Discouraging \texttt{rand()} in C++14, v2

Abstract

In their final Chicago deliberations re \[N3775\] vis-à-vis National Body comment US21, LEWG and LWG achieved joint consensus (1) to deprecate \texttt{std::random_shuffle} for C++14 as proposed, and (2) to strengthen the existing Note in \[c.math]/5 in order to further encourage \texttt{rand()} users to migrate to the \texttt{<random>} component of the C++11 standard library. This paper provides wording to implement these decisions.

1 Background and proposal

If a feature is not deprecated [I] don’t see any point in not using it.

— Hariharan Subramanian

By common consensus at several consecutive WG21 meetings during which the C++11 random number facility was being discussed and shaped into its final form, it has for a number of years been the long-term plan to excise the legacy C random number facility (made up of functions \texttt{rand} and \texttt{srand} and of macro \texttt{RAND_MAX}). Indeed, WG21 voted several years ago to insert a Note\(^1\) into \[c.math]/5 as a head start on this plan: “The random number generation (26.5) facilities in this standard are often preferable to \texttt{rand}.”\(^2\)

Throughout deliberations in Chicago vis-à-vis National Body comment US21, LEWG and LWG independently agreed that we should continue to encourage \texttt{rand()} users to migrate to the \texttt{<random>} component of the C++11 standard library.\(^3\) Taking into account feedback received from WG21, LEWG and LWG achieved a joint final consensus to address US21 by making two adjustments to the text of the C++14 draft standard:

---

\(^1\)This language originated with Beman Dawes in \[N2669\]; \[N2691\] was the first Working Paper to incorporate it.

\(^2\)See also Stephan T. Lavavej’s talk, “\texttt{rand()} Considered Harmful,” given at the GoingNative 2013 event. Recorded on 2013-09-06; available at http://channel9.msdn.com/Events/GoingNative/2013/rand-Considered-Harmful.

\(^3\)Readers seeking greater familiarity with this component may find \[N3551\] to be a helpful source of background information and tutorial guidance with numerous usage examples.
1. Strengthen the existing Note, quoted above, in [c.math]/5.
2. Deprecate `std::random_shuffle` as proposed in [N3775] because “one overload is specified so as to depend on `rand`, while the other overload is specified so as to require a hard-to-produce distribution object from the user; such a distribution is already an implicit part of `shuffle`, which we retain.”

The next section proposes wording to implement both parts of this decision.

2 Proposed wording

(1) Augment [c.math]/5 as shown. (The added wording has been adapted from the introductory section of [N3551].)

5 . . . [Note: The random number generation (26.5) facilities in this standard are often preferable to `rand`, because `rand`’s underlying algorithm is unspecified. Use of `rand` therefore continues to be nonportable, with unpredictable and oft-questionable quality and performance. — end note]

(2) Copy all of the current [alg.random.shuffle] to a new section in Annex D, applying to the copy the changes shown below.

25.3.12 D.x Random shuffle [depr.alg.random.shuffle]

The function templates `random_shuffle` are deprecated.

```cpp
template<class RandomAccessIterator>
void random_shuffle(RandomAccessIterator first, RandomAccessIterator last);

template<class RandomAccessIterator, class RandomNumberGenerator>
void random_shuffle(RandomAccessIterator first, RandomAccessIterator last,
                    RandomNumberGenerator&& randrng);

template<class RandomAccessIterator, class UniformRandomNumberGenerator>
void shuffle(RandomAccessIterator first, RandomAccessIterator last,
             UniformRandomNumberGenerator&& g);
```

**Effects:** Permutes the elements in the range `[first, last)` such that each possible permutation of those elements has equal probability of appearance.

**Requires:** `RandomAccessIterator` shall satisfy the requirements of `ValueSwappable` (17.6.3.2). The random number generating function object `randrng` shall have a return type that is convertible to `iterator_traits<RandomAccessIterator>::difference_type`, and the call `randrng(n)` shall return a randomly chosen value in the interval `[0, n)`, for `n > 0` of type `iterator_traits<RandomAccessIterator>::difference_type`. The type `UniformRandomNumberGenerator` shall meet the requirements of a uniform random number generator (26.5.1.3) type whose return type is convertible to `iterator_traits<RandomAccessIterator>::difference_type`.

**Complexity:** Exactly `(last – first) – 1` swaps.

**Remarks:** To the extent that the implementation of these functions makes use of random numbers, the implementation shall use the following sources of randomness:

---

All proposed **additions** and **deletions** are relative to the post-Chicago Working Draft [N3797]. Editorial notes are displayed against a gray background. We make no recommendation for any SG10 feature-test macro because no feature is being added or removed.
The underlying source of random numbers for the first form of the function is implementation-defined. An implementation may use the \texttt{rand} function from the standard C library.

In the second form of the function, the function object \texttt{randrng} shall serve as the implementation's source of randomness.

In the third form of the function, the object \texttt{g} shall serve as the implementation's source of randomness.

(3) In the synopsis in \texttt{[algorithms.general]}:

- apply the comment //\texttt{Deprecated} to each of the two declarations of \texttt{random_shuffle};
- at the Project Editor's discretion, append to these same declarations a cross-reference to the new Annex D section \texttt{[depr.alg.random.shuffle]};
- change the parameter name \texttt{rand} to \texttt{rng} in the second of the two declarations of \texttt{random_shuffle} so as to avoid confusion with the C library function \texttt{rand}; and
- change the parameter name \texttt{rand} to \texttt{g} in the declaration of \texttt{shuffle} so as to make this declaration consistent with that in \texttt{shuffle}'s later exposition.

(4) Finally, excise vestiges of \texttt{std::random_shuffle} from \texttt{[alg.random.shuffle]} by adjusting as follows:

```cpp
25.3.12 Random\_Shuffle \hfill [alg.random.shuffle]

template<class RandomAccessIterator>
    void random_shuffle(RandomAccessIterator first, RandomAccessIterator last);

template<class RandomAccessIterator, class RandomNumberGenerator>
    void random_shuffle(RandomAccessIterator first, RandomAccessIterator last, RandomNumberGenerator&& rand);

template<class RandomAccessIterator, class UniformRandomNumberGenerator>
    void shuffle(RandomAccessIterator first, RandomAccessIterator last, UniformRandomNumberGenerator&& g);
```

\textbf{Effects:} Permutates the elements in the range \texttt{[first, last)} such that each possible permutation of those elements has equal probability of appearance.

\textbf{Requires:} \texttt{RandomAccessIterator} shall satisfy the requirements of \texttt{ValueSwappable} (17.6.3.2). The random number generating function object \texttt{rand} shall have a return type that is convertible to \texttt{iterator_traits<RandomAccessIterator>::difference_type}, and the call \texttt{rand(n)} shall return a randomly chosen value in the interval \texttt{[0, n)}, for \texttt{n > 0} of type \texttt{iterator_traits<RandomAccessIterator>::difference_type}. The type \texttt{UniformRandomNumberGenerator} shall meet the requirements of a uniform random number generator (26.5.1.3) type whose return type is convertible to \texttt{iterator_traits<RandomAccessIterator>::difference_type}.

\textbf{Complexity:} Exactly \texttt{(last - first)} – 1 swaps.

\textbf{Remarks:} To the extent that the implementation of these \texttt{this} function makes use of random numbers, the implementation shall use the following sources of randomness:

The underlying source of random numbers for the first form of the function is implementation-defined. An implementation may use the \texttt{rand} function from the standard C library.

In the second form of the function, the function object \texttt{rand} shall serve as the implementation's source of randomness.
In the third shuffle form of the function, the object g shall serve as the implementation’s source of randomness.

3 Acknowledgments

Many thanks, for their thoughtful comments, to Stephan T. Lavavej and the other reviewers of early drafts of this paper.

4 Bibliography


5 Revision history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2014-01-01</td>
<td>• Published as N3841.</td>
</tr>
<tr>
<td>2</td>
<td>2014-02-14</td>
<td>• Tweaked proposed wording per LWG guidance. • Published as N3924.</td>
</tr>
</tbody>
</table>