Linear Congruent Engine Correction Proposal

Clarifying Output Range and Template Parameters

Document #:	D0000R1
Date:	2025-04-28
Project:	Programming Language C++
Audience:	Library Evolution Library
Revises:	D0000R0

Reply-to:	Author Juan Lucas Rey <juanlucasrey@gmail.com>

1 Introduction

This proposal highlights and addresses an inconsistency in the `linear_congruential_engine` regarding the specification of its output width compared to other engines like `mersenne_twister_engine`, `subtract_with_carry_engine`, and `philox_engine`.

2 Motivation and Problem Description

Unlike `mersenne_twister_engine`, `subtract_with_carry_engine`, and `philox_engine`, the `linear_congruential_engine` does not include a `w` template parameter specifying the number of significant bits in its output.

As a result, instantiations such as:

```
std::linear_congruential_engine<std::uint_fast32_t, a, c, m>
```

yield output ranges that are **implementation-dependent** based on the actual width of `UIntType`.

This does not alter the sequence of numbers generated internally but affects **how the numbers are interpreted by distributions**.

For example, the behavior of `std::uniform_real_distribution` depends on the range defined by `engine.max()`. Differences in the effective number of bits between implementations can therefore lead to different generated floating-point numbers, despite using identical seeds and parameters.

An illustration of this issue is provided in this test case, where different platforms produce different uniformly distributed real numbers.

3 Proposed Resolution

We propose introducing an optional `w` template parameter to `linear_congruential_engine`, defaulting to the full bit width of `UIntType` if unspecified.

More specifically:

— Define `w` as the number of significant bits in the generated output.
— Specify that `engine.max()` should correspond to $2^w - 1$.
— If `w` is omitted, it should be automatically set to the bit width of `UIntType`.

This change would: - Align `linear_congruential_engine` with other random engines regarding output width specification. - Enable portable behavior across implementations. - Ensure compatibility with the expectations of standard distributions.

4 Impact on Existing Implementations

Existing code that does not explicitly use a `w` parameter would remain valid. The observable behavior (output sequences) for integer types would not change, but floating-point distribution outcomes could be stabilized across platforms.

5 References

— Reference test case highlighting the issue
— cppreference page on `linear_congruential_engine`