KD01

1 SUMMARY

Sets up a dictionary of words, allowing the user to insert new words, and search for and remove existing words. Also allows the user to rebuild the dictionary if the maximum allowed word-size, or the total space provided, proves too small. Provided sufficient room is allowed, the expected number of operations required for an insertion, search or removal is $O(1)$. The method is based on the chained scatter table insertion method of F. A. Williams.

ATTRIBUTES — Version: 1.0.0. **Types:** `KD01A`. **Original date:** July 1990. **Origin:** N.I.M. Gould, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Calling sequence

All the words in the dictionary are entered into a so-called “chained scatter table”. Before the first word is entered, the table must be initialized by a call to `KD01A`. Words are inserted in the table by calling `KD01B`. The table may be searched for an existing word with a call to `KD01C`; an existing word may be flagged as deleted from the table by calling `KD01D`. Finally, the table may be rebuilt to allow for an increase in the maximum allowed word-size or the total number of entries in the table with a call to `KD01E`.

2.2 Initialization step

Call `KD01A` to initialize the table prior to insertions, searches and deletions

```
CALL KD01A ( LENGTH, NCHAR, ITABLE, KEY, DPRIME, IEMPTY )
```

`LENGTH` is an `INTEGER` variable set by the user to the lengths of the arrays `KEY` and `ITABLE`. It is an upper bound on the number of words that can be held in the dictionary. In general, the more space that can be supplied, the faster insertions and look-ups may be performed. It is unchanged by the subroutine. **Restriction:** `LENGTH` > 0.

`NCHAR` is an `INTEGER` variable set by the user to an upper bound on the maximum number of characters that will occur in any word placed in the dictionary. It is unchanged by the subroutine. **Restriction:** `NCHAR` > 0.

`ITABLE` is an `INTEGER` array of length `LENGTH` which need not be set by the user on entry. The i–th component of `ITABLE` is used in `KD01B/C/D` to indicate the status of entry i in the table. On exit from `KD01A`, all entries of `ITABLE` will be set to `-(LENGTH+1)` to indicate that the entries are currently unused. After subsequent calls to `KD01B/C/D/E`, a non-negative value of `ITABLE(`i`)` indicates that the i–th entry of the table is filled, while a negative one indicates an unused entry.

`KEY` is an `CHARACTER` array of length `LENGTH`, each element being of character length `NCHAR`, which need not be set by the user on entry. `KEY` is used to store the list of words in the dictionary; `KEY(`i`)` holds the word in table position i.

`DPRIME` is a `DOUBLE PRECISION` variable which need not be set by the user on entry. On exit, `DPRIME` gives the largest prime number that is no larger than `LENGTH`.

`IEMPTY` is an `INTEGER` variable which need not be set by the user on entry. On exit, `IEMPTY` gives the number of unused locations that remain in the table.

2.3 Inserting a word

Call KD01B to insert a word into the table.

```
    CALL KD01B ( LENGTH, NCHAR, ITABLE, KEY, DPRIME, IEMPTY,
   *             FIELD, IFREE )
```

The parameters LENGTH, NCHAR, ITABLE, KEY, DPRIME and IEMPTY are exactly as described for KD01A. They must not be altered between calls to KD01A/B/C/D/E.

FIELD is a CHARACTER variable of length NCHAR which must be set by the user to the word that is to be inserted into the dictionary. If the word contains fewer than NCHAR characters, it should be padded with blanks. FIELD is unchanged by the subroutine.

IFREE is an INTEGER variable which need not be set by the user on entry to KD01B. On exit, a positive value of IFREE indicates that the word has been inserted as table entry IFREE. A negative exit value of IFREE indicates that the word already exists as table entry -IFREE. If IFREE is zero on exit, there is no more room in the table. In this case, LENGTH should be increased and the dictionary rebuilt using KD01E.

2.4 Finding a word

Call KD01C to find a word into the table.

```
    CALL KD01C ( LENGTH, NCHAR, ITABLE, KEY, DPRIME, IEMPTY,
   *             FIELD, IFREE )
```

The parameters LENGTH, NCHAR, ITABLE, KEY, DPRIME and IEMPTY are exactly as described for KD01A. They are unchanged by the subroutine and must not be altered between calls to KD01A/B/C/D/E.

FIELD is a CHARACTER variable of length NCHAR which must be set by the user to the word whose position is sought in the dictionary. If the word contains fewer than NCHAR characters, it should be padded with blanks. FIELD is unchanged by the subroutine.

IFREE is an INTEGER variable which need not be set by the user on entry to KD01C. If IFREE is positive on exit, the desired word occupies location IFREE in the table. A non-positive exit value of IFREE indicates that the word could not be found in the dictionary; a negative value indicates that the word once occupied location IFREE of the table but has subsequently been flagged as having been removed.

2.5 Removing a word

Call KD01D to flag an existing word as absent from the table.

```
    CALL KD01D ( LENGTH, NCHAR, ITABLE, KEY, DPRIME, IEMPTY,
   *             FIELD, IFREE )
```

The parameters LENGTH, NCHAR, ITABLE, KEY, DPRIME and IEMPTY are exactly as described for KD01A. They must not be altered between calls to KD01A/B/C/D/E.

FIELD is a CHARACTER variable of length NCHAR which must be set by the user to the word which is to be flagged as absent from the dictionary – subsequent calls to KD01C will reveal that the word was once present but is no longer considered to be so (see §2.4). If the word contains fewer than NCHAR characters, it should be padded with blanks. FIELD is unchanged by the subroutine.

IFREE is an INTEGER variable which need not be set by the user on entry to KD01C. On exit, a positive value of IFREE gives the position in the table of the word that has been flagged as deleted. A non-positive value indicates that the word could not be found in the table.

2.6 Rebuilding the table

Call KD01E to rebuild the table if the current maximum word-size, NCHAR, is too small or if there is insufficient room, LENGTH, in the current table.

```
    CALL KD01E ( LENGTH, NCHAR, ITABLE, KEY, DPRIME, IEMPTY,
   *             NEWLEN, NEWNCH, NEWTAB, NEWKEY, DNEWPR,
   *             INEWEM, CHARWK, INFORM )
```

The parameters LENGTH, NCHAR, ITABLE, KEY, DPRIME and IEMPTY are exactly as described for KD01A. They are unchanged by the subroutine and must not be have been altered since the last call to KD01A/B/C/D.

NEWLEN is an INTEGER variable set by the user to the lengths of the arrays NEWKEY and NEWTAB. It is the new upper bound on the number of words that can be held in the dictionary. In general, the more space that can be supplied, the faster insertions and look-ups may be performed. It is unchanged by the subroutine. On subsequent calls to KD01B/C/D/E, NEWLEN must replace LENGTH. **Restriction:** NEWLEN >0.

NEWNCH is an INTEGER variable set by the user to the new upper bound the maximum number of characters that will occur in any word placed in the dictionary. It is unchanged by the subroutine. On subsequent calls to KD01B/C/D/E, NEWNCH must replace NCHAR. **Restriction:** NEWNCH $\geq$ NCHAR.

NEWTAB is an INTEGER array of length NEWLEN which need not be set by the user on entry. The *i*–th component of ITABLE is used to indicate the status of entry *i* in the table. On exit from KD01E, a non-negative value of NEWTAB(*i*) indicates that the *i*–th entry of the table is filled, while a negative one indicates an unused entry. On subsequent calls to KD01B/C/D/E, NEWTAB must replace ITABLE.

NEWKEY is an CHARACTER array of length NEWLEN, each element being of character length NEWNCH which need not be set by the user on entry. NEWKEY is used to store the list of words in the new dictionary; NEWKEY(*i*) holds the word in table position *i*. On subsequent calls to KD01B/C/D/E, NEWKEY must replace KEY.

DNEWPR is a DOUBLE PRECISION variable which gives the largest prime number that is no larger than NEWLEN. On subsequent calls to KD01B/C/D/E, DNEWPR must replace DPRIME.

INEWEM is an INTEGER variable which gives the number of unused locations that remain in the new table. On subsequent calls to KD01B/C/D/E, INEWEM must replace IEMPTY.

CHARWK is a CHARACTER array of size NEWNCH which is used as workspace by KD01E. It need not be set on entry.

INFORM is an INTEGER variable which need not be set by the user on entry to KD01E. On exit, a successful call is indicated when INFORM=0. If INFORM=-1, NEWLEN has not been set sufficiently large to accommodate the existing list of words in the new table. If INFORM=-2, NEWNCH $<$ NCHAR on entry. In both cases, the dictionary will not have been rebuilt.

2.7 Error returns

Unsuccessful calls to KD01B/C/D are indicated by non-positive exit values of the parameter IFREE. An unsuccessful call to KD01E is indicated by a negative exit value of the parameter INFORM. See the relevant sections above.

3 GENERAL INFORMATION

Use of common: None.

Workspace: None.

Other routines called directly: Calls KD01F, KD01G and KD01H.

Input/output: None.

Restrictions: LENGTH >0 and NCHAR >0.

4 METHOD

The chained scatter table search and insertion method is due to F. A. Williams (Communications of the ACM **2** (1959), 21-24)

To insert a word in the table, the word is first mapped onto an integer value, the entry integer. This mapping is often called hashing. As many words may be mapped to the same value (a collision), a chain of used locations starting from the entry integer is searched until an empty location is found. The word is inserted in the table at this point and the chain extended to the next unoccupied entry. The hashing routine is intended to reduce the number of collisions. Words are located and flagged as deleted from the table in exactly the same way; the word is hashed and the resulting chain searched until the word is matched or the end of the chain reached. Provided there is sufficient space in the table, the expected number of operations needed to perform an insertion, search or removal is $O(1)$.

5 EXAMPLE OF USE

As a simple example, we read a set of words into a table and then test if a second set of words is present. Each set is terminated by a blank. A maximum word length of 10 characters is specified. We might use the following piece of code.

```
      PROGRAM MAIN
      INTEGER          LENGTH, NCHAR, IEMPTY, IFREE
      DOUBLE PRECISION DPRIME
      PARAMETER        ( NCHAR = 10, LENGTH = 100 )
      INTEGER          ITABLE( LENGTH )
      CHARACTER * 10   FIELD, KEY( LENGTH )
      EXTERNAL         KD01A , KD01B , KD01C
C
C  SET UP THE INITIAL TABLE.
C
      CALL KD01A ( LENGTH, NCHAR, ITABLE, KEY, DPRIME, IEMPTY )
C
C  STORE A SET OF WORDS IN THE TABLE.
C
   10 CONTINUE
      READ( 5, 1000 ) FIELD
      IF ( FIELD .NE. '          ' ) THEN
         CALL KD01B ( LENGTH, NCHAR, ITABLE, KEY, DPRIME, IEMPTY,
     *                FIELD, IFREE )
         IF ( IFREE .GT. 0 ) THEN
            WRITE( 6, 2000 ) FIELD, IFREE
         ELSE
            WRITE( 6, 2010 ) FIELD, - IFREE
         END IF
         GO TO 10
      END IF
C
C  SEARCH THE TABLE FOR A SECOND SET OF WORDS.
C
   20 CONTINUE
      READ( 5, 1000 ) FIELD
      IF ( FIELD .NE. '          ' ) THEN
         CALL KD01C ( LENGTH, NCHAR, ITABLE, KEY, DPRIME, IEMPTY,
     *                FIELD, IFREE )
         IF ( IFREE .GT. 0 ) THEN
            WRITE( 6, 2020 ) FIELD, IFREE
         ELSE
            WRITE( 6, 2030 ) FIELD
         END IF
         GO TO 20
      END IF
      STOP
 1000 FORMAT( A10 )
 2000 FORMAT( ' WORD ', A10, ' INSERTED  IN TABLE POSITION ', I3 )
 2010 FORMAT( ' WORD ', A10, ' ALREADY   IN TABLE POSITION ', I3 )
 2020 FORMAT( ' WORD ', A10, ' FOUND     IN TABLE POSITION ', I3 )
 2030 FORMAT( ' WORD ', A10, ' ABSENT  FROM TABLE ' )
      END
```

Suitable data is:

```
ALPHA
BETA
GAMMA
DELTA
X111111111
X111111112
X111111111
X111111114

ALPHA
BETA
GAMMA
DELTA
EPSILON
X111111112
X111111113
X111111114
X111111111
OMEGA
```

This produces the following output:

```
 WORD ALPHA      INSERTED  IN TABLE POSITION  41
 WORD BETA       INSERTED  IN TABLE POSITION  81
 WORD GAMMA      INSERTED  IN TABLE POSITION  56
 WORD DELTA      INSERTED  IN TABLE POSITION  54
 WORD X111111111 INSERTED  IN TABLE POSITION  55
 WORD X111111112 INSERTED  IN TABLE POSITION  19
 WORD X111111111 ALREADY   IN TABLE POSITION  55
 WORD X111111114 INSERTED  IN TABLE POSITION  44
 WORD ALPHA      FOUND     IN TABLE POSITION  41
 WORD BETA       FOUND     IN TABLE POSITION  81
 WORD GAMMA      FOUND     IN TABLE POSITION  56
 WORD DELTA      FOUND     IN TABLE POSITION  54
 WORD EPSILON    ABSENT  FROM TABLE
 WORD X111111112 FOUND     IN TABLE POSITION  19
 WORD X111111113 ABSENT  FROM TABLE
 WORD X111111114 FOUND     IN TABLE POSITION  44
 WORD X111111111 FOUND     IN TABLE POSITION  55
 WORD OMEGA      ABSENT  FROM TABLE
```