL,NWLB
C
C
      N=NIN(IN)
      IF (N.LE.0) GO TO 40
      KF=IN+KIN-1
      IF ((N.GT.1).OR.(KF.NE.INLB)) GO TO 10
      NW=NWLB
      GO TO 20
   10 NW=NWPB
   20 CALL SORTG(KF,W,NW)
      NIN(IN)=N-1
      N=NINB(IN)+1
      IF (N.LE.NBPCI) GO TO 30
      N=1
      NW=-NW
   30 NINB(IN)=N
      RETURN
C
   40 NW=0
      RETURN
      END
```

```
C
C  SUPPORT ROUTINE FOR SORTF.
C
C
C  THIS SEEMINGLY UNNEEDED ROUTINE IS REQUIRED TO
C  OPTIMIZE POOR FORTRAN CODE FREQUENTLY GENERATED
C  BY IMPLIED DO LOOP LISTS.
      SUBROUTINE SORTG(IN,W,N)
      DIMENSION W(N)
      READ(IN) W
      RETURN
      END
```

```
C
C   OUTPUT ROUTINE FOR MERGE PHASE FILES.
C
      SUBROUTINE SORTH(W,N)
      DIMENSION W(N)
      COMMON /SORTC/
     1 IAS(7),IAE(7),NWPR,NWPR1,NB,NRPS,IOS,IOE,LASTP,
     2  NWPB,NOUT(7),NIN(7),NINB(7),KOUT,KOUTL,KIN,
     3  NBPCO,NBPCI,INLB,IOB,IOUT,NWOL,NWLB
C
C
C   BUMP SECTION BLOCK COUNT
      IOB=IOB+1
      IF (IOB.LE.NBPCO) GO TO 10
C
C   DONE WITH SECTION, BUMP FILE
      IOB=1
      IOUT=IOUT+1
      IF (IOUT.GT.KOUTL) IOUT=KOUT
C
C   BUMP FILE RECORD COUNTER
   10 I=IOUT-KOUT+1
      NOUT(I)=NOUT(I)+1
      NWOL=N
      WRITE(IOUT) W
      RETURN
      END
```