Doc. No.: J16/97-0098R1
WG21/N1136R1
Date: 12 November 1997
Project: Programming Language C++
Reply to: Beman Dawes
beman@esva.net

Library Motions for Morristown

Motion (to resolve outstanding Library issues):

Move we:

A) Amend the WP as described in N1111R1 = 97-0073R1 by adopting the proposed resolutions described below:

1. London edit change needs correction: [lib.global.names] 17.3.3.1.2 Global names
2. USA CD2-17-002 Portable C Lib function linkage
4. UK 626, Sweden _20451 Auto_ptr semantics: *Accept resolution A.8* only
13. Japan Technical 3 Clause 21.1.2/1 Table 37 and Clause 21.1.2/ 2
15. US CD2-23-015 Is X:: reverse_iterator convertable to X::const_reverse_iterator?
16. UK 699 template member function ambiguities (p105); should apply to more widely
19. Japan "Editorials" 14 26.2.6 paragraph 10 (p156)
20. Library typename syntax and related issues
21. Ratification of certain London changes *(No change to WP)*
22. Class codecvt<intenT, externT, stateT> change questioned*(No change to WP)*
23. Class reverse_iterator<Iterator> iterator_type questioned *(No change to WP)*
24. Res of CD2-20-006 Incomplt; mem_fun adaptors can't be used w/const mbr funcs
25. Fallout from CD2-27-042 (22.2.2.2)
26. [lib.limits] 18.2.1 Numeric limits specializations need to define static mbr constants
28. Missing public in ostreambuf_iterator (24.5.4)
29. basic_string::reserve() unclear
31. Containers overspecify size_type and difference_type
32. vector<bool> pointer and const_pointer problem
34. Input iterator operator* return type
36. Template class fpos<class stateT> ambiguous constructors
37. Typedef event_callback invalid throw() *(but change "cannot" to "must not")*
38. basic_filebuf::seekpos must call unshift
39. Where is basic_iostream declared?
40. Clause 27 Inconsistent with regards to typedefs
42. D.7 Description of freeze() is contradictory *(select resolution option 2)*
44. Return from first call to std::set_new_handler unclear
45. iterator_traits<> underspecification for nested types
46. Stream iterators missing default argt for charT
53. Core vs library editorial conflicts (5.3.4 vs 18.4) *(purely editorial; appears here only as a reminder)*

B) Amend the WP to resolve issues from N1111R1 = 97-0073R1 by adopting the resolutions described below.

3. Germany "Editorials" 17.2.2.1.2/2 bitmask

As described in the proposed resolution except use using static_casts

5. Sweden _2133/b Return of capacity(), US CD2-21-004 Capacity() desc unclear

To 23.2.4.2 lib.vector.capacity add:

Throws: length_error if n>max_size().

To both 23.2.4.2 lib.vector.capacity and 21.3.3 lib.string.capacity add a footnote:

reserve() uses Allocator::allocate() which may throw an appropriate exception.

17. CD2-23-018 Container typedefs unimplementable

Add a new paragraph after paragraph 1 of section 21.3 [lib.basic.string].

The template parameter traits must be a traits class whose char type is charT, and the template parameter Allocator must be an allocator class whose value type is charT.

Modify paragraph 8 of section 23.1 ([lib.container.requirements]). (The change is one word in the first sentence and a new clause in the second sentence.)

Copy constructors for all container types defined in this clause copy an allocator argument from their respective first parameters. All other constructors for these container types take an Allocator& argument, an allocator (20.1.5) whose value type is the same as the container's value type. A copy of this argument is used for any memory allocation performed, by these constructors and by all member functions, during the lifetime of each container object. In all container types defined in this clause, the member get_allocator() returns a copy of the Allocator object used to construct the container.

Change two lines of the header <map> synopsis at the top of section 23.3 [lib.associative]. Change the declaration of class map to

```
template <class Key, class T, class Compare = less<Key>,
        class Allocator = allocator<pair<const Key, T> > >
  class map;
```

and the declaration of class multimap to

```
template <class Key, class T, class Compare = less<Key>,
        class Allocator = allocator<pair<const Key, T> > >
  class multimap;
```

Change the class map synopsis in section 23.3.1 [lib.map] in the same way. The change is in the third line of the synopsis. The first four lines of the synopsis now read:

```
namespace std {
  template <class Key, class T, class Compare = less<Key>,
          class Allocator = allocator<pair<const Key, T> > >
  class map {
```

Change the class multimap synopsis in section 23.3.2 [lib.multimap] in the same way. The change is in the third line of the synopsis. The first four lines of the synopsis now read:

```
namespace std {
```

```
template <class Key, class T, class Compare = less<Key>,
        class Allocator = allocator<pair<const Key, T> > >
class multimap {
```

18. Japan "Editorials" 13 Clause 26.2.2 (p156) resolution:

    Insert the missing operators for 26.2.1

27. [lib.limits] 18.2.1 Numeric limits has_denorm undeterminable until runtime

    Add one line to header <limits> synopsis in 18.2.1 [lib.limits].
    Immediately after

    ```
    enum float_round_style;
    ```

    add the line

    ```
    enum float_denorm_style;
    ```

    In section 18.2.1.1 [lib.numeric.limits], change the line

    ```
    static const bool has_denorm = false;
    ```

    to

    ```
    static const float_denorm_style has_denorm = denorm_absent;
    ```

    In section 18.2.1.2 [lib.numeric.limits.members], change paragraph 40 (and the line of code just before it) to read

    ```
    static const float_denorm_style has_denorm;
    ```

    Denorm_present if the type allows denormalized values (variable number of exponent bits).(194), denorm_absent if the type does not allow denormalized_values, and denorm_indeterminate if it is indeterminate at compile time whether the type allows denormalized values.

    Add a new section between 18.2.1.3 [lib.round.style] and 18.2.1.4 [lib.numeric.special]

    18.2.1.4 Type float_denorm_style [lib.denorm.style]

    ```
    namespace std {
      enum float_denorm_style {
        denorm_indeterminate = -1,
        denorm_absent = 0,
        denorm_present = 1
      };
    }
    ```

    The presence or absence of denormalization (variable number of exponent bits) is characterized by the values:

    -- denorm_indeterminate if it cannot be determined whether the type allows denormalized values
    -- denorm_absent if the type does not allow denormalized values
    -- denorm_present if the type does allow denormalized values

33. Input iterator assignment return type resolution:

    Change 24.1 table to an input iterator assignment return type of T&.

41. Library uses illegal default args in <iosfwd> resolution:

> Default arguments to act as if supplied several places. (Jerry Schwarz to provided detailed wording).

47. For_each overspecified?

> Remove 25.1.1 paragraph 2, which begins "Requires….."

51. basic_ifstream & basic_ofstream prototype errors

> In 27.8.1.8, remove "| trunc" and "| ios_base::trunc"

C) Amend the WP by accepting the resolution proposed by 97-0090 = N1128, Fixing auto_ptr, by Gibbons and Colvin.

D) Amend the WP by accepting the resolution proposed by 97-0094 = N1132, Item 38, basic_filebuf::seekpos must call unshift, by Myers and Plum.

E) Amend the WP by changing 27.4.2.4 sync_with_stdio returns clause to read:

> True if the standard iostream objects (lib.iostream.objects) are synchronized and otherwise returns false. The first time it is called, the function returns true.

F) Amend the WP by accepting the proposed resolutions in 97-0093 = N1131, Library Glitches, by Becker.

G) Amend the WP by accepting the initial portion of the proposed resolution in 97-0075 = N1113, Swapping of Containers, by Nico Jusuttis. The portion accepted is the proposed resolution up to but not including the portion which begins "All references, pointers and iterators to the elements of a swapped…"

H) Amend the WP by accepting the proposed resolutions in 97-0076 = N1114, Making the C++ Standard Library More Exception Safe, by Abrahams and Colvin.

I) To resolve the issue raised by UK issue 679, amend the WP by adding the following wording to lib.associative.reqmnts 23.1.2 paragraph 7:

> c is a value of type `X::key_compare`.

J) To resolve Japan "Editorial" issues 7 21.1.1 paragraph 1, 9 21.1.3, 10 21.1.3 paragraph 1, and 11 Clause 21.1.3 (Morristown items 8, 10, 11, 12), amend the WP as follows:

> In section 21.1.3[lib.char.traits.typedef] replace "Requires" paragraphs for OFF_TYPE and POS_TYPE by
>
> > requirements on OFF_T and POS_T are described in 27.1.2.1[lib.iostream.limits.pos]
>
> In section 21.1.3[lib.char.traits.typedef] replace "Requires" paragraph for STATE_T by
>
> > STATE_T shall satisfy the CopyConstructible requirements of 20.1.3[lib.copyconstructible]
>
> In 21.1[lib.char.traits]/3 add "Traits::char_type shall be the same as charT."