Doc. No.:	X3J16/96-0093
	WG21/ N0911
Date:	May 28, 1996
Project:	Programming Language C++
Reply To:	Richard K. Wilhelm
	Strategic Technology Resources
	rwilhelm@str.com

Clause 21 (Strings Library) Issues List
Revision 16

Revision History

Version 1 - January 30, 1995: Distributed in pre-Austin mailing.
Version 2 - March 6, 1995: Distributed at Austin meeting.
Version 3 - March 24, 1995: Distributed in post-Austin mailing. Several issues added. Several issues updated to reflect decisions at Austin meeting.
Version 4 - May 19, 1995: Distributed in pre-Monetery mailing.
Version 5 - July 9, 1995: Distributed at the Monterey meeting. Includes many issues added from public comments.
Version 6 - July 11, 1995: Distributed at the Monterey meeting. Added no new issues from previous version. Included issues prepared for formal vote. Added solutions for issues 8, 21,31, 38, 69, 71. Made only changes to reflect the decisions of the string sub-group, correct working paper text and to correct typographical errors.
Version 7 - July 27, 1995: Distributed in the post-Monterey mailing. Reflects the resolutions and discussions of the Monterey meeting.
Version 8 - September 24, 1995: Distributed in the pre-Tokyo mailing. Some new issues added.
Version 9 - November 2, 1995: Distributed at the Tokyo meeting. Added issue 79. Added solutions for issues: 29, 30, 61, 62, and 63.
Version 10 - November 8, 1995: Distributed at the Tokyo meeting. Contains resolutions for issues to be closed by a vote.
Version 11 - December 2, 1995: Distributed in the post-Tokyo mailing. Updated issues closed in Tokyo. Added several new issues
Version 12 - January 29, 1996: Distributed in the pre-Santa Cruz mailing.
Version 13 - March 10, 1996: Distributed at the Santa Cruz meeting.
Version 14 - March 13, 1996: Distributed at the Santa Cruz meeting. Reflects changes to resolutions make by the library group.
Version 15 - March 28, 1996: Distributed in the post-Santa Cruz mailing. Updated issues closed in Santa Cruz.
Version 16 - May 28, 1996: Distributed in the pre-Stockholm mailing.

Introduction

This document is a summary of the issues identified in Clause 21. For each issue the status, a short description, and pointers to relevant reflector messages and papers are given. This evolving document will serve as a basis of discussion and historical record for Strings issues and as a foundation of proposals for resolving specific issues.

For clarity, active issues are separated from issues recently closed. Closed issues are retained for one revision of the paper to serve as a record of recent resolutions. Subsequently, they will be removed from the paper for brevity. Any issue which has been removed will include the document number of the final paper in which it was included.

Active Issues

Issue Number: 21-062
Title: Missing explanation of requirements on `charT`.
Section: 21.1.1.3 [lib.basic.string]
Status: active
Description:

A public comment noted:
Paragraph 1 doesn't say enough about the properties of a "char-like object." It should say that it doesn't need to be constructed or destroyed (otherwise, the primitives in string_char_traits are woefully inadequate). string_char_traits::assign (and copy) must suffice either to copy or initialize a char-like element. The definition should also say than an allocator must have the same definitions for the types size_type, difference_type, pointer, const_pointer, reference, and const_reference as class allocator::types<charT> (again because string_char_traits has no provision for funny address types).

Proposed Resolution:

Add the following text after paragraph 1 in 21.1.1.3 [lib.basic.string]

> A "char-like type" does not need to be constructed or destroyed. A string's allocator shall have the same definitions for the types `size_type`, `difference_type`, `pointer`, `const_pointer`, `reference`, `const_reference` as class `allocator::types<charT>`.

In private email, P.J. Plauger wrote:
"In reviewing my code, I realize that I overstated the case here. It is more accurate to say that the basic_string class presumes that charT has a default constructor (and a destructor), which the class uses to construct (and destroy) all elements of the controlled sequence. Whenever the class is asked to copy out elements, as with the copy member function, it assumes that it need only assign to previously constructed elements.

"A better design of string_char_traits would probably include uninitialized_copy and uninitialized_fill members, but I feel it's way too late to propose such additions."

Requester: Public comment T21 (p. 108).
Owner:
Emails: (none)
Papers: (none)

Issue Number: 21-085
Title: Awkward argument order for basic_string traits.
Section: 21.1.1.2 [lib.string.char.traits.members]
Status: active
Description:

Two string_char_traits members have the following signatures:

```
static const char_type*
find(const char_type* s, int n, const char_type& a)
```

```
static char_type*
assign(char_type* s, size_t n, const char_type& a)
```

The semantics of these members emulate memchr() and memset(). However, the argument order is slightly different. In the interest of consistency, the order of these arguments should be corrected.

Additionally, change the type of the `find()` member's '`n`' argument to `size_t`

Proposed Resolution:

In section 21.1.1.2 [lib.string.char.traits.members] change the signatures of find() and assign() as follows:

```
static const char_type*
find(const char_type* s, const char_type& a, size_t n)

static char_type*
assign(char_type* s, const char_type& a, size_t n)
```

Requester: LWG
Owner:
Emails: (none)
Papers: (none)

Issue Number: 21-090
Title: operator>> consuming whitespace
Section: 21.1.1.10.8 [lib.string.io]
Status: active
Description:

From a public comment:
"It seems to me that, to be useful, operator>>() must eat zero or more delimiters specified by basic_string<...>::traits::is_del() prior to reading each string. This should be specifed in the standard, to prevent varying implementations. If that is not the committee's intent, it should be explicitly stated in the standard what the intent is."

Judy Ward (j_ward@decc.enet.dec.com) commented that operator>> should call is.ipfx() not is.ipfx(true); calling ipfx(true) does not skip white space.

Proposed Resolution:

In 21.1.1.10.8 [lib.string.io], change the call to `is.ipfx(true)` to `is.ipfx()`.

Requester: John Mulhern (jmulhern@empros.com).
Owner:
Emails: (none)
Papers: (none)

Issue Number: 21-092
Title: Incorrect description for traits::compare()
Section: 21.1.1.2 [lib.string.char.traits.members]
Status: active
Description:

At the end of the second sentence of the member's description, the description of the range for `i` is incorrectly stated as `[0, n)`.

Proposed Resolution:

In the second sentence of the description for traits::compare(), change:

…and for each `i` in the range `[0, n)`…

to:

…and for each `i` in the range `[0, j)`…

Requester: Judy Ward (j_ward@decc.enet.dec.com).
Owner:
Emails: (none)
Papers: (none)

Issue Number: 21-093
Title: Clarify return value for traits::find()
Section: 21.1.1.2 [lib.string.char.traits.members]
Status: active
Description:

The description of the function does not define what should be returned if the character cannot be found

Proposed Resolution:

Add the following to the end of the Returns section:

`…, zero otherwise`

Requester: Judy Ward (j_ward@decc.enet.dec.com).
Owner:
Emails: (none)
Papers: (none)

Issue Number: 21-094
Title: Description for operator>> does not cleanse string
Section: 21.1.1.10.8 [lib.string.io]
Status: active
Description:

The description of the stream extraction operator states:

The function extracts characters and appends them to *str* as if by calling *str*.append(1, *c*).

This incorrectly implies that extracting into a string which already contains data would append the data to the string

Proposed Resolution:

In the description for operator>>() in 21.1.1.10.8 [lib.string.io] change:

The function extracts characters and appends them to *str* as if by calling *str*.append(1, *c*).

to:

The string is initially made empty by calling str.erase(). Then the function extracts characters and appends them to *str* as if by calling *str*.append(1, *c*).

Requester: Judy Ward (j_ward@decc.enet.dec.com).
Owner:
Emails: (none)
Papers: (none)

Issue Number: 21-095
Title: basic_string::getline() cannot set the count in basic_istream
Section: 21.1.1.10.8 [lib.string.io]
Status: active
Description:

The description of the getline() member states:

The function ends by storing the count in `is`…

There is no basic_istream member which would allow this to happen.

Proposed Resolution:
None yet.
Requester: Judy Ward (j_ward@decc.enet.dec.com).
Owner:
Emails: (none)
Papers: (none)

Issue Number: 21-096
Title: Add several headers to basic_string
Section: 21.1 [lib.string.classes]
Status: active
Description:
The declaration of the basic_string template does not include all headers required.

Proposed Resolution:
Add the following headers to the declaration of basic_string:

```
#include <stdexcept>
#include <iterator>
#include <locale>
#include <cwchar>
#include <cwctype>
```

(The addition of iterator has also been recommended by the German delegation.)
Requester: Judy Ward (j_ward@decc.enet.dec.com).
Owner:
Emails: lib-4691
Papers: (none)

Closed Issues

Issues which have been recently closed are included in their entirety. Issues which have appeared in a previous version of the issues list as “closed” have the bulk of their content deleted for brevity. The document number of the paper in which they last appeared is included in parentheses for reference.

21-001 Should basic_string have a getline() function? (N0721=95-0121)
21-002 Are string_traits members char_in() and char_out() necessary? (N0815=95-0215)
21-003 Character-oriented assign function has incorrect signature (N0721=95-0121)
21-004 Character-oriented replace function has incorrect signature (N0759=95-0159)
21-005 How come the string class does not have a prepend() function? (N0759=95-0159)
21-006 Should the Allocator be the last template argument to basic_string? (N0721=95-0121)
21-007 Should the string_char_traits speed-up functions be specified as inline? (N0759=95-0159)
21-008 Should an iostream inserter and extractor be specified for basic_string? (N0759=95-0159)
21-009 Why are character parameters passed as “const charT”? (N0721=95-0121)
21-010 Should member parameters passed as “const_pointer”? (N0721=95-0121)
21-011 Why are character parameters to the string traits functions passed by reference? (N0721=95-0121)
21-012 Why are character parameters to the string functions passed by value? (N0800=95-0200)
21-013 There is no provision for errors caused by implementation limits. (N0815=95-0215)

21-014 Argument order for copy() is incorrect. (N0899=96-0081)
21-015 The copy() member should be const. (N0759=95-0159)
21-016 The error conditions are not well-specified for the find() and rfind() functions. (N0759=95-0159)
21-017 Can `reserve()` cause construction of characters? (N0815=95-0215)
21-018 Specification of traits class is constraining. (N0815=95-0215)
21-019 The `Allocator` template parameter is not reflected in a member typedef. (N0759=95-0159)
21-020 Header for Table 42 is incorrect. (N0759=95-0159)
21-021 `compare()` has unexpected results (N0759=95-0159)
21-022 s.append('c') appends 99 nulls. (N0759=95-0159)
21-023 Non-conforming default `Allocator` arguments (N0759=95-0159)
21-024 Name of traits delimiter function is confusing (N0815=95-0215)
21-025 Does `string_char_traits` need a locale? (N0815=95-0215)
21-026 Description of `string_char_traits::compare()` is expressed in code. (N0815=95-0215)
21-027 Description of `string_char_traits::compare()` overspecifies return value. (N0815=95-0215)
21-028 Description of `string_char_traits::length()` is expressed in code. (N0815=95-0215)
21-029 Description of `string_char_traits::copy()` is overconstraining. (N0815=95-0215)
21-030 Description of `string_char_traits::copy()` is silent on overlapping strings. (N0815=95-0215)
21-031 Copy constructor takes extra argument to switch allocator but does not allow allocator to remain the same. (N0815=95-0215)
21-032 Description for operator+() is incorrect (N0759=95-0159)
21-033 Requirements for `const charT*` arguments not specified (N0759=95-0159)
21-034 Inconsistency in requirements statements involving npos (N0815=95-0215)
21-034a Expand ability to throw length_error (N0815=95-0215)
21-035 Character replacement does not change length. (N0759=95-0159)
21-036 Character case disregarded during common operations. (N0759=95-0159)
21-037 Traits needs a move() for overlapping copies. (N0815=95-0215)
21-038 Operator < clashes cause ambiguity (N0759=95-0159)
21-039 Iterator parameters can get confused with size_type parameters. (N0759=95-0159)
21-040 Repetition parameter non-intuitive (N0759=95-0159)
21-041 Assignment operator defined in terms of itself (N0759=95-0159)
21-042 Character assignment defined in terms of non-existent constructor (N0759=95-0159)
21-043 Character append operator defined in terms of non-existent constructor (N0759=95-0159)
21-044 Character modifiers defined in terms of non-existent constructor (N0759=95-0159)
21-045 Iterator typenames overspecified (N0759=95-0159)
21-046 basic_string type syntactically incorrect in some descriptions (N0759=95-0159)
21-047 Error in description of replace() member (N0759=95-0159)
21-048 Inconsistency in const-ness of compare() declarations (N0759=95-0159)
21-049 Inconsistency constructor effects and semantics of data() (N0759=95-0159)
21-050 Incorrect semantics for operator+() (N0759=95-0159)
21-051 Incorrect return type for insert() member (N0759=95-0159)
21-052 Unconstrained position arguments for find members. (N0759=95-0159)
21-053 Semantics of size() prevents null characters in string (N0759=95-0159)
21-054 Change the semantics of length() (N0759=95-0159)
21-055 append(), assign() have incorrect requirements (N0759=95-0159)
21-056 Requirements for insert() are too weak. (N0759=95-0159)
21-057 replace has incorrect requirements (N0759=95-0159)
21-058 Description of data() is over-constraining. (N0759=95-0159)

21-059 String traits have no relationship to iostream traits. (N0899=96-0081)
21-060 string_char_traits::ne not needed (N0815=95-0215)
21-061 Missing explanation of traits specialization (N0815=95-0215)
21-063 No constraints on constructor parameter. (N0815=95-0215)
21-064 Miscellaneous errors in resize(size_type n) (N0759=95-0159)
21-065 Incorrect return value for insert() (N0759=95-0159)
21-066 Description of remove() is over-specific (N0759=95-0159)
21-067 Traits specializations are over-constrained for `eos()` member (N0815=95-0215)
21-068 What is the proper role of the "Notes" section in Clause 21. (N0815=95-0215)
21-069 Swap complexity underspecified. (N0759=95-0159)
21-070 operator>= described incorrectly (N0759=95-0159)
21-071 Does getline() have the correct semantics? (N0759=95-0159)
21-072 Incorrect use of size_type in third table in section (N0759=95-0159)
21-073 Add overloads to functions that take default character object. (N0759=95-0159)
21-074 Should `basic_string` have a member semantically equivalent to `strlen()` (N0815=95-0215)
21-075 Incomplete specification for assignment operator (N0800=95-0200)
21-076 Inconsistent pattern of arguments in basic_string overloads (N0815=95-0215)
21-077 basic_string not identified as a Sequence. (N0815=95-0215)
21-078 Possible problem with reference counting and strings. (N0815=95-0215)
21-079 Possible problem with `operator<<()` (N0815=95-0215)
21-080 Allow template specialization for `basic_string` and `string_char_traits`?
21-082 Typedef for reverse_iterator is incorrect. (N0899=96-0081)
21-083 Traits member eos() is not forced to return the same value every time. (N0899=96-0081)
21-084 Specialize swap() algorithm for basic_string. (N0899=96-0081)
21-086 New type added to table (N0899=96-0081)
21-087 Different return values for index operations (N0899=96-0081)
21-088 Slight glitch in return value for find() (N0899=96-0081)
21-089 Should basic_string have a release() member. (N0899=96-0081)
21-091 More specific description for capacity() and reserve() (N0899=96-0081)