Simple Way to Randomize an Array

By: APena 10-16-2009 10:06 AM Last Edited By: Example_Scrubber_Christoph 03-21-2017 10:42 AM

Products and Environment

To download NI software, including the products shown below, visit ni.com/downloads.

Software
LabVIEW

Code and Documents

Attachment

- Randomize_Array_LV2009.vi 9 KB
- Randomize Numerical Array LV2012 NIVerified.vi 18 KB

Download All

Overview
The example demonstrates how to randomize the order of an array.

Description
To randomize the order of an array there are many possibilities. In this case the example generates a random array with the same length and is sorted. Looking at the indices of the array after the sort, you get a pretty random order that can be used to rearrange the input array in a random way.

Requirements
Software

- LabVIEW 2012 Base Development System (or compatible)

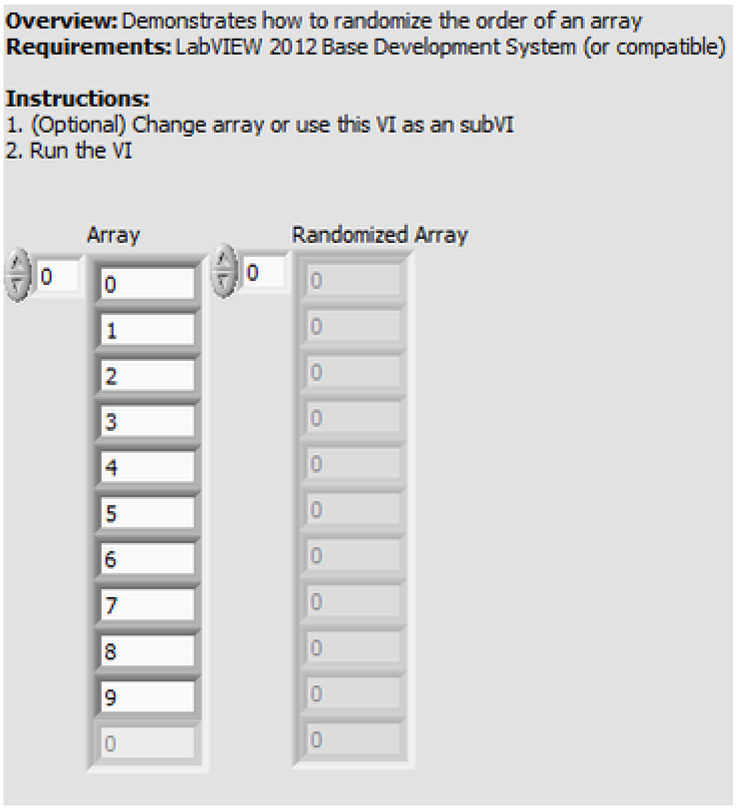
Hardware

- No hardware is necessary to use this example VI

Steps to Implement or Execute Code

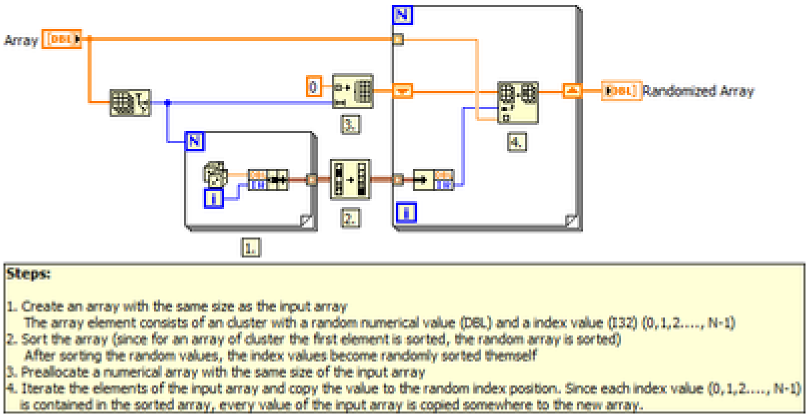
1. (Optional) Change array or use this VI as an subVI
2. Run the VI

Additional Information or References



Overview: Demonstrates how to randomize the order of an array
Requirements: LabVIEW 2012 Base Development System (or compatible)

Instructions:
1. (Optional) Change array or use this VI as an subVI
2. Run the VI



Steps:

1. Create an array with the same size as the input array
   The array element consists of an cluster with a random numerical value (DBL) and a index value (I32) (0,1,2...., N-1)
2. Sort the array (since for an array of cluster the first element is sorted, the random array is sorted)
   After sorting the random values, the index values become randomly sorted themself
3. Preallocate a numerical array with the same size of the input array
4. Iterate the elements of the input array and copy the value to the random index position. Since each index value (0,1,2...., N-1) is contained in the sorted array, every value of the input array is copied somewhere to the new array.

Remark:
A previous example used another algorithm to randomize the order of the input array. Since the calculation takes orders of magnitudes longer to calculate and does not effectively randomize longer arrays (>300 elements), the algorithm has been changed to reduce memory access and the number of random number generation.
Also as already mentioned in the comments:
The 'Riffle' function can be used to achive the same task.

The code for this example has been edited to meet the new Community Example Style Guidelines. The edited copy is marked with the text 'NIVerified'. Read here for more information about the new Example Guidelines and Community Platform.

★ 0 KUDOS SHARE

COMMENTS

ColeTrain NI EMPLOYEE (RETIRED) on 10-21-2009 04:20 PM

10-21-2009 04:20 PM

Hey Dr.Horrible,
Good example. If anyone has at least the full version, they can also use the "Riffle" function to get the same functionality. Cheers!

★ 0 KUDOS

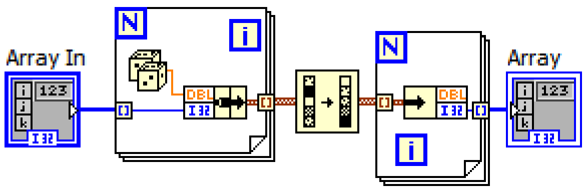
altenbach KNIGHT OF NI on 06-24-2010 01:41 PM

06-24-2010 01:41 PM

There a several problem:

- If you input array has a million elements, 1000 iterations won't cut it. Thee code does not scale!
- That constant array resizing (delete from array, insert into array) causes a huge memory allocation overhead penalty. Very inefficient.
- There is a statistical flaw. Because you round to the nearest integer, the first and last elements only have half the chance to be touched compared to all others. (Solution: Don't subtract (-1) from the array size, but round to -infinity instead.)

For a better solution, see my comment here for a much easier way.



...and if you have anything above LabVIEW base, use riffle. No code needed.

★ 2 KUDOS