

NAG Library Routine Document

M01DCF

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

M01DCF ranks a vector of character data in ASCII or reverse ASCII order of a specified substring.

2 Specification

```
SUBROUTINE M01DCF (CH, M1, M2, L1, L2, ORDER, IRANK, IFAIL)
INTEGER          M1, M2, L1, L2, IRANK(M2), IFAIL
CHARACTER(*)    CH(M2)
CHARACTER(1)    ORDER
```

3 Description

M01DCF uses a variant of list-merging, as described on pages 165–166 in Knuth (1973). The routine takes advantage of natural ordering in the data, and uses a simple list insertion in a preparatory pass to generate ordered lists of length at least 10. The ranking is stable: equal elements preserve their ordering in the input data.

Only the substring (L1:L2) of each element of the array CH is used to determine the rank order.

4 References

Knuth D E (1973) *The Art of Computer Programming (Volume 3)* (2nd Edition) Addison–Wesley

5 Parameters

- | | | |
|----|--|--------------|
| 1: | CH(M2) – CHARACTER(*) array | <i>Input</i> |
| | <i>On entry:</i> elements M1 to M2 of CH must contain character data to be ranked. | |
| | <i>Constraint:</i> the length of each element of CH must not exceed 255. | |
| 2: | M1 – INTEGER | <i>Input</i> |
| | <i>On entry:</i> the index of the first element of CH to be ranked. | |
| | <i>Constraint:</i> M1 > 0. | |
| 3: | M2 – INTEGER | <i>Input</i> |
| | <i>On entry:</i> the index of the last element of CH to be ranked. | |
| | <i>Constraint:</i> M2 ≥ M1. | |
| 4: | L1 – INTEGER | <i>Input</i> |
| 5: | L2 – INTEGER | <i>Input</i> |
| | <i>On entry:</i> only the substring (L1:L2) of each element of CH is to be used in determining the rank order. | |
| | <i>Constraint:</i> 0 < L1 ≤ L2 ≤ LEN(CH(1)). | |

- 6: ORDER – CHARACTER(1) *Input*
On entry: if ORDER = 'A', the values will be ranked in ASCII order.
 If ORDER = 'R', in reverse ASCII order.
Constraint: ORDER = 'A' or 'R'.
- 7: IRANK(M2) – INTEGER array *Output*
On exit: elements M1 to M2 of IRANK contain the ranks of the corresponding elements of CH. Note that the ranks are in the range M1 to M2: thus, if CH(*i*) is the first element in the rank order, IRANK(*i*) is set to M1.
- 8: 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.

IFAIL = -99

An unexpected error has been triggered by this routine. Please contact NAG.
 See Section 3.8 in the Essential Introduction for further information.

IFAIL = -399

Your licence key may have expired or may not have been installed correctly.
 See Section 3.7 in the Essential Introduction for further information.

IFAIL = -999

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The average time taken by the routine is approximately proportional to $n \times \log(n)$, where $n = M2 - M1 + 1$.

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

10 Example

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

10.1 Program Text

```

Program m01dcfe
!      M01DCF Example Program Text
!
!      Mark 25 Release. NAG Copyright 2014.
!
!      .. Use Statements ..
Use nag_library, Only: m01dcf
!      .. Implicit None Statement ..
Implicit None
!      .. Parameters ..
Integer, Parameter          :: nin = 5, nout = 6
!      .. Local Scalars ..
Integer                    :: i, ifail, l1, l2, m1, m2
!      .. Local Arrays ..
Integer, Allocatable       :: irank(:)
Character (12), Allocatable :: ch(:)
!      .. Executable Statements ..
Write (nout,*) 'M01DCF Example Program Results'

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

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

      m1 = 1

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

      l1 = 7
      l2 = 12

      ifail = 0

```

```

Call m01dcf(ch,m1,m2,l1,l2,'Reverse ASCII',irank,ifail)

Write (nout,*)
Write (nout,99999) 'Records ranked on columns ', l1, ' to ', l2
Write (nout,*)
Write (nout,*) 'Data           Ranks'
Write (nout,*)
Write (nout,99998)(ch(i),irank(i),i=m1,m2)

99999 Format (1X,A,I2,A,I2)
99998 Format (1X,A,I7)
End Program m01dcfe

```

10.2 Program Data

M01DCF Example Program Data

```

11
A02AAF  289
A02ABF  523
A02ACF  531
C02ADF  169
C02AEF  599
C05AUF 1351
C05AVF  240
C05AWF  136
C05AXF  211
C05AYF  183
C05AZF 2181

```

10.3 Program Results

M01DCF Example Program Results

Records ranked on columns 7 to 12

Data		Ranks
A02AAF	289	6
A02ABF	523	5
A02ACF	531	4
C02ADF	169	10
C02AEF	599	3
C05AUF	1351	2
C05AVF	240	7
C05AWF	136	11
C05AXF	211	8
C05AYF	183	9
C05AZF	2181	1
