

NAG 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 (see Singleton (1969)), but with two additional modifications. First, small subfiles are sorted by an insertion sort on a separate final pass (see 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 ≥ 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: $0 < L1 \leq L2 \leq \text{LEN}(\text{CH}(1))$.

6: ORDER – CHARACTER(1) *Input*

On entry: if ORDER = 'A', the values will be sorted into ASCII order.

If ORDER = '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. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

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, if you are 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.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

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 $n = M2 - M1 + 1$. The worst case time is proportional to n^2 , but this is extremely unlikely to occur.

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

9 Example

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

9.1 Program Text

```

Program m01ccfe

!      M01CCF Example Program Text

!      Mark 24 Release. NAG Copyright 2012.

!      .. Use Statements ..
      Use nag_library, Only: m01ccf
!      .. Implicit None Statement ..
      Implicit None
!      .. Parameters ..
      Integer, Parameter          :: nin = 5, nout = 6
!      .. Local Scalars ..
      Integer                    :: i, ifail, l1, l2, m1, m2
!      .. Local Arrays ..
      Character (12), Allocatable :: ch(:)
!      .. Executable Statements ..
      Write (nout,*) 'M01CCF Example Program Results'

!      Skip heading in data file
      Read (nin,*)

      Read (nin,*) m2
      Allocate (ch(m2))

      m1 = 1

      Do i = m1, m2
         Read (nin,'(A)') ch(i)
      End Do

      l1 = 7
      l2 = 12

      ifail = 0
      Call m01ccf(ch,m1,m2,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=m1,m2)

99999 Format (1X,A,I2,A,I2)
99998 Format (1X,A)
      End Program m01ccfe

```

9.2 Program Data

```

M01CCF Example Program Data
11
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
