nag_ran_permut_vec (g05ehc)

1. Purpose
nag_ran_permut_vec (g05ehc) performs a pseudo-random permutation of a vector of integers.

2. Specification
#include <nag.h>
#include <nag05.h>

void nag_ran_permut_vec(Integer index[], Integer n, NagError *fail)

3. Description
The function generates a single pseudo-random permutation of the elements of index without inspecting their values. Each of the \( n! \) possible permutations of the \( n \) values may be regarded as being equiprobable.

4. Parameters
index[n]
   Input: the \( n \) integer values to be permuted.
   Output: the \( n \) permuted integer values.

n
   Input: the number of values to be permuted.
   Constraint: \( n \geq 1 \).

fail
   The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings
\[ \text{NE_INT_ARG_LT} \]
   On entry, \( n \) must not be less than 1: \( n = \langle \text{value} \rangle \).

6. Further Comments
   It should be noted that if \( n \) is 20 or more it is theoretically impossible to generate all \( n! \) permutations as \( n! \) exceeds the cycle length of the internal random number generator.
   The time taken by the function is of order \( n \).
   In order to permute other kinds of objects (i.e., vectors, or matrices of higher dimensions), the following technique may be used:
   (a) Set \( \text{index}[i - 1] = i \), for \( i = 1, 2, \ldots, n \) (where \( n \) is the number of objects)
   (b) Use nag_ran_permut_vec to permute \( \text{index} \)
   (c) Use the contents of \( \text{index} \) as a set of indices to access the relevant object.
   In order to divide pseudo-randomly an array of \( n \) objects (\( \text{obj_array}[n] \)) into \( k \) subgroups of chosen sizes \( S_1, S_2, \ldots, S_k \), a similar procedure may be used. For the first \( S_1 \) elements of \( \text{index} \) set \( \text{index}[i] = 1, i = 0 \ldots S_1 - 1 \), for the next \( S_2 \) elements of \( \text{index} \) set \( \text{index}[S_1 + i] = 2, i = 0 \ldots S_2 - 1 \), for size \( S_j \) set \( \text{index}[S_1 + S_2 + \ldots + S_{j-1} + i] = j, i = 0 \ldots S_j - 1 \) etc. Permute \( \text{index} \) using nag_ran_permut_vec and then, if \( \text{index}[i] = j \), \( \text{obj_array}[i] \) is to be included in the \( j \)th subgroup.

6.1. Accuracy
   Not applicable.

6.2. References
7. See Also

nag_ran_sample_vec (g05ejc)

8. Example

A vector containing 0 and the first 7 positive integers in ascending order is permuted and the permutation is printed. This is repeated a total of 10 times.

8.1. Program Text

/* nag_ran_permut_vec(g05ehc) Example Program  *
 * Copyright 1994 Numerical Algorithms Group. *
 * Mark 3, 1994. */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg05.h>

#define NMAX 8

main()
{
    Integer j, k, n, m;
    Integer index[NMAX];
    Integer seed = 0;
    Vprintf("g05ehc Example Program Results\n");
g05cbc(seed);
n = NMAX;
m = 10;
Vprintf("%ld Permutations of the first %ld integers \n\n", m, n);
for (j = 0; j < m; ++j)
{
    /* construct index vector to be permuted */
    for (k = 0; k < n; ++k)
        index[k] = k;
g05ehc(index, n, NAGERR_DEFAULT);
    for (k = 0; k < n; ++k)
        Vprintf("%ld ", index[k]);
    Vprintf("\n");
}
exit(EXIT_SUCCESS);
}

8.2. Program Data

None.

8.3. Program Results

g05ehc Example Program Results

10 Permutations of the first 8 integers

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>