NAG Library Function Document

nag_search_int (m01nbc)

1 Purpose

nag_search_int (m01nbc) searches an ordered vector of integer numbers and returns the index of the first value equal to the sought-after item.

2 Specification

```
#include <nag.h>
#include <nagm01.h>
Integer nag_search_int (Nag_Boolean validate, const Integer iv[], Integer m1,
        Integer m2, Integer item, NagError *fail)
```

3 Description

nag_search_int (m01nbc) 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, -1 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: **validate** – Nag_Boolean *Input*

On entry: if **validate** is set to Nag_TRUE argument checking will be performed. If **validate** is set to Nag_FALSE nag_search_int (m01nbc) will be called without argument checking (which includes checking that array **iv** is sorted in ascending order) and the function will return with **fail.code** = NE_NOERROR. See Section 9 for further details.

2: **iv**[**m2** $+ 1$] – const Integer *Input*

On entry: elements **m1** to **m2** contain integer values to be searched.

Constraint: elements **m1** to **m2** of **iv** must be sorted in ascending order.

3: **m1** – Integer *Input*

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

Constraint: **m1** ≥ 0.

4: **m2** – Integer *Input*

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

Constraint: **m2** $\geq$ **m1**.

5: **item** – Integer *Input*

On entry: the sought-after item.

6: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument $\langle value\rangle$ had an illegal value.

NE_INT

On entry, **m1** $= \langle value\rangle$.
Constraint: **m1** ≥ 0.

NE_INT_2

On entry, **m1** $= \langle value\rangle$, **m2** $= \langle value\rangle$.
Constraint: **m1** $\leq$ **m2**.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_NOT_INCREASING

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

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_search_int (m01nbc) is not threaded in any implementation.

9 Further Comments

The argument **validate** should be used with caution. Set it to Nag_FALSE only if you are confident that the other arguments are correct, in particular that array **iv** is in fact arranged in ascending order. If you wish to search the same array **iv** many times, you are recommended to set **validate** to Nag_TRUE on first call of nag_search_int (m01nbc) and to Nag_FALSE on subsequent calls, in order to minimize the amount of time spent checking **iv**, which may be significant if **iv** is large.

The time taken by nag_search_int (m01nbc) is $O(\log(n))$, where $n = \mathbf{m2} - \mathbf{m1} + 1$, when **validate** = Nag_FALSE.

10 Example

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

10.1 Program Text

```
/* nag_search_int (m01nbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagm01.h>

int main(void)
{
  /*Logical scalar and array declarations */
  Nag_Boolean validate;
  /*Integer scalar and array declarations */
  Integer exit_status = 0;
  Integer i, index, item, leniv, m1, m2;
  Integer *iv = 0;
  NagError fail;

  INIT_FAIL(fail);

  printf("%s\n", "nag_search_int (m01nbc) Example Program Results");
  printf("\n");
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &leniv);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &leniv);
#endif
  if (!(iv = NAG_ALLOC(leniv, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  /* Read in Reference Vector iv */
  for (i = 0; i < leniv; i++)
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " ", &iv[i]);
#else
    scanf("%" NAG_IFMT " ", &iv[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  /* Read items sought in the reference vector */
  validate = Nag_TRUE;
  m1 = 0;
  m2 = leniv - 1;
#ifdef _WIN32
```

```
  while (scanf_s("%" NAG_IFMT "%*[^\n] ", &item) != EOF)
#else
  while (scanf("%" NAG_IFMT "%*[^\n] ", &item) != EOF)
#endif
  {
    /*
     * nag_search_int (m01nbc)
     * Binary search in set of integer numbers
     */
    index = nag_search_int(validate, iv, m1, m2, item, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_search_int (m01nbc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
    if (validate) {
      /* Print the reference vector */
      printf("%s\n", "Reference Vector is:");
      for (i = 0; i < leniv; i++)
        printf("%5" NAG_IFMT "%s", iv[i], (i + 1) % 8 ? " " : "\n");
      printf("\n");
      validate = Nag_FALSE;
    }
    printf("\n");
    printf("  Search for item %5" NAG_IFMT " returned index: %4" NAG_IFMT
           "\n", item, index);
  }

END:
  NAG_FREE(iv);

  return exit_status;
}
```

10.2 Program Data

```
nag_search_int (m01nbc) Example Program Data
16                                                     : leniv
5  6  11  12  13  13  21  23
23  41  58  59  65  65  86  99                         : iv
21                                                     : item 1
4                                                      : item 2
71                                                     : item 3
100                                                    : item 4
```

10.3 Program Results

```
nag_search_int (m01nbc) Example Program Results

Reference Vector is:
    5     6    11    12    13    13    21    23
   23    41    58    59    65    65    86    99


  Search for item    21 returned index:    6

  Search for item     4 returned index:   -1

  Search for item    71 returned index:   13

  Search for item   100 returned index:   15
```