NAG Library Routine Document

M01ECF

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

M01ECF rearranges a vector of character data into the order specified by a vector of ranks.

2 Specification

```
SUBROUTINE M01ECF (CH, M1, M2, IRANK, IFAIL)

INTEGER      M1, M2, IRANK(M2), IFAIL
CHARACTER(*) CH(M2)
```

3 Description

M01ECF is designed to be used typically in conjunction with the M01D ranking routines. After one of the M01D routines has been called to determine a vector of ranks, M01ECF can be called to rearrange a vector of character data into the rank order. If the vector of ranks has been generated in some other way, then M01ZBF should be called to check its validity before M01ECF is called.

4 References

None.

5 Parameters

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

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

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

On exit: these values are rearranged into rank order. For example, if IRANK(i) = M1, then the initial value of CH(i) is moved to CH(M1).

2: M1 – INTEGER *Input*

3: M2 – INTEGER *Input*

On entry: the range of the ranks supplied in IRANK and the elements of CH to be rearranged.

Constraint: $0 <$ M1 $\leq$ M2.

4: IRANK(M2) – INTEGER array *Input/Output*

On entry: elements M1 to M2 of IRANK must contain a permutation of the integers M1 to M2, which are interpreted as a vector of ranks.

On exit: used as internal workspace prior to being restored and hence is unchanged.

5: 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.

IFAIL $= 2$

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

IFAIL $= 3$

Elements M1 to M2 of IRANK contain a value outside the range M1 to M2.

IFAIL $= 4$

Elements M1 to M2 of IRANK contain a repeated value.

If IFAIL $= 3$ or 4, elements M1 to M2 of IRANK do not contain a permutation of the integers M1 to M2. On exit, the contents of CH may be corrupted. To check the validity of IRANK without the risk of corrupting CH, use M01ZBF.

7 Accuracy

Not applicable.

8 Further Comments

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

9 Example

This example reads a file of 12-character records, each of which contains in characters 1 to 6 a name of a NAG routine, and in characters 7 to 12 an integer frequency. The program first calls M01DBF to rank the integers in descending order, and then calls M01ECF to rearrange the names into the order specified by the ranks.

9.1 Program Text

```
    Program m01ecfe

!     M01ECF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: m01dbf, m01ecf
!     .. Implicit None Statement ..
      Implicit None
```

```
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, m1, m2
!     .. Local Arrays ..
      Integer, Allocatable             :: ifreq(:), irank(:)
      Character (6), Allocatable       :: ch(:)
!     .. Executable Statements ..
      Write (nout,*) 'M01ECF Example Program Results'

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

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

      m1 = 1

      Do i = m1, m2
        Read (nin,99999,End=100) ch(i), ifreq(i)
      End Do

      ifail = 0
      Call m01dbf(ifreq,m1,m2,'Descending',irank,ifail)

      ifail = 0
      Call m01ecf(ch,m1,m2,irank,ifail)

      Write (nout,*)
      Write (nout,*) 'Names in order of frequency'
      Write (nout,*)
      Write (nout,99998)(ch(i),i=m1,m2)

100   Continue

99999 Format (A6,I6)
99998 Format (1X,A)
    End Program m01ecfe
```

9.2 Program Data

```
M01ECF 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

```
 M01ECF Example Program Results

 Names in order of frequency

 C05AZF
 C05ADF
 C02AEF
 A02ACF
 A02ABF
 A02AAF
```

C05AGF
C05AVF
C05AXF
C02ADF
C05AJF
