Proposal to Consolidate the Subtract-with-Carry Engines

Document number: WG21/N2033= J16/06-0103
Date: 2006-06-22
Project: Programming Language C++
Reference: ISO/IEC IS 14882:2003(E)
Reply to: Walter E. Brown <wb@fnal.gov>
Mark Fischler <mf@fnal.gov>
Jim Kowalkowski <jbk@fnal.gov>
Marc Paterno <paterno@fnal.gov>
CEPA Dept., Computing Division
Fermi National Accelerator Laboratory
Batavia, IL 60510-0500
U.S.A.

This document proposes to modify N2032 = Brown, *et al.*: **Random Number Generation in C++0X: A Comprehensive Proposal, version 2**. In particular, we propose to unify the random number engines `subtract_with_carry_engine<>` and `subtract_with_carry_01_engine<>`. We do so because of these engines' inherently close relationship: they employ the identical transition algorithm as well as the identical generation algorithm. In our opinion, these engines' separation was a historical accident based solely on the types (integer- versus real-valued) of the values each was designed to return. We believe it is unnecessary to provide two distinct engines which are near-identical in all important respects.

By design, the principal role of a random number engine is to serve as a source of random bits for use by a distribution. From this perspective, the actual type in which these bits are packaged is largely irrelevant to this purpose. Since all the standard distributions can cope with integer-valued engines as well as with real-valued engines, and since (at the request of the LWG) N2032 now proposes to standarize the `generate_canonical` template (which handles the essence of this "coping") as an aid to users writing their own distributions, we believe the choice of data type for an engine to deliver can be left to the engine and need not be specified by the programmer.

If adopted, the text labelled 26.4.3.3 in this document would replace the text of N2032's sections 26.4.3.3 and 26.4.3.4. Corresponding changes to the synopsis [26.4.2] and to the engines with predefined parameters [26.4.5] are also intended as indicated, as are the implied changes to the table of contents and to the index. Editorial instructions, not part of the proposed wording, are denoted *like this*.

We would like to acknowledge the Fermi National Accelerator Laboratory's Computing Division, sponsors of our participation in the C++ standards effort, for its support.

Contents

26 Numerics library [lib.numerics]

26.4 Random number generation [lib.random.numbers]

26.4.2 Header <random> synopsis [lib.rand.synopsis]

Replace the synopsis for `subtract_with_carry_engine` *with the following text.*

```
// [26.4.3.3] Class template subtract_with_carry_engine
template <int w, int s, int r>
  class subtract_with_carry_engine;
```

Delete the synopsis for `subtract_with_carry_01_engine`.

Replace the synopses for `ranlux_base_01` *and* `ranlux64_base_01` *with the following text.*

```
typedef see below ranlux24_base;
typedef see below ranlux48_base;
```

Delete the synopses for `ranlux3_01` *and* `ranlux4_01`.

26.4.3 Random number engine class templates [lib.rand.eng]

26.4.3.3 Class template `subtract_with_carry_engine` [lib.rand.eng.sub]

Replace the entire body of this subsection [lib.rand.eng.sub] with the following text.

1 A `subtract_with_carry_engine` random number engine object produces non-negative integer random numbers, if it is integer-valued, or floating-point random numbers, if it is real-valued.

2 The state $\mathtt{x}_i$ of a `subtract_with_carry_engine` object x is of size $\mathcal{O}(\mathrm{r})$, and consists of a sequence X of r integer values $0 \leq X_i < m = 2^w$; all subscripts applied to X are to be taken modulo r. The state $\mathtt{x}_i$ additionally consists of an integer c (known as the *carry*) whose value is either 0 or 1.

3 The state transition is performed as follows:

a) Let $Y = X_{i-s} - X_{i-r} - c$.

b) Set X_i to $Y \bmod m$. Set c to 1 if $Y < 0$, otherwise set c to 0.

[*Note:* This algorithm corresponds to a modular linear function of the form $\mathsf{TA}(\mathtt{x}_i) = (a \cdot \mathtt{x}_i) \bmod b$, where b is of the form $m^r - m^s + 1$ and $a = b - \frac{b-1}{m}$. — *end note*]

4 The generation algorithm is given by:

$$\mathsf{GA}(\mathtt{x}_i) = \begin{cases} y & \text{if x is integer-valued} \\ (y + .25) \cdot 2^{-w} & \text{if x is real-valued} \end{cases},$$

where y is the value of Y mod m produced as a result of advancing the engine's state as described above. [*Note:* For a real-valued engine, adding ε guarantees that the value produced will lie in the required open interval $(0, 1)$. — *end note*]

```
template <int w, int s, int r>
class subtract_with_carry_engine
{
public:
  // types
  typedef unspecified result_type;

  // parameter values and engine characteristics
  static const int bits_of_randomness = w;
  static const int short_lag = s;
  static const int long_lag = r;
  static const see Table 1 min = 0;
  static const see Table 1 max = is_floating_point<result_type>::value ? 1 : m-1;
  static const unsigned long default_seed = 19780503uL;

  // constructors and seeding functions
  explicit subtract_with_carry_engine(unsigned long value = default_seed);
  template <class Gen> explicit subtract_with_carry_engine(Gen& g);
  void seed(unsigned long value = default_seed);
  template <class Gen> void seed(Gen& g);

  // generating functions
  result_type operator()();
};
```

5 The following relations shall hold: $0 < \mathtt{s} < \mathtt{r}$, and $0 < \mathtt{w}$. Further, the value of `w` shall not exceed the largest value of `numeric_limits<T>::digits` corresponding to any fundamental type `T`.

6 The textual representation consists of the textual representations of X_{i-r}, ..., X_{i-1}, in that order, followed by c. The textual representation of each X_k consists of the sequence of $n = \lfloor (\mathtt{w} + 31)/32 \rfloor$ non-negative integer numbers z_0, ..., z_{n-1}, in that order, defined such that $\sum_{j=0}^{n-1} z_j \cdot 2^{32j} = X_k$ with each $z_j < 2^{32}$,

```
explicit subtract_with_carry_engine(unsigned long value = default_seed);
```

7 *Effects:* Constructs a `subtract_with_carry_engine` object as if via a call `subtract_with_carry_engine(g)`, where `g` is a newly-instantiated object constructed as if by the following definition:

```
linear_congruential_engine<unsigned long,
                           40014uL,0uL,2147483563uL> g(value == 0uL ? default_seed : value);
```

```
template <class Gen> explicit subtract_with_carry_engine(Gen& g);
```

8 *Effects:* Sets the values of $X_{-r}, \ldots, X_{-1}$, in that order, as required below. If X_{-1} is then 0, sets c to 1; otherwise sets c to 0.

9 *Required behavior:* To set an X_k, use new values $z_0, \ldots, z_{n-1}$ obtained from n successive invocations of g taken modulo 2^{32}. Set X_k to $\sum_{j=0}^{n-1} z_j \cdot 2^{32j}$ taken modulo m.

10 *Complexity:* Exactly $n \cdot \mathtt{r}$ invocations of g.

26.4.3.4 Class template `subtract_with_carry_01_engine` [lib.rand.eng.sub1]

Delete the heading and entire body of this subsection [lib.rand.eng.sub1].

26.4.5 Engines with predefined parameters [lib.rand.predef]

Delete the `typedefs` and the Required behavior paragraphs for `ranlux_base_01`, `ranlux64_base_01`, `ranlux3`, `ranlux4`, `ranlux3_01`, and `ranlux4_01`; replace them with the following text.

```
typedef subtract_with_carry_engine<24, 10, 24> ranlux24_base;
```

1 *Required behavior:* The 10000^{th} consecutive invocation of a default-constructed object of type `ranlux24_base` shall produce either the value $y = 7937952$ or else the value given by $(y + .25) \cdot 2^{-24}$ according to whether the object is integer- or real-valued, respectively.

```
typedef subtract_with_carry_engine<48, 5, 12> ranlux48_base;
```

2 *Required behavior:* The 10000^{th} consecutive invocation of a default-constructed object of type `ranlux48_base` shall produce either the value $y = 61839128582725$ or else the value given by $(y + .25) \cdot 2^{-48}$ according to whether the object is integer- or real-valued, respectively.

```
typedef discard_block_engine<ranlux24_base, 223, 23> ranlux24;
```

3 *Required behavior:* The 10000^{th} consecutive invocation of a default-constructed object of type `ranlux24` shall produce either the value $y = 9901578$ or else the value given by $(y + .25) \cdot 2^{-24}$ according to whether the object is integer- or real-valued, respectively.

```
typedef discard_block_engine<ranlux48_base, 389, 11> ranlux48;
```

4 *Required behavior:* The 10000^{th} consecutive invocation of a default-constructed object of type `ranlux48` shall produce either the value $y = 249142670248501$ or else the value given by $(y + .25) \cdot 2^{-48}$ according to whether the object is integer- or real-valued, respectively.

Index