Doc. No.: WG21/N1024
X3J16/96-0206
Date: November 12, 1996
Project: C++ Standard Library
Reply to: Pete Becker
pbecker@oec.com

Clause 24 (Iterators Library) Motions

Motion (to close various issues without action)

Move we close the following clause 24 issue without taking any action: 24-021 in N1015 = 96-0197.

Motion (to adopt various changes to clause 24) :

Amend the WP as follows, thus closing issue 24-038:
-- strike the text "class proxy;" from the definition of the template class istreambuf_iterator in clause 24.5.3 [lib.istreambuf.iterator]
-- strike the text "proxy operator++(int);" from the definition of the template class istreambuf_iterator in clause 24.5.3 [lib.istreambuf.iterator], and replace it with the following:

```
istreambuf_iterator<charT,traits> operator++(int);
```

-- strike clause 24.5.3.1 [lib.istreambuf.iterator::proxy]
-- strike the following text from clause 24.5.3.4 [lib.istreambuf.iterator::op++]

```
proxy istreambuf_iterator<charT,traits>::operator++(int);
Returns: proxy(sbuf_->sbumpc(), sbuf_).
```

and replace it with

```
istreambuf_iterator<charT,traits>
        istreambuf_iterator<charT,traits>::operator++(int);
Effects:    istreambuf_iterator<charT,traits> tmp = *this;
            sbuf_->sbumpc();
            return (tmp);
```

Amend the WP as follows, thus closing issue 24-042:
-- strike the text "insert_iterator<Container> operator++(int);" from the definition of template class insert_iterator in clause 24.4.2.5 [lib.insert.iterator] and replace it with the following:

```
insert_iterator<Container>& operator++(int);
```

-- strike the text "insert_iterator<Container> operator++(int);" from clause 24.4.2.6.4 [lib.insert.iter.op++] and replace it with the following:

```
insert_iterator<Container>& operator++(int);
```

-- strike the text "ostream_iterator<T,charT,traits> operator++(int);" from the definition of template class ostream_iterator in clause 24.5.2 [lib.ostream.iterator] and replace it with the following:

```
ostream_iterator<T,charT,traits>& operator++(int);
```

-- strike the text “ostreambuf_iterator operator++(int);” from the definition of the template class ostreambuf_iterator in clause 24.5.4 [lib.ostreambuf.iterator] and replace it with the following:

ostreambuf_iterator& operator++(int);

-- strike the text “ostreambuf_iterator<charT,traits> operator++(int);” in clause 24.5.4.2 [lib.ostreambuf.iter.ops] and replace it with the following:

ostreambuf_iterator<charT,traits>& operator++(int);

Amend the WP as set out in N0910 = 96-0092, thus closing issue 24-044.

Amend the WP as follows, thus closing issue 24-045:

-- add the following private members to the definition of the template class istream_iterator in clause 24.5.1 [lib.istream.iterator]:

private:
  basic_istream<charT,traits>* in_stream; exposition only
  T value; exposition only

-- add immediately after clause 24.5.1 [lib.istream.iterator] the following new clauses:

24.5.1.1 istream_iterator constructors and destructor

istream_iterator();

Effects: Constructs the end-of-stream iterator.

istream_iterator(istream_type& s);

Effects: Initializes in_stream with s. value may be initialized during construction or the first time it is referenced.

istream_iterator(const istream_iterator<T,Distance>& x);

Effects: Constructs a copy of x.

~istream_iterator();

Effects: The iterator is destroyed.

24.5.1.2 istream_iterator operations

const T& operator*() const;

Returns: value

const T* operator->() const;

Returns: &(operator*())

istream_iterator<T,Distance>& operator++();

Effects: *in_stream >> value
Returns: *this

istream_iterator<T,Distance> operator++(int);

Effects:
  istream_iterator<T,Distance> tmp = *this;
  *in_stream >> value;
  return (tmp);

template <class T, class Distance>
 bool operator==(const istream_iterator<T,Distance>& x,
          const istream_iterator<T,Distance>& y);

Returns: (x.in_stream == y.in_stream)

-- add the following private members to the definition of the template class ostream_iterator in clause 24.5.2 [lib.ostream.iterator]:

private:
  basic_ostream<charT, traits> out_stream; exposition only
  const char* delim; exposition only

-- add immediately after clause 24.5.2 [lib.ostream.iterator] the following new clauses:

24.5.2.1 ostream_iterator constructors and destructor

ostream_iterator(ostream_type& s);

Effects: Initializes out_stream with s and delim with null.

ostream_iterator(ostream_type& s, const charT* delimiter);

Effects: Initializes out_stream with s and delim with delimiter.

ostream_iterator(const ostream_iterator<T>& x);

Effects: Constructs a copy of x.

~ostream_iterator();

Effects: The iterator is destroyed.

24.5.2.2 ostream_iterator operations

ostream_iterator<T>& operator=(const T& value);

Effects:
  *out_stream << value;

```
    if (delim != 0) *out_stream << *delim;
    return (*this);
```

ostream_iterator<T>& operator*();

Returns: *this

ostream_iterator<T>& operator++();
ostream_iterator<T>& operator++(int);

Returns: *this

Amend the WP as follows:
-- strike the text "{ TBS }" from Table 86 in clause 24.1.5 [lib.random.access.iterators] and replace it with the following:

(a<b) ? distance(a,b) : -distance(b,a)