NAG Library Routine Document

M01CBF

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

M01CBF rearranges a vector of integer numbers into ascending or descending order.

2 Specification

```
SUBROUTINE M01CBF (IV, M1, M2, ORDER, IFAIL)
INTEGER      IV(M2), M1, M2, IFAIL
CHARACTER(1) ORDER
```

3 Description

M01CBF 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.

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: IV(M2) – INTEGER array *Input/Output*

On entry: elements M1 to M2 of IV must contain integer values to be sorted.

On exit: these values are rearranged into sorted order.

2: M1 – INTEGER *Input*

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

Constraint: M1 $>$ 0.

3: M2 – INTEGER *Input*

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

Constraint: M2 $\geq$ M1.

4: ORDER – CHARACTER(1) *Input*

On entry: if ORDER = 'A', the values will be sorted into ascending (i.e., nondecreasing) order.

If ORDER = 'D', into descending order.

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

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, ORDER is not ‘A’ or ‘D’.

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

M01CBF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

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

10 Example

This example reads a list of integers and sorts them into descending order.

10.1 Program Text

```
    Program m01cbfe

!     M01CBF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: m01cbf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, m1, m2
!     .. Local Arrays ..
      Integer, Allocatable             :: iv(:)
!     .. Executable Statements ..
      Write (nout,*) 'M01CBF Example Program Results'

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

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

      m1 = 1

      Read (nin,*)(iv(i),i=m1,m2)

      ifail = 0
      Call m01cbf(iv,m1,m2,'Descending',ifail)

      Write (nout,*)
      Write (nout,*) 'Sorted numbers'
      Write (nout,*)
      Write (nout,99999)(iv(i),i=m1,m2)

99999 Format (1X,10I7)
    End Program m01cbfe
```

10.2 Program Data

```
M01CBF Example Program Data
16
23 45 45 67 69 90 999 1
78 112 24 69 96 99 45 78
```

10.3 Program Results

```
 M01CBF Example Program Results

 Sorted numbers

     999    112     99     96     90     78     78     69     69     67
      45     45     45     24     23      1
```