1 SUMMARY

To **sort a table of fixed length records** into ascending or descending order. The subroutine sorts on a **key field** defined within each record and returns an index to the required ordering so as to leave the records in their original order and unchanged.

The key field is assumed to contain numeric information and the length of the key field and its position in each record must be the same for every record being sorted during each call. The user must initialize the index array to specify which records in the table are to be sorted. The index is permuted by the subroutine to indicate the required ascending and descending order.

The subroutine uses a stable sort (preserves the order of two records with identical key fields), so the user may subsort on several different fields by consecutive calls. Note that as the record is not moved the key field is allowed to lie outside the boundary of the record.

The method used is the two-way merge, see Knuth, 'The Art of Computer Programming', Vol 3, pp 160.

ATTRIBUTES — Version: 1.0.0. **Types:** `KB09A, KB09AD, KB09AI.` **Original date:** April 1980. **Origin:** C.Birch*, Harwell.

2 HOW TO USE THE PACKAGE

2.1 The argument list and calling sequence

The single precision version

```
CALL KB09A(TAB,INDX,NN,N,KPOS,KLEN,LREC,IWORK,IND)
```

The double precision version

```
CALL KB09AD(TAB,INDX,NN,N,KPOS,KLEN,LREC,IWORK,IND)
```

The integer version

```
CALL KB09AI(TAB,INDX,NN,N,KPOS,KLEN,LREC,IWORK,IND)
```

`TAB` is a two-dimensional `REAL` (`DOUBLE PRECISION` in the D version or `INTEGER` in the I version) array of dimensions `LREC` by `NN.` It must be set by the user to contain the table of records to be sorted. This argument is not altered by the subroutine.

`INDX` is an `INTEGER` array at least of length `N` which must be set by the user to point to the records in `TAB` which are to be sorted. Setting `INDX(I)=J` specifies that the `J`th record in `TAB` is to be included in the sort and has initial position `I.` If every record in `TAB` is to be included in the sort set `N=NN` and set `INDX(I)=I, I=1,N.` On return from the subroutine the contents of `INDX` will give the required ordering, i.e. the `I`th record in the final order will be located as the `INDX(I)`th record in `TAB.` If subsorting on several key fields is being done the contents of `INDX` should not be altered by the user between calls and the premier fields should be sorted first.

`NN` is an `INTEGER` variable which must be set by the user to the total number of records in `TAB.` This argument is not altered by the subroutine.

`N` is an `INTEGER` variable which must be set by the user to the number of records which have been selected from the total to be sorted. The index array `INDX` should specify which records have been selected. This argument is not altered by the subroutine.

`KEYPOS` is an `INTEGER` variable which must be set by the user to the position of the key field in the record, e.g. if `TAB` is `REAL` and the key field begins with the fourth word in each record then `KEYPOS=4.` This argument is not altered by the subroutine.

`KLEN` is an `INTEGER` variable which must be set by the user to the length of the key field (in words), e.g. if the table is to be sorted on a key field which is one word (the normal case) set `KLEN=1.` If the key length is greater than one the sort is done primarily on the first number and equalities are resolved by bringing in the second and third, etc., into the comparisons. This argument is not altered by the subroutine.

`LREC` is an `INTEGER` variable which must be set by the user to the length of each record (in words). It is also used as the first dimension of the array `TAB.` This argument is not altered by the subroutine.

`IWORK` ia an `INTEGER` array of length at least `N` which is used by the subroutine as workspace.

`IND` is a `LOGICAL` variable which must be set by the user to indicate the direction of the sort. The two options are

`.TRUE.`	sort into ascending order.
`.FALSE.`	sort into descending order.

This argument is not altered by the subroutine.

2.2 The common area and error messages

The subroutine uses a common area which the user may also reference. To do this the calling program should contain a common statement of the form

The single precision version

```
COMMON/KB09C/ LP,MERR
```

The double precision version

```
COMMON/KB09CD/ LP,MERR
```

The integer version

```
COMMON/KB09CI/ LP,MERR
```

`LP` is an `INTEGER` variable which specifies the Fortran stream number to be used for error messages. The default value is 6 (line printer). To suppress the printing of error messages set `LP` ≤ 0.

`MERR` is an `INTEGER` variable which is set by the subroutine to indicate success or failure. On return from the subroutine `MERR` is one of two values

0	successful sort
1	an error prevented sorting from being completed.

3 GENERAL INFORMATION

Use of common: uses common areas `KB09C/CD/CI,` see §2.2.

Workspace: provided by the user (`N` words).

Other subroutines: contains a private subroutine `KB09B/BD/BI/BJ.`

Input/Output: in the event of errors diagnostic messages are printed, see §2.2.

Restrictions:

`0≤N≤NN,`

`KPOS>0,`

`KLEN>0,`

`LREC>0.`

4 METHOD

The subroutine uses a simple two-way merge as described by Knuth, ‘The Art of Computer Programming’, (1973), Vol 3, Sorting and searching, pp 160.

5 EXAMPLE OF USE

Suppose we have an integer array `TAB` containing 100 columns of length 10, so that `TAB` is dimensioned `TAB(10,100)`, and we wish to create a matrix from `TAB` containing the columns of `TAB` in ascending order. The code for this might be as shown in Figure 1.

For a second example, suppose we had stored in `TAB` fifty 10×20 matrices of integers so that `TAB` was dimensioned `TAB(10,1000)`. Now suppose it is required to write out these matrices in ascending order as determined primarily by the 1st entry in the 1st column and secondarily by first the 7th, and then the 8th columns in the 4th column, then the code might be as shown in Figure 2.

```
      INTEGER TAB(10,100),MATRIX(10,100)
      INTEGER INDX(100),IWORK(100)
       -
C      SET ARGUMENT VALUES
length of table
      NN=100
 number of records to be sorted
      N=100
key field is the 8th word ...
      KEYPOS=8
... in each record (column)
      KLEN=1
each record (column) is 10 words
      LREC=10
indicate ascending order
      IND=.TRUE.
  select all the columns ...
      DO 1 I=1,N
 ... to be included in the sort
    1 INDX(I)=I
sort table
      CALL KB09AI(TAB,INDX,NN,N,KEYPOS,KLEN,LREC,IWORK,IND)
C      CREATE AN ORDERED MATRIX USING ORDERING IN INDX
      DO 2 I=1,N
      K=INDX(I)
      DO 3 J=1,10
    3 MATRIX(J,I)=TAB(J,K)
    2 CONTINUE
        -
```

Figure 1. The first example code

```
      INTEGER TAB(10,1000)
      INTEGER INDX(100),IWORK(100)
C      SET ARGUMENT VALUES FOR THE FIRST SUBSORT
      NN=1000
      N=50
      KEYPOS=7
      KEYLEN=2
      LREC=10
      IND=.TRUE.
C      INITIALIZE THE INDEX ARRAY
      K=4
      DO 1 I=1,N
      INDX(I)=K
    1 K=K+20
C      SORT THE MATRICES ON THE 7TH AND 8TH ENTRIES
C      OF THEIR 4TH COLUMNS
      CALL KB09AI(TAB,INDX,NN,N,KEYPOS,KEYLEN,LRECL,IWORK,IND)
C      RESET INDX TO REFERENCE 1ST COLUMN INSTEAD OF 4TH
      DO 2 I=1,N
    2 INDX(I)=INDX(I)-3
C      RESET ARGUMENT VALUES
      KEYPOS=1
      KEYLEN=1
C      PERFORM THE PRIMARY SORT OF THE MATRICES
      CALL KB09AI(TAB,INDX,NN,N,KEYPOS,KEYLEN,LREC,IWORK,IND)
C      PRINT MATRICES IN ORDER
      DO 3 I=1,N
      K=INDX(I)
      MATEND=K+9
      DO 4 J=1,10
    4 WRITE(6,5) (TAB(J,M),M=K,MATEND)
    3 CONTINUE
    5 FORMAT(5X,20(1X,I4))
        -
```

Figure 2. The second example