

## Re: When random isn't random

**Source:**

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.language.objectpascal/2003-12/1228.html>

---

**From:** Dr John Stockton (*spam\_at\_merlyn.demon.co.uk*)

**Date:** 12/13/03

Date: Sat, 13 Dec 2003 19:05:02 +0000

JRS: In article <3fdae44c@newsgroups.borland.com>, seen in news:borland.public.delphi.language.objectpascal, Jens Gruschel <nospam@pegtop.net> posted at Sat, 13 Dec 2003 11:07:35 :-

>> >Description: Machines are getting so fast, I wonder if it is time to  
>> >slow down the Random function by making it use 64-bit seeds and results.  
>> >Maybe call it Random64( or 63 or 62 if we need to waste a couple of  
>> >bits)?

>>

>> *IMHO, the Random and Randomize functions should not be changed, for  
>> compatibility reasons.*

>

>That's why John wants to introduce the new function with a new name.

Obviously; he is intelligent. But the point needs stressing; not only should the improved functions have new names, but the old ones should remain exactly as they are (unless a bug fix is found necessary). That requires, but is not implied by, new names for new functions.

>> *A new Unit should instead be provided, containing thoughtfully-named  
>> and -written routines including versions of Random, Deal, and Shuffle  
>> (and, if there is not one already, a Sorting Unit). The task could be  
>> subcontracted to skilled volunteer labour, such as is found here.*

>

>A new unit for randomizing might be useful. A unit for sorting? What do you  
>want to sort? TList has a Sort method. And shuffling is nothing but a  
>special case of sorting. Try it with a TList and in the compare function  
>return -1 or 1 at random.

Efficient, reliable shuffling is not a special case of sorting. The behaviour of a sort with an inconsistent comparison function is essentially undefined; there may indeed be no guarantee of termination. I do expect that many sorting methods will, with a random comparison function, give shuffled-looking results; but that is not good enough. For a shuffle, there must be a guarantee that, when using a perfect random number source, the probabilities of finding each item in any location must be identical, for a start.

There seems to be, sometimes, a requirement for a Shuffle that leaves everything in a new place; nothing unmoved. IIRC, this is a trivial modification to a good algorithmic shuffle; but I doubt whether a Random Sort can do it.

Dealing a small number of cards from a pack, or awarding a number of prizes to a larger number of candidates, is another matter; a full shuffle is not needed, and neither is draw-at-random,—reject-if—drawn.

TList's Sort provides, I presume, one method of sorting, applicable only to TLists and descendants.

A Sort Unit should provide other methods, including for arrays and other ordered structures. The best Sort for a probably—nearly—ordered set of data is not necessarily the one provided for sorting quasi—random data.

>> *If a reasonably efficient reverse algorithm is known, it should be  
>> provided; it is necessarily an alternative, and reversibility might be  
>> of use.*  
>  
>*There is. For example  $x' = (x * 1664525 + 1013904223) \bmod \$100000000$  can be  
>undone by  $x = (4276115635 * (x' - 1013904223)) \bmod \$100000000$  (that's the  
>numbers I use in my own unit – see below)*

Not necessarily, AFAIK. The random number generating method is not so far specified. I see benefit on offering various methods (we have intelligent linking, and the code is not likely to be large); only one will be reversible by the method you describe, and ISTM that some methods may not be efficiently reversible, or may be efficiently reversible only by unknown methods.

--

© John Stockton, Surrey, UK. ?@merlyn.demon.co.uk Delphi 3 Turnpike 4 ©  
<URL:<http://www.merlyn.demon.co.uk/>> TP/BP/Delphi/&c., FAQy topics & links;  
<URL:<http://www.bancoems.com/CompLangPascalDelphiMisc-MiniFAQ.htm>> clpdmFAQ;  
<URL:<http://www.borland.com/newsgroups/quide.html>> [news:borland](mailto:news:borland). \* Guidelines