Clause 24 (Iterators) Issues List

David Dodgson
dsd@tr.unisys.com
UNISYS

The following list contains the issues for Clause 24 on Iterators. The list is divided based upon the status of the issues. The status is either *active* - under discussion, *resolved* - resolution accepted but not yet in the working paper, *closed* - working paper updated, or *withdrawn* - issue withdrawn or rejected. They are numbered chronologically as entered in the list. Only the active and resolved issues are presented here. Those wishing a complete list may request one.

The proposed resolutions are my understanding of the consensus on the reflector.

1. *Active Issues*

```
---------------------------------------------------------------------------
 Work Group:     Library Clause 24
 Issue Number:   24-003
 Title:          const operation for iterators
 Section:        24.3
 Status:         active
 Description:
         24.3.1 p24-13  Box 116
    Suggest that the operator *() for STL iterators be made
    into a const operation.

    The function
       void fn (const ReverseIterator & x) {
        ...
           y = x*;
        ...
       }
     shows that the operation * is not defined as const in the
     reverse_iterator (DRAFT 20 Sept 1994, 24.2.1.2).  However, the
     body of the function does not modify the iterator object.

     Of course, const Iterator is different from const_iterator and from
     const const_iterator.

 Proposed Resolution:
    Both base() and operator*() should be const.

 Requestor:      Bob Fraley <fraley@porter.hpl.hp.com>
 Owner:          David Dodgson (Iterators)
 Emails: c++std-lib-3135
 Papers:

---------------------------------------------------------------------------
 Work Group:     Library Clause 24
 Issue Number:   24-008
 Title:          Iterator Requirements
 Section:        24.1.3 and 24.1.4
 Status:         active
 Description:
         24.1.3 Table 59 and 24.1.4 Table 60
   The requirement r == s and r is dereferenceable implies ++r == ++r
```

```
   should read ++r == ++s in table 59.  Similarly in table 60,
   --r == --r implies r == s should read --r == --s.

 Resolution:
    Table 59 for forward iterators was updated.
    Table 60 for bidirectional iterators is not updated.
       It should read: --r == --s implies r == s.
 Requestor:      Nathan Myers
 Owner:          David Dodgson (Iterators)
 Emails: c++std-lib-3543
 Papers:


--------------------------------------------------------------------------
 Work Group:     Library Clause 24
 Issue Number:   24-010
 Title:          Operator-> in Iterators
 Section:        24.
 Status:         active
 Description:
         Throughout clause 24:

The suggestion is for inclusion of operator-> in iterators.

Sean Corfield asks in c++std-lib-3596:

Each iterator has operator*() defined to return T& (or const T& as
appropriate).  Builtin pointer types also have this.

However,builtin pointer types also have operator->() when the underlying
type is a struct/class/union. Is there any reason why iterators don't
have T* operator->() defined? Did we ever decide to delay checking of
the return type of -> to the point of use? I remember we discussed it...
Without this, we have the slightly unpalatable:

  StructThing* p1 = &v1[0];
  StructThing* e1 = &v1[SIZE];
  while (p1 != e1) { process(p1->member); ++p1; }

  vector<StructThing>::iterator p2 = v2.begin();
  vector<StructThing>::iterator e2 = v2.end();
  while (p2 != e2) { process((*p2).member); ++p2; } // ugh!

Bob Fraley and Richard Minner offer agreement,
stating that it is an obvious need and would be extremely confusing
otherwise.

Nathan Myers and Jerry Schwarz dissent,
stating that there are objects for which -> may be meaningless
and that the current interface for iterators is minimal.

John Max Skaller in message c++std-lib-3602 points out that

So I think the question is whether the Standard Library
iterators should, or should not, mandate operator->().
This is not the same question as whether STL should require
operator->().

John Bruns and Fergus Henderson argue in favor of adding operator ->.

Alex Stepanov (and others) argues that operator-> should be provided
for all iterators or none.  Anything else would be too confusing.
Note that this would apply only to iterators over class type.
```

Unresolved questions:
Given an output iterator o what are the semantics of o->member? Since insert iterators and ostream_iterator derive from output iterator, should they define operator->?

Proposed Resolution:
A.
Add the following row in Table 59-Forward iterator requirements in lib.forward.iterators [24.1.3] after the row describing *a:

Expression:	a->m
Semantics:	(a->m == (*a).m)
Conditions:	pre: (*a) refers to a class object and m is a member of that class

B.
Update the predefined iterators to include operator->. Specifically:
lib.reverse.bidir.iter [24.3.1.1]
include operator-> after lib.reverse.bidir.iter.op.star [24.3.1.2.3]
lib.reverse.iterator [24.3.1.3]
after lib.reverse.iter.op.star [24.3.1.4.3]

Requestor: Sean Corfield
Owner: David Dodgson (Iterators)
Emails: lib 3596-3603,lib 3607-3620,3624,3636-3629
Papers:

Work Group: Library Clause 24
Issue Number: 24-012
Title: Addition operators added to iterators
Section: 24.1
Status: active
Description:
24.1.3-24.1.5 p24-3 to 24-6:
Add addition and subtraction operators to non-random iterators.

Alex Stepanov in lib-3611:
And if you reconsider the iterator requirements, you might as well reconsider the exclusion of + (and related operators) for non-random iterator categories. I really hate advance and distance templates. They are such a pain to use and they are really ugly. (To see what I mean, take a look at what we now need to do to implement, say, lower_bound algorithm. It is in algo.h in our implementation.)

Later discussions show that this should not include output iterators, and at most only - operations for input iterators.

Proposed Resolution:
Update Table 59 in lib.forward.iterators to include the row describing r += n, the row describing a + n and n + a, and the row describing b - a from Table 61-Random access iterator.

Update Table 60 in lib.bidirectional.iterators to include the rows describing r -= n and the row describing a - n from Table 61-Random access iterator.

Update Table 61 in lib.random.access.iterators to remove the first five rows (r += n to b - a).

Update description of lib.reverse.bidir.iter [24.3.1.1] to include the + and - operators.

It should be noted that r + n for forward iterators and r - n for bidirectional operators need not be a constant time operation (see 24.1 Iterator requirements para. 8). These operators may be implemented by successive ++r or --r operations.

Update lib.iterator.operations [24.2.6] to note that advance and distance are not required for forward and bidirectional iterators, but that '+ n' and '- n' are the equivalent of advance and that 'b - a' is the equivalent of distance (i.e. linear time operations).

Requestor: Alex Stepanov
Owner: David Dodgson (Iterators)
Emails: lib 3611-3613
Papers:

Work Group: Library Clause 24
Issue Number: 24-013
Title: Const declaration of operator[]
Section: 24.3.1.3 [lib.reverse.iterator]
Status: active
Description:

24..3.1.3 p24-15.16: [Box 117]

Should operator[] of reverse_iterator be specified as const?

Proposed Resolution:

Same resolution as issue 3 (Box 116 in lib.reverse.bidir.iter section 24.3.1.1 for reverse_bidirectional_iterator)

Requestor: Editorial box
Owner: David Dodgson (Iterators)
Emails:
Papers:

Work Group: Library Clause 24
Issue Number: 24-014
Title: Typo
Section: 24.4.3 [lib.istreambuf.iterator]
Status: active
Description:

24.4.3 p24-23

The closing braces for class istreambuf_iterator are in italic bold. They should be in normal font.

Resolution: Use normal font
Requestor: David Dodgson
Owner: David Dodgson (Iterators)
Emails:
Papers:

Work Group: Library Clause 24
Issue Number: 24-015
Title: Char-oriented stream iterators
Section: 24.4.3 [lib.istreambuf.iterator]
Status: active
Description:

24.4.3 p24-23: [Box 118]

The istream_iterator and ostream_iterator are defined only for the char-oriented, but not the wchar_t-oriented or parameterized streams.

```
Resolution:
Requestor:      Editorial Box
Owner:          David Dodgson (Iterators)
Emails:
Papers:


------------------------------------------------------------------------
Work Group:     Library Clause 24
Issue Number:   24-016
Title:          Typo
Section:        24.2 [lib.iterator.primitives]
Status:         active
Description:
        24.2 p24-11:
   The word definable is spelled as 'def inable'
Resolution:
Requestor:      David Dodgson
Owner:          David Dodgson (Iterators)
Emails:
Papers:
```

2. *Resolved Issues*

```
------------------------------------------------------------------------
Work Group:     Library Clause 24
Issue Number:   24-006
Title:          Relaxing Requirement on Iterator++ Result
Section:        24.4.3
Status:         resolved
Description:
        24.4.3 p24-23
   The return type of operator++ for istreambuf_iterator is listed
   as 'proxy'.  This suggestion is to make the return type an object
   which is "convertible to const X&" rather than "X&".
Resolution:     accepted in Austin
Requestor:      Nathan Myers
Owner:          David Dodgson (Iterators)
Emails:
Papers: 95-0021/N0621 (Pre-Austin mailing)


------------------------------------------------------------------------
Work Group:     Library Clause 24
Issue Number:   24-007
Title:          Fixing istreambuf_iterator
Section:        24.4.3
Status:         resolved
Description:
        24.4.3 p24-23:
   Proposes the addition to istreambuf_iterator of
      inline istreambuf::proxy::operator istreambuf_iterator()
             { return sbuf_; }
   to better conform to the Forward Iterator specification.
Resolution:     accepted in Austin
Requestor:      Nathan Myers
Owner:          David Dodgson (Iterators)
Emails:
Papers: 95-0022/N0622 (Pre-Austin mailing)
```

```
--------------------------------------------------------------------------
Work Group:     Library Clause 24
Issue Number:   24-011
Title:          Small Issues in Austin
Section:        24.
Status:         resolved
Description:
        Throughout clause 24
   Numerous small issues as specified in N0614/95-0014 in pre-Austin
   mailing.
Resolution:     Accepted in Austin
   Sections 2.4.6 and 2.4.13 of N0614 regarding the inclusion of friend
   declarations are not included in the April 95 WP (intentional?)

   Sections 2.4.9 and 2.4.10 of N0614 regarding the return type of
   operator++(int) being a reference are not included in the April
   95 WP (intentional?)

Requestor:      Larry Podmolik
Owner:          David Dodgson (Iterators)
Emails: none
Papers: N0614/95-0014 in pre-Austin mailing
```