Document Number:	J16/97-0035
	WG21/N1073
Date:	29 May 1997
Project:	Programming Language C++
Reply-To:	Steve Rumsby
	steve@maths.warwick.ac.uk

UK Issues List

Compiled by

Steve Rumsby

steve@maths.warwick.ac.uk

Introduction

This document is a list of all problems found by the BSI C++ panel (IST/5/-/21) in the C++ ballot document N2356.
The issues and their descriptions have been contributed by many of the panel members — the author of this document is simply the keeper of the database. The "Comments" section of each issue is a container for miscellaneous commentary and discussion of the issue, and is not part of the issue itself.

Issue Number: 654 *WP Section:* 17-27

Creation Date 07/03/97 *Severity:* Major

Category: *Related Issues:*

Owner: Andy Sawyer

Summary: Library resource usage policy

Description: We would like better guarantees that library components release resources when appropriate.

For memory in particular, we request that all library components that allocate memory via an allocator allocate() function be required to deallocate that memory via the corresponding deallocate() function. Further we require that the points at which such deallocation occur be defined, and should be the "obvious" places when memory is no longer required.

Comments:

Issue Number: 629 *WP Section:* 20.4.5 [lib.auto.ptr]

Creation Date 18/10/96 *Severity:* Major

Category: *Related Issues:*

Owner: Steve Rumsby

Summary: auto_ptr broken

Description: The copy semantics of auto_ptr are undesirable, and dangerous. Make the copy constructor and copy assignment operator of auto_ptr private, and delete the conversion constructor and conversion assignment operator templates. Alternatively, remove auto_ptr from the library.

Comments:

Issue Number:	669	*WP Section:*	23
Creation Date	07/04/97	*Severity:*	Major
Category:		*Related Issues:*	
Owner:	Steve Rumsby		
Summary:			

Description: There are numerous unfortunate ambiguities involving template member functions, including construstors, taking iterator ranges. These ambiguities occur when attempting to call a non-template function with reasonable argument types that happen to be a better match for the template version. These ambiguities must be removed.

For example, see this posting from Sanda Whitman to the library reflector:

I ran into a problem with vector constructors when using a compiler with member function template support. According to CD2 vector has the following constructors:

```
explicit vector(size_type n, const T& value = T(),
            const Allocator& = Allocator());

template <class InputIterator>
vector (InputIterator first, InputIterator last,
      const Allocator& = Allocator());
```

When I create a vector<int> v2(1,0) the compiler is matching on the member function vector (InputIterator first, InputIterator last, ...). I had not expected this. In my environment size_t is an unsigned long so vector (InputIterator first, InputIterator last ...) does seem the better match. Is this a problem in my compiler, or in the library specification? I noticed a similar thing in basic_string but haven't checked any other containers yet.

Comments:

Issue Number:	229	*WP Section:*	01.1 [intro.scope] ¶2
Creation Date	27/06/95	*Severity:*	Minor
Category:	Misc	*Related Issues:*	Core issue
Owner:	Derek Jones		
Summary:			

Description: Delete the last sentence and Annex C.1.2. This is the first standard for C++, what happened prior to 1985 is not relevant to this document.

Comments:

Issue Number:	573	*WP Section:*	01.8 [intro.exectution] ¶6,7
Creation Date	15/09/95	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	John Borthwick		
Summary:			

Description: Should the word "volatile" in these two paragraphs actually be the keyword "volatile", i.e. typeset in courier? This would seem to be exactly the purpose for while "volatile" is intended.

Comments:

Issue Number: 263 | *WP Section:* 01.8 [intro.execution] ¶9
Creation Date 03/07/95 | *Severity:* Minor
Category: Misc | *Related Issues:* Core issue
Owner: Derek Jones
Summary:
Description: What is a "needed side-effect"? Should this be "observable side-effect", using "observable" as defined in paragraph 6? Also, delete the parenthetical remark.
Comments:

Issue Number: 273 | *WP Section:* 02.3 [lex.trigraph] ¶3
Creation Date 03/07/95 | *Severity:* Minor
Category: Editorial | *Related Issues:*
Owner: Derek Jones
Summary:
Description:

> [Note: no other trigraph sequence exists. Each ? that does not begin one of the trigraphs listed above is not changed.]

This is a normative statement and should not be a note.
Comments:

Issue Number: 673 | *WP Section:* 03.03.05 [basic.scope.namespace], ¶1
Creation Date 09/04/97 | *Severity:* Minor
Category: Example bug | *Related Issues:*
Owner: Lois Goldthwaite
Summary:
Description: In the example, within function

```
int g( char a ) { return k+a; }
```

Should ‘k’ be ‘l’ ?
Comments:

Issue Number: 421 | *WP Section:* 05 [expr] ¶4
Creation Date 01/08/95 | *Severity:* Minor
Category: Editorial | *Related Issues:*
Owner: Derek Jones
Summary:
Description: In the example, change "undefined" to "unspecified".
Comments:

Issue Number:	89	*WP Section:*	05.2.1 [expr.sub]
Creation Date	24/09/94	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Sean Corfield		
Summary:			

Description: "One of the expressions ... and the other must be of enumeration or integral type."
Since enumeration types convert to integral types isn't this overspecified? In fact, shouldn't it be "convertible to an integral type" to allow the following:

```
struct X { operator int(); };
X x;
char buf[];
buf[x];
```

(the same applies to "must have the type 'pointer to T'" -- this should be "convertible to a pointer to object type")

Comments: Josée: "enumeration or integral type" is overspecified. I think "expression that can be converted to a pointer to T" should also be included.

Issue Number:	488	*WP Section:*	05.3.3 [expr.sizeof] ¶6
Creation Date	01/08/95	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Derek Jones		
Summary:			

Description: Replace this paragraph with:

> The result is a constant with type size_t, as defined in the standard header <cstddef>

Comments:

Issue Number:	493	*WP Section:*	05.3.4 [expr.new] ¶3
Creation Date	01/08/95	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Derek Jones		
Summary:			

Description: This requirement is very poorly worded. The following example all contain parentheses, but some are regarded as being well formed. Reword the requirement more precisely.

Comments:

Issue Number:	632	*WP Section:*	06.4 [stmt.select], ¶1
Creation Date	18/10/96	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Kevlin Henney		
Summary:	Declarations in conditions		

Description: What's the point of preventing legit declarations that choose not to (or cannot) use the '=' syntax? Eg.

```
if(file some_file("filename")) // op bool returns true on opening
{
    ...
}
```

This constraint seems to be in error given that it offers complete support for the concepts that it intends to support.

Comments:

Issue Number:	114	*WP Section:*	07.3.2 [namespace.alias] ¶3
Creation Date	03/11/94	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Sean Corfield		
Summary:			

Description: Change

in that declarative region to refer to the namespace to which it already refers.

to

in that declarative region to refer only to the namespace to which it already refers.

The old wording only says what is possible, and not what isn't possible.

Comments:

Issue Number:	612	*WP Section:*	16.2 [cpp.include], ¶4
Creation Date	26/02/96	*Severity:*	Minor
Category:	Typo	*Related Issues:*	
Owner:	Richard De Morgan		
Summary:			

Description: Change:

.)If

to:

.) If

Comments:

Issue Number:	682	*WP Section:*	17 [lib.library]
Creation Date	09/04/97	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Lois Goldthwaite		
Summary:			

Description: Throughout this chapter, there are references both to the "C++ Standard library" and the "C++ Standard Library". 17.3.4.7 [lib.derivation] even manages to use both forms in the same sentence. We believe one usage or the other should be used consistently. We prefer "C++ Standard library".

Comments:

Issue Number:	648	*WP Section:*	17.1 [lib.definitions]
Creation Date	28/02/97	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Alan Stokes		
Summary:			

Description: In [lib.definitions], at the end of the definition of 'reserved functions' there is a missing para break before "Clause intro.defs ...".

Comments:

Issue Number:	684	*WP Section:*	17.2.2.1.3.3, ¶1
Creation Date	09/04/97	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Lois Goldthwaite		
Summary:			

Description: A reference to (_basic.fundmental_) should be spelled basic.fundamental (3.9.1)

Comments:

Issue Number:	658	*WP Section:*	17.3.1.2 [lib.headers], ¶4
Creation Date	07/04/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Alan Stokes		
Summary:			

Description: In [lib.headers], para 4 gives a general rule for each header which begins with a c. However the rule is probably not intended to apply to header complex, and it's not clear it doesn't (except that C doesn't have a header omplex.h).

Comments:

Issue Number:	649	*WP Section:*	17.3.1.3 [lib.compliance], ¶2
Creation Date	28/02/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Alan Stokes		
Summary:			

Description: In [lib.compliance] para 2 "has has" should be just "has".

Comments:

Issue Number:	696	*WP Section:*	18.4.1.1[lib.new.delete.single], ¶3,7
Creation Date	09/04/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Sean Corfield		
Summary:			

Description:

> 3 Required behavior: Return a pointer to dynamically allocated storage
> (3.7.3), or else throw a bad_alloc exception.

If this is a requirement on user-supplied versions it is too restrictive. It precludes, for example, a version that manages memory
from a static array. Change to "Return a non-null pointer to suitably aligned storage (3.7.3), or else throw a bad_alloc exception."

Note: 3.7.3 does not use the term "dynamically allocated storage".

> 7 Required behavior: Return a pointer to dynamically allocated storage
> (3.7.3), or else return a null pointer. This nothrow version of operator
> new returns a pointer obtained as if acquired from the ordinary version.
> This requirement is binding on a replacement version of this function.

Similarly, change to "Return a pointer to suitably aligned storage (3.7.3), or else return a null pointer."

Should "This requirement is..." be present on para 3 above? I think so.

Comments:

Issue Number: 697 | *WP Section:* 18.4.2.1 [lib.bad.alloc], ¶2-4
Creation Date 09/04/97 | *Severity:* Minor
Category: | *Related Issues:*
Owner: Sean Corfield
Summary:
Description:

2 Effects: Constructs an object of class bad_alloc.

3 Effects: Copies an object of class bad_alloc.

4 Notes: The result of calling what() on the newly constructed object is implementation-defined.

Reorder paras 3 and 4 (so the note applies to the construction, not the copy operations).

Comments:

Issue Number: 664 | *WP Section:* 18.5.1 [lib.type.info]
Creation Date 07/04/97 | *Severity:* Minor
Category: | *Related Issues:* 665
Owner: Alan Stokes
Summary:
Description: The result of name() may be a NTBS, but may not. So you can't portably use it. So what's the point? Surely it's not unreasonable to require it to be an NTBS - an implementation can always return the null string if it's not feeling friendly. The returned string should at least be null-terminated. The return from "what()" gives this guarantee - why not "name()"?

Comments:

Issue Number: 699 | *WP Section:* 18.5.2 [lib.bad.cast], ¶2-4
Creation Date 09/04/97 | *Severity:* Minor
Category: | *Related Issues:*
Owner: Sean Corfield
Summary:
Description:

2 Effects: Constructs an object of class bad_cast.

3 Effects: Copies an object of class bad_cast.

4 Notes: The result of calling what() on the newly constructed object is implementation-defined.

Swap paras 3 and 4.

Comments:

Issue Number: 701 *WP Section:* 18.5.2 [lib.bad.exception], ¶2-4
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Sean Corfield
Summary:
Description:

> 2 Effects: Constructs an object of class bad_exception.
>
> 3 Effects: Copies an object of class bad_exception.
>
> 4 Notes: The result of calling what() on the newly constructed object is implementation-defined.

Swap paras 3 and 4.

Comments:

Issue Number: 700 *WP Section:* 18.5.2 [lib.bad.typeid], ¶2-4
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Sean Corfield
Summary:
Description:

> 2 Effects: Constructs an object of class bad_typeid.
>
> 3 Effects: Copies an object of class bad_typeid.
>
> 4 Notes: The result of calling what() on the newly constructed object is implementation-defined.

Swap paras 3 and 4.

Comments:

Issue Number: 665 *WP Section:* 18.6.1 [lib.exception]
Creation Date 07/04/97 *Severity:* Minor
Category: *Related Issues:* 664
Owner: Alan Stokes
Summary:
Description: The result of what() is "An implementation-defined NTBS", but ".. may be a NTBS, ...". These two seem to contradict each other. what() should certainly return an NTBS.
Comments:

Issue Number: 692 *WP Section:* 20.1.5 [lib.allocator.requirements]
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Sean Corfield
Summary:
Description: The Table 32 footnote currently reads:

> 204) It is intended that a::allocate be an efficient means of allocating a single object of type T, even when sizeof(T) is small. That is, there is no need for a container to maintain its own ‘‘free list’’.

This should read either "... a.allocate ..." or "... X::allocate…"

Comments:

Issue Number: 675 *WP Section:* 20.1.5 [lib.allocator.requirements], Table 32
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Lois Goldthwaite
Summary:
Description: Should x.construct() and x.destroy() refer to 'a' instead of 'x'?
Comments:

Issue Number: 641 *WP Section:* 20.3.2
Creation Date 27/02/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Kevlin Henney
Summary:
Description: We have negate for -x, but what about identity for +x? For completeness, identity is an important functional concept.

Part of the motivation is, as mentioned, completeness and another is that it provides an ideal default template parameter that developers can use, e.g. for providing a filter operation on some other kind of container/process/whatever.

I have since discovered that SGI's implementation of STL has an identity type that does exactly this. In contacting Matt Austern he wondered whether it was worth raising as a comment on CD2.

Compose is also missing?
Comments:

Issue Number: 676 *WP Section:* 20.3.3 [lib.comparisons], ¶1
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Lois Goldthwaite
Summary:
Description: "In all cases, type T is convertible to type bool."
I'm not sure I understand what this means in context. Does this imply that only integral types can be used with these function objects? That doesn't seem logical. Delete the sentence.
Comments:

Issue Number: 693 *WP Section:* 20.3.5 [lib.negators]
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Sean Corfield
Summary:
Description:

```
template <class Predicate>
  class unary_negate : public unary_function<Predicate::argument_type,bool>
  ...

template <class Predicate>
  class binary_negate : public binary_function<Predicate::first_argument_type,
Predicate::second_argument_type, bool>
  ...
```

I *think* both of these require 'typename Predicate::...argument_type'
Comments:

Issue Number: 694 *WP Section:* 20.3.6.1 [lib.binder.1st]
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Sean Corfield
Summary:
Description:

```
template <class Operation>
  class binder1st : public unary_function<Operation::second_argument_type, Operation::result_type>
  ...
  Operation::first_argument_type value;
  ...
```

Should be 'typename Operation::...'

Comments:

Issue Number: 695 *WP Section:* 20.3.6.3 [lib.binder.2nd]
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Sean Corfield
Summary:
Description:

```
template <class Operation>
  class binder2nd : public
unary_function<Operation::first_argument_type, Operation::result_type>
  ...
  Operation::second_argument_type value;
  ...
```

Should be 'typename Operation::...'

Comments:

Issue Number: 704 *WP Section:* 20.4.4.3 [lib.uninitialized.fill.n]
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Steve Rumsby
Summary:
Description: Change From:

```
  while (n--)
        new (static_cast<void*>(&*result++))
              typename iterator_traits<ForwardIterator>::value_type(*first++);
```

To:

```
  while (n--)
        new (static_cast<void*>(&*first++))
              typename iterator_traits<ForwardIterator>::value_type(x);
```

Comments:

Issue Number: 639 *WP Section:* 21.3.4 [lib.string.access]
Creation Date 27/02/97 *Severity:* Minor
Category: Clarification *Related Issues:*
Owner: Sean Corfield
Summary: Sub-optimal lifetime of reference returned from operator[]
Description: Invalidating the reference from operator[]() on any non-const member function call is clearly broken since s[i] = s[j]; is undefined.
Comments:

Issue Number:	671	*WP Section:*	21.3.5.2 [lib.string::append]
Creation Date	07/04/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Andy Sawyer		
Summary:			

Description: basic_string::append(charT) - any chance of aliasing this as basic_string::push_back(charT), thus enabling the use of
back_insert_iterators on basic_strings? As far as I can see, append() satisfies all the requirements of push_back, thus enabling the use of back_insert_iterator/back_inserter with strings. FWIW, this came out of a conversation I was having earlier today with a colleague who needed this functionality in a real-world application - whilst it is fairly trivial to specialise
back_insert_iterator for basic_string, it would be nice not to have to.

It's worth noting that this does not add any additional functionality to basic_string, merely improves the 'cohesiveness' of the standard library - i.e. makes it feel less like it was designed by a committee...:-)

Comments:

Issue Number:	677	*WP Section:*	23.1 [lib.container.requirements], Table 66
Creation Date	09/04/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Lois Goldthwaite		
Summary:			

Description: Should "distance type" be "difference type"?

Comments:

Issue Number:	680	*WP Section:*	23.1.2 Table 70
Creation Date	09/04/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Lois Goldthwaite		
Summary:			

Description:

a.clear() means erase(s.begin, s.end))
Shouldn't this be "erase(a.begin(), a.end())"?

Comments:

Issue Number:	678	*WP Section:*	23.1.2 [lib.associative.reqmts], ¶2
Creation Date	09/04/97	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Lois Goldthwaite		
Summary:			

Description: Change:

Each associative containers...

to:

Each associative container...

Comments:

Issue Number:	679	*WP Section:*	23.1.2, ¶7
Creation Date	09/04/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Lois Goldthwaite		
Summary:			

Description: Add "c is a function object or function pointer"?

Comments:

Issue Number: 691 *WP Section:* 23.3.1 [lib.map]
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Lois Goldthwaite
Summary:
Description: Change:

```
    typedef Allocator::pointer pointer;
    typedef Allocator::const_pointer const_pointer;
```

to:

```
    typedef typename Allocator::pointer pointer;
    typedef typename Allocator::const_pointer const_pointer;
```

Comments:

Issue Number: 681 *WP Section:* 23.3.4 [lib.multiset], ¶2
Creation Date 09/04/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Lois Goldthwaite
Summary:
Description: "a_eu" should be "a_eq".
Comments:

Issue Number: 642 *WP Section:* 24.4.1.1 [lib.reverse.iterator]
Creation Date 27/02/97 *Severity:* Minor
Category: *Related Issues:*
Owner: Kevlin Henney
Summary:
Description: I don't know if this problem has been picked up yet, but in [lib.reverse.iterator] in the November WP the changes made to have reverse_iterator derive from iterator are not complete:

```
template <class Iterator>
class reverse_iterator :
    public iterator<
        iterator_traits<Iterator>::iterator_category,
        iterator_traits<Iterator>::value_type,
        iterator_traits<Iterator>::difference_type,
        iterator_traits<Iterator>::pointer,
        iterator_traits<Iterator>::reference>
{
    ...
    reverse_iterator operator+ (Distance n) const;
    reverse_iterator& operator+=(Distance n);
    reverse_iterator operator (Distance n) const;
    reverse_iterator& operator=(Distance n);
    Reference operator[](Distance n) const;
};
```

The argument type for the members listed above needs to be changed to difference_type. and the return type of operator[]() needs to be changes to reference.

Might this be considered an editorial change?

Comments:

Issue Number: 688
Creation Date 09/04/97
Category:
Owner: Lois Goldthwaite
Summary:
Description: Two references to "pred" should be "binary_pred"; two references to "first" should be "first1"
Comments:

WP Section: 25 [lib.algorithms], ¶8
Severity: Minor
Related Issues:

Issue Number: 687
Creation Date 09/04/97
Category:
Owner: Lois Goldthwaite
Summary:
Description:

7 The function object pred _is assumed not_ to
apply any non-constant function through the dereferenced iterator.

8 binary_pred _shall not_ apply any non-constant function
through the dereferenced iterators.

Why the difference between "is assumed not" and "shall not"?

Comments:

WP Section: 25 [lib.algorithms], ¶8,9
Severity: Minor
Related Issues:

Issue Number: 689
Creation Date 09/04/97
Category: Editorial
Owner: Lois Goldthwaite
Summary:
Description: Change:

In these cases the semantics of a+n is the same is that of ...

To:

In these cases the semantics of a+n is the same as that of ...

Comments:

WP Section: 25 [lib.algorithms], ¶9
Severity: Minor
Related Issues:

Issue Number: 702
Creation Date 09/04/97
Category:
Owner: Andy Sawyer
Summary:
Description: Clause 25 [lib.algorithms], Para 9 contains the following:
..

```
{
  Distance n;
  distance( a, b, n );
  return n;
}
```

but distance (24.3.4 [lib.iterator.operators]) no longer takes three parameters (thanks to iterator_traits). Should this now be:

```
return distance(a,b)
```

Comments:

WP Section: 25 [lib.algorithms], ¶9
Severity: Minor
Related Issues:

Issue Number:	613	*WP Section:*	26.2.2 [lib.complex]
Creation Date	26/02/96	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Andy Sawyer		
Summary:			

Description: The declaration of class complex is missing a closing brace & semi-colon. (It's actually the namespace std that's missing it)

Comments:

Issue Number:	706	*WP Section:*	26.3.1 [lib.valarray.synopsis]
Creation Date	09/04/97	*Severity:*	Minor
Category:	Editorial	*Related Issues:*	
Owner:	Steve Rumsby		
Summary:			

Description: Change From: "- for every functions taking a const"
To: "- for every function taking a const"

Comments:

Issue Number:	705	*WP Section:*	26.3.9.2 [lib.indirect.array.assign]
Creation Date	09/04/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Steve Rumsby		
Summary:			

Description: Change From: "array[indirect] = b;"
To: "a[indirect] = b;"

Comments:

Issue Number:	633	*WP Section:*	27
Creation Date	18/10/96	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Kevlin Henney		
Summary:			

Description: [lib.ifstream] 27.8.1.5 Template class basic_ifstream
[lib.ofstream] 27.8.1.8 Template class basic_ofstream
[lib.fstream] 27.8.1.11 Template class basic_fstream

Both of these define is_open to be a non-const function that returns rdbuf()->is_open(). Why are they non-const? The filebuf is_open is const, and logically the stream query should also be const.

Comments:

Issue Number:	634	*WP Section:*	27.4.2 [lib.ios.base]
Creation Date	17/01/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Kevlin Henney		
Summary:	The destructor for ios_base is non-virtual.		

Description: The destructor for ios_base is non-virtual. The issue here is not one of deletion safety but of polymorphism: ios_base is a non-polymorphic base class, and thus dynamic_cast cannot be used w/ it. The problem arises if developers wish to use the event_callback mechanism safely: it is not possible to get from the ios_base& argument to the correct derived class if needed for a callback.
Solution: Make the dtor virtual.

Comments:

Issue Number:	635	*WP Section:*	27.4.2.1.1 [lib.ios::failure]
Creation Date	17/01/97	*Severity:*	Minor
Category:		*Related Issues:*	Library issue 27-001
Owner:	Kevlin Henney		
Summary:			

Description: The problem I posted a while back with the exception classes having the postcondition what() == msg.c_str() has been fixed for <stdexcept> classes, but I (and indeed the library group) missed the change required for ios_base::failure. To refresh memories, the expression uses pointer equality rather than content equality. In the case of ios_base::failure the problem is made worse by referring to a string member function that does not exist (str() is used instead of c_str())!
Solution: Change "what() == msg.str()" to "strcmp(what(), msg.c_str()) == 0".

Comments:

Issue Number:	636	*WP Section:*	27.4.2.7 [lib.ios.base.cons]
Creation Date	17/01/97	*Severity:*	Minor
Category:		*Related Issues:*	
Owner:	Kevlin Henney		
Summary:			

Description: Change:

The ios_base members are have an indeterminate value after construction.

to:

The ios_base members have indeterminate values after construction.

Comments:
