A Proposal to Improve `const_iterator` Use (version 3)

Document #:	WG21/N1913 = J16/05-0173
Date:	2005–10–20
Revises:	WG21/N1865 = J16/05–0125
	WB21/N1674 = J16/04–0114
Project:	Programming Language C++
Reference:	ISO/IEC IS 14882:2003(E)

Reply to: Walter E. Brown<wb@fnal.gov>
CEPA Dept., Computing Division
Fermi National Accelerator Laboratory
Batavia, IL 60510-0500

Contents

1 Introduction **1**

2 Proposal: new container member functions **2**

3 Proposed wording **3**

4 Additional containers **3**

5 Summary **4**

6 Acknowledgments **4**

Bibliography **5**

If you have an apple and I have an apple and we exchange apples then you and I will still each have one apple. But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas.

— George Bernard Shaw

1 Introduction

This paper updates N1865 [Bro05], a proposal to improve user access to the `const` versions of C++ container `iterator`s and their `reverse_iterator` variants[1]. N1865 was discussed and very favorably received by the Library Working Group at the 2005 Mt. Tremblant meeting. In addition to minor editorial improvements, the following changes have been made, largely reflecting the LWG's findings and decisions made during those discussions.

- In the interest of brevity, the entire section on "Motivation" (with subsections "Background," "The problem," "Today's workarounds," and "More examples") has been removed.
- Given LWG's approval of "Alternative 1: new container member functions" and rejection of "Alternative 2: new generic adapter templates," we have removed all mention of Alternative 2, including its proposed wording, as well as all discussion comparing the Alternatives.

[1] The proposal was first set forth in N1674 [Bro04], which was heard by the LWG at the 2005 Lillehammer meeting.

- The previous excisions have made it feasible to consolidate the previous "Proposal" and "Design" sections, and we have done so.
- LWG's preferences as to the names of the new member functions are consistently used throughout, and we have removed the discussion of naming alternatives.
- Given LWG's strong support for the present proposal, we have removed the discussion in which we contrasted this proposal with a prepublication version of Ottosen's "Range Library" proposal[2].
- In light of TR1's recent approval, we have revised some wording to clarify this proposal's impact on what we previously termed the "Additional containers" that that document sets forth.
- We have removed from this proposal all mention of rvalue references [AAD+05]; we no longer believe that the proposals have any impact on each other, especially given the removal of Alternative 2.

Finally, while there was some small LWG controversy regarding the desirability of updating container requirements to reflect the additional member functions provided by this proposal, we have decided to make no changes to our proposal in this regard. We are in general extremely sensitive to issues of code compatibility, but are not persuaded as to the severity of the attendant hardship in this case. We are willing to revisit this issue in future if the LWG so desires, or in connection with future developments related to C++0X concepts[3].

2 Proposal: new container member functions

We believe that the C++ standard library should provide support, absent from C++03, so that a programmer can directly obtain a const_iterator from even a non-const container. **We therefore propose to augment the interface to each C++ standard library container template** with two new pairs of member functions:

```
//                                 Listing 1
const_iterator  cbegin()  const;
const_iterator  cend  ()  const;

const_reverse_iterator  crbegin()  const;
const_reverse_iterator  crend  ()  const;
```

If adopted, this proposal would permit user code to take the form:

```
//                                 Listing 2
vector<MyType> v;
// fill v ...
for( auto it = v.cbegin(), end = v.cend(); it != end; ++it )  {
  // use *it ...
}
```

We find such code very appealing, for it makes clear to a reader that the loop is non-mutating with respect to the container being traversed. Further, use of these proposed member functions in an inappropriate context such as:

[2] This proposal was since published as [Ott05].

[3] Some recent efforts in the rapidly-changing world of C++0X concepts include [Aus05a, CO05, DRS05a, DRS05b, GS05, GSW+05, SDR05].

```
//                                 Listing 3
void reset( double & d )  { d = 0.0; }
void resee( double   d )  { cout << '␣' << d; }
vector<double> v;
// fill v ...
for_each( v.cbegin(), v.cend(), reset ); // oops:  resee intended
```

would now yield a compile-time diagnostic as desired.

This design could, in theory, replace the `const` overloads of the extant container member functions `begin`, `end`, `rbegin`, and `rend`. This is because the proposed functions would subsume these overloads' functionality. However, in order to preserve backwards compatibility, we prefer to retain all present forms of these member functions (although we remain open to the possibility of deprecating their `const` overloads).

3 Proposed wording

The following few additions constitute the necessary changes, with respect to C++03, to standardize our proposal.

3.1 Container requirements

Add the following two new rows to **Table 65—Container requirements** in Clause 23, section [lib.container.requirements]:

expression	return type	assertion/note ...	complexity
`a.cbegin();`	`const_iterator`	`const_cast<X const &>(X).begin();`	constant
`a.cend();`	`const_iterator`	`const_cast<X const &>(X).end();`	constant

3.2 Reversible container requirements

Add the following two new rows to **Table 66—Reversible container requirements** in Clause 23, section [lib.container.requirements]:

expression	return type	assertion/note ...	complexity
`a.crbegin();`	`const_reverse_iterator`	`const_cast<X const &>(X).rbegin();`	constant
`a.crend();`	`const_reverse_iterator`	`const_cast<X const &>(X).rend();`	constant

3.3 Synopses

Add the following four declarations to the *iterators* part of Clause 21, section [lib.basic.string], as well as to the *iterators* parts of Clause 23, sections [lib.deque], [lib.list], [lib.vector], [lib.vector.bool], [lib.map], [lib.multimap], [lib.set], and [lib.multiset]:

```
const_iterator          cbegin() const;
const_iterator          cend() const;
const_reverse_iterator  crbegin() const;
const_reverse_iterator  crend() const;
```

4 Additional containers

While we focus herein on the native and library containers of only C++03, we believe analogous additions would be desirable for homogeneous sequential containers that might in the future be adopted into C++0X. We therefore intend that approval of either or both of the present proposal's Alternatives shall constitute authorization for the Project Editor at an appropriate time to make the comparable obvious additions with respect to those containers.

The unordered associative containers and fixed size arrays in TR1 [Aus05b, clause 6] are prime candidates for such additions. Accordingly, we respectfully recommend that, upon approval of the present proposal, the LWG open an issue to update those parts of TR1 analogously.

5 Summary

This paper has described per-container member functions, the design favored by the Library Working Group, of containers' `begin` and `end` variations: The new member functions' return types are always `const_iterator`s, independent of a container's `const`ness. Proposed wording has been provided to implement the LWG's chosen design.

6 Acknowledgments

I am pleased to acknowledge, with sincere thanks, a number of individuals for their technical comments during the production of this paper and its previous incarnations: Mark Fischler, Chris Green, Jim Kowalkowski, John Marraffino, Thorsten Ottosen, and Marc Paterno. I also wish to thank the Fermi National Accelerator Laboratory's Computing Division, sponsor of our participation in the C++ standards effort, for its support. Finally, many, many thanks to Richard Brown for his careful proofreading of many, many drafts.

Bibliography

[AAD+05] David Abrahams, J. Stephen Adamczyk, Peter Dimov, Howard E. Hinnant, and Andreas Hommel. A proposal to add an rvalue reference to the C++ language: Proposed wording. Paper N1770, ISO/IEC SC22/JTC1/WG21, March 5 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1770.html; same as ANSI NCITS/J16 05-0030.

[Aus05a] Matt Austern. Concept proposal comparison. Paper N1899, ISO/IEC SC22/JTC1/WG21, October 6 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1899.pdf; same as ANSI NCITS/J16 05-0159.

[Aus05b] Matt Austern. (Draft) technical report on standard library extensions. Paper N1836, ISO/IEC SC22/JTC1/WG21, June 24 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1836.pdf; same as ANSI NCITS/J16 05-0096.

[Bro04] Walter E. Brown. A proposal to improve `const_iterator` use from C++0x containers. Paper N1674, ISO/IEC SC22/JTC1/WG21, August 31 2004. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2004/n1674.pdf; same as ANSI NCITS/J16 04-0114.

[Bro05] Walter E. Brown. A proposal to improve `const_iterator` use from C++0x containers (version 2). Paper N1865, ISO/IEC SC22/JTC1/WG21, August 24 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1865.pdf; same as ANSI NCITS/J16 05-0125.

[CO05] Lawrence Crowl and Thorsten Ottosen. Synergies between contract programming, concepts and static assertions. Paper N1867, ISO/IEC SC22/JTC1/WG21, August 24 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1867.html; same as ANSI NCITS/J16 05-0127.

[DRS05a] Gabriel Dos Reis and Bjarne Stroustrup. A formalism for C++. Paper N1885, ISO/IEC SC22/JTC1/WG21, July 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1885.pdf; same as ANSI NCITS/J16 05-0145.

[DRS05b] Gabriel Dos Reis and Bjarne Stroustrup. Specifying C++ concepts. Paper N1886, ISO/IEC SC22/JTC1/WG21, July 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1886.pdf; same as ANSI NCITS/J16 05-0146.

[GS05] Douglas Gregor and Jeremy Siek. Implementing concepts. Paper N1848, ISO/IEC SC22/JTC1/WG21, August 26 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1848.pdf; same as ANSI NCITS/J16 05-0108.

[GSW+05] Douglas Gregor, Jeremy Siek, Jeremiah Willcock, Jaako Järvi, Ronald Garcia, and Andrew Lumsdaine. Concepts for C++0x: Revision 1. Paper N1849, ISO/IEC SC22/JTC1/WG21, August 26 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1849.pdf; same as ANSI NCITS/J16 05-0109.

[Ott05] Thorsten Ottosen. Range library proposal. Paper N1871, ISO/IEC SC22/JTC1/WG21, August 27 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1871.html; same as ANSI NCITS/J16 05-0131.

[SDR05] Bjarne Stroustrup and Gabriel Dos Reis. A concept design (rev. 1). Paper N1782, ISO/IEC SC22/JTC1/WG21, April 2005. Online: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1782.pdf; same as ANSI NCITS/J16 05-0042.