NAG Fortran Library Routine Document

M01CCF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

M01CCF rearranges a vector of character data so that a specified substring is in ASCII or reverse ASCII order.

2 Specification

```
SUBROUTINE M01CCF(CH, M1, M2, L1, L2, ORDER, IFAIL)
INTEGER           M1, M2, L1, L2, IFAIL
CHARACTER*(*)     CH(M2)
CHARACTER*1       ORDER
```

3 Description

M01CCF is based on Singleton's implementation of the 'median-of-three' Quicksort algorithm (Singleton (1969)), but with two additional modifications. First, small subfiles are sorted by an insertion sort on a separate final pass (Sedgewick (1978)) Second, if a subfile is partitioned into two very unbalanced subfiles, the larger of them is flagged for special treatment: before it is partitioned, its end-points are swapped with two random points within it; this makes the worst case behaviour extremely unlikely.

Only the substring (L1:L2) of each element of the array CH is used to determine the sorted order, but the entire elements are rearranged into sorted order.

4 References

Sedgewick R (1978) Implementing Quicksort programs *Comm. ACM* **21** 847–857

Singleton R C (1969) An efficient algorithm for sorting with minimal storage: Algorithm 347 *Comm. ACM* **12** 185–187

5 Parameters

1: CH(M2) – CHARACTER*(*) array *Input/Output*

On entry: elements M1 to M2 of CH must contain character data to be sorted.

Constraint: the length of each element of CH must not exceed 255.

On exit: these values are rearranged into sorted order.

2: M1 – INTEGER *Input*

On entry: the index of the first element of CH to be sorted.

Constraint: $M1 > 0$.

3: M2 – INTEGER *Input*

On entry: the index of the last element of CH to be sorted.

Constraint: $M2 \geq M1$.

4: L1 – INTEGER *Input*
5: L2 – INTEGER *Input*

On entry: only the substring (L1:L2) of each element of CH is to be used in determining the sorted order.

Constraint: $0 < \text{L1} \leq \text{L2} \leq \text{LEN(CH(1))}$.

6: ORDER – CHARACTER*1 *Input*

On entry: if ORDER is 'A', the values will be sorted into ASCII order; if ORDER is 'R', into reverse ASCII order.

Constraint: ORDER = 'A' or 'R'.

7: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, M2 < 1,
or M1 < 1,
or M1 $>$ M2,
or L2 < 1,
or L1 < 1,
or L1 $>$ L2,
or L2 $>$ LEN(CH(1)).

IFAIL $= 2$

On entry, ORDER is not 'A' or 'R'.

IFAIL $= 3$

On entry, the length of each element of CH exceeds 255.

7 Accuracy

Not applicable.

8 Further Comments

The average time taken by the routine is approximately proportional to $n \times \log n$, where $n = \text{M2} - \text{M1} + 1$. The worst case time is proportional to n^2, but this is extremely unlikely to occur.

The routine relies on the Fortran 77 intrinsic functions LLT and LGT to order characters according to the ASCII collating sequence.

9 Example

The example program reads a file of 12-character records, and sorts them into reverse ASCII order on characters 7 to 12.

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     M01CCF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX
      PARAMETER        (MMAX=100)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, L1, L2, M
*     .. Local Arrays ..
      CHARACTER*12     CH(MMAX)
*     .. External Subroutines ..
      EXTERNAL         M01CCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'M01CCF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      DO 20 M = 1, MMAX
         READ (NIN,'(A)',END=40) CH(M)
   20 CONTINUE
   40 M = M - 1
      L1 = 7
      L2 = 12
      IFAIL = 0
*
      CALL M01CCF(CH,1,M,L1,L2,'Reverse ASCII',IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'Records sorted on columns ', L1, '  to ', L2
      WRITE (NOUT,*)
      WRITE (NOUT,99998) (CH(I),I=1,M)
      STOP
*
99999 FORMAT (1X,A,I2,A,I2)
99998 FORMAT (1X,A)
      END
```

9.2 Program Data

```
M01CCF Example Program Data
A02AAF   289
A02ABF   523
A02ACF   531
C02ADF   169
C02AEF   599
C05ADF  1351
C05AGF   240
C05AJF   136
C05AVF   211
C05AXF   183
C05AZF  2181
```

9.3 Program Results

```
 M01CCF Example Program Results

 Records sorted on columns  7  to 12

 C05AZF  2181
 C05ADF  1351
 C02AEF   599
 A02ACF   531
 A02ABF   523
 A02AAF   289
 C05AGF   240
 C05AVF   211
 C05AXF   183
 C02ADF   169
 C05AJF   136
```