NAG Library Routine Document

M01NCF

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

M01NCF examines an ordered vector of null terminated strings and returns the index of the first value equal to the sought-after item. Character items are compared according to the ASCII collating sequence.

2 Specification

```
FUNCTION M01NCF (VALID, CH, M1, M2, ITEM, IFAIL)
INTEGER M01NCF
INTEGER      M1, M2, IFAIL
LOGICAL      VALID
CHARACTER(*) CH(M2), ITEM
```

3 Description

M01NCF is based on Professor Niklaus Wirth's implementation of the Binary Search algorithm (see Wirth (2004)), but with two modifications. First, if the sought-after item is less than the value of the first element of the array to be searched, 0 is returned. Second, if a value equal to the sought-after item is not found, the index of the immediate lower value is returned.

4 References

Wirth N (2004) *Algorithms and Data Structures* 35–36 Prentice Hall

5 Arguments

1: VALID – LOGICAL *Input*

On entry: if VALID is set to .TRUE. argument checking will be performed. If VALID is set to .FALSE. M01NCF will be called without argument checking, which includes checking that array CH is sorted in ascending order and the routine will return with IFAIL $= 0$. See Section 9 for further details.

2: CH(M2) – CHARACTER(*) array *Input*

On entry: elements M1 to M2 contain character data to be searched.

Constraint: elements M1 to M2 of CH must be sorted in ascending order. The length of each element of CH must not exceed 255. Trailing space characters are ignored.

3: M1 – INTEGER *Input*

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

Constraint: M1 ≥ 1.

4: M2 – INTEGER *Input*

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

Constraint: M2 $\geq$ M1.

5: ITEM – CHARACTER(*) *Input*

On entry: the sought-after item. Trailing space characters are ignored.

6: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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 argument, 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:

(**Note:** these errors will only be returned if VALID = .TRUE..)

IFAIL $= 2$

On entry, CH must be sorted in ascending order: CH element $\langle value \rangle >$ element $\langle value \rangle$.

IFAIL $= 3$

On entry, M1 $= \langle value \rangle$.
Constraint: M1 ≥ 1.

IFAIL $= 4$

On entry, M1 $= \langle value \rangle$, M2 $= \langle value \rangle$.
Constraint: M2 $\geq$ M1.

IFAIL $= 5$

On entry, the length of each element of CH must be at most 255: maximum string length $= \langle value \rangle$.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

M01NCF is not threaded in any implementation.

9 Further Comments

The argument VALID should be used with caution. Set it to .FALSE. only if you are confident that the other arguments are correct, in particular that array CH is in fact arranged in ascending order. If you wish to search the same array CH many times, you are recommended to set VALID to .TRUE. on first call of M01NCF and to .FALSE. on subsequent calls, in order to minimize the amount of time spent checking CH, which may be significant if CH is large.

The time taken by M01NCF is $O(\log(n))$, where $n = \mathrm{M2} - \mathrm{M1} + 1$, when VALID = .FALSE..

10 Example

This example reads a list of character data and sought-after items and performs the search for these items.

10.1 Program Text

```
    Program m01ncfe

!     M01NCF Example Program Text
!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: m01ncf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, index, ioerr, m1, m2
      Logical                          :: first
      Character (6)                    :: item
!     .. Local Arrays ..
      Character (6), Allocatable       :: ch(:)
!     .. Executable Statements ..
      Write (nout,*) 'M01NCF Example Program Results'

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

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

      m1 = 1

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

      first = .True.

data: Do
        Read (nin,'(A)',Iostat=ioerr) item

        If (ioerr<0) Then
          Exit data
        End If

        ifail = 0
        index = m01ncf(first,ch,m1,m2,item,ifail)
```

```
        If (first) Then
          Write (nout,*)
          Write (nout,*) 'Reference Vector is:'
          Write (nout,99999)(ch(i),i=m1,m2)
          first = .False.
        End If

        Write (nout,*)
        Write (nout,99998) item, index
      End Do data

99999 Format (10(1X,A))
99998 Format (1X,'Search for item ',A,' returned index: ',I4)
    End Program m01ncfe
```

10.2 Program Data

```
M01NCF Example Program Data
10                                                  : M2
A02AAF A02ABF A02ACF C02ADF C02AEF
C05AUF C05AVF C05AWF C05AXF C05AYF                  : CH
C02ADF                                              : Item 1
A01AAF                                              : Item 2
C04AYF                                              : Item 3
D01NBF                                              : Item 4
```

10.3 Program Results

```
 M01NCF Example Program Results

 Reference Vector is:
 A02AAF A02ABF A02ACF C02ADF C02AEF C05AUF C05AVF C05AWF C05AXF C05AYF

 Search for item C02ADF returned index:    4

 Search for item A01AAF returned index:    0

 Search for item C04AYF returned index:    5

 Search for item D01NBF returned index:   10
```