A Proposal to Add `typedef default_random_engine` to C++0X

Document #: WG21/N2478 = J16/07-0348
Date: 2007-12-03
Revises: None
Project: Programming Language C++
Reference: ISO/IEC IS 14882:2003(E)
Reply to: Walter E. Brown<wb@fnal.gov>
LSC Dept., Computing Division
Fermi National Accelerator Laboratory
Batavia, IL 60510-0500

Contents

The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them.

— SIR WILLIAM BRAGG

1 Background

Andrei Alexandrescu posted to `comp.std.c++` on 2007-11-07 the following comments about the random number component of the C++0X standard library:

> I think it's the best random number library design of all, by a mile. If I were a random number, I'd think I died and went to heaven.
>
> [However, w]hat many people want is a reasonable (not best!) random number generator, that's not too slow, not too bulky, not too predictable — one that gets the job done. Right now the standard library random library, for all of its stamina, fails to deliver on that.

We agree with Andrei's assessment; it seems reasonable that the standard library provide for its use by non-experts. To address the noted lack of an obvious random number engine for

relatively casual, inexpert and/or lightweight use, this paper proposes an additional `typedef`, `default_random_engine`, for incorporation into C++0X.

2 Discussion

It seems reasonable that the standard library provide for use by non-experts, and to do so by recommending some particular random number engine[1]. To denote the recommended engine, we have selected the identifier `default_random_engine`. But which engine should we adopt for this purpose? As we wrote in a reply to Andrei,

> The problem with trying to decide what might be best for Joe Coder is that there's no way to decide. Sometimes he wants speed of performance, sometimes he wants to minimize size, sometimes he wants the best quality of randomness (which itself can be measured in several different ways). No one engine is uniformly best on all counts, but each of the predefined engines (the `typedef`s in 26.4.5 [rand.predef]) in the Standard has been recognized as "best" according to some of these criteria.

Andrei agreed, saying:

> What you say is correct — that different people have different needs, and as such it's good to have something for each. But it would be a fallacy to think that a default choice prevents people with specific needs [from making] a different choice.

We would therefore like an engine that provides reasonable performance, code size, and quality (*i.e.*, randomness properties). The selected engine need not be the best possible by any of these criteria, but it must be at least acceptable according to each.

While we certainly could select and require a specific engine to meet this need, we firmly believe that the implementor is in the best position to make that selection for each particular platform, in order to take best advantage of that platform's features. We conclude that the best way to achieve our goal is to propose "an implementation-defined typedef for some [engine] that the implementer believes is best for a given platform and implementation."

3 Proposed standard library wording

We propose to augment [rand.predef] by appending the following:

> `typedef` *`implementation-defined`* `default_random_engine;`
>
> 9 *Required behavior:* The named entity shall meet the requirements of Random Number Engine ([rand.req.eng]).
>
> 10 The choice of engine type named by this `typedef` is implementation-defined. [*Note:* The implementation may select this type on the basis of performance, size, quality, or any combination of such factors, so as to provide at least acceptable engine behavior for relatively casual, inexpert, and/or lightweight use. Because different implementations may select different underlying engine types, code that uses this `typedef` need not generate identical sequences across implementations. —*end note*]

[1] Why an engine rather than a URNG? After all, a URNG would suffice on a platform with specialized hardware, and so an implementor could produce the best possible performance (quality/speed). However, while a URNG can be used portably, there is no standard interface to construct/seed it. In contrast, an engine, by design, does have such features.

In addition, the following corresponding line is to be appended to the "engines with predefined parameters" section of [rand.synopsis]:

```
typedef see below default_random_engine;
```

4 Summary and conclusion

This paper has proposed a new `typedef`, `default_random_engine`, for addition to C++. We respectfully urge that the proposal be considered on a time scale consistent with C++0X.

5 Acknowledgments

I thank the Fermi National Accelerator Laboratory's Computing Division, sponsor of our participation in the C++ standards effort, for its past and continuing support of our efforts to improve C++ for all our user communities.