Document Number: WG21/N0880
X3J16/96-0062
Date: 29 May 1996
Project: Programming Language C++
Reply to: Dan Saks
dsaks@wittenberg.edu

X3J16 Meeting No. 20
WG21 Meeting No. 15
11 - 15 March 1996

Dream Inn Resort
175 West Cliff Drive
Santa Cruz, CA 95060 USA

1 Opening activities

Clamage convened the meeting as chair at 09:10 (PST) on Monday, 11 March 1996. Lajoie was the vice-chair, and Saks was the secretary.

Borland (represented by Becker) hosted the meeting.

1.1 Opening comments

1.2 Introductions

Saks circulated an attendance list each day, which is attached as Appendix A of these minutes. Lajoie circulated a copy of the membership list (SD-2) for members to make corrections.

1.3 Membership, voting rights, and procedures for the meeting

Clamage explained who has voting rights in X3J16: an X3J16 member organization may vote at this meeting if it has paid its dues and has met X3's attendance requirements. New X3J16 members may not vote at this meeting.

1.4 Distribution of position papers, WG progress reports, WG work plans for the week, and other documents not distributed before the meeting

1.5 Approval of the minutes of the previous meeting

Saks submitted the minutes from the previous meeting (N0817 = 95-0217). Myers asked to change the sentence on page 18 (under item 6.2) which read:

> "Myers said it was too late to change this."

to:

> "Myers said the Library WG decided it was too late to change this."

Motion by Dawes/Miller:

> Move we approve N0817 = 95-0217 as the minutes of the previous meeting with this correction.

Motion passed X3J16: lots yes, 0 no, 0 abstain.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

1.6 Report on the WG21 Sunday meeting

Harbison summarized the outcome of Sunday's WG21 meeting. He said WG21

identified three issues that were of particular concern to some delegations:
-- the template compilation model
-- extended characters in identifiers and literals
-- the general state of the library
He said WG21 discussed ways to insure that we will address these issues adequately at this meeting.

Harbison also explained that we have the precedents we need to publish the CD electronically to the public during the CD ballot (for example, by placing it in a public ftp site). It is not clear that it's okay to do this for other drafts or for any other committee documents.

1.7 Agenda review and approval

Clamage submitted the proposed agenda (N0825 = 96-0007) for approval.

Clamage said there are two technical issues that need discussion, and he wanted to ensure that the agenda gave interested parties ample time to discuss them. Those issues are: (1) the template compilation models, and (2) the interaction of locales with streams and strings.

The committee considered how to partition the library work among the library WGs (working groups). Dawes said we need to keep the locale and i/o issues together. Schwarz agreed to chair that group. Dawes agreed to chair the WG handling issues of the library introduction, language support, diagnostics, utilities, strings and containers. Becker and Dodgson agreed to take turns leading the work on algorithms, iterators, and grudgingly, numerics.

Lajoie said the core issues would be handled by the usual three subgroups, chaired by Adamczyk, Lajoie, and Gibbons. Gibbon's Core WG will address the template compilation model. Lajoie said that, in the likely event that WG can't reach a decision, we should expand the discussion to include wider participation. Thus, we should plan to hold a technical session on the template compilation model on Monday evening.

Plum said he'd chair the C compatibility WG. They would work on the extended character issue, as well as a few others.

Plum explained that the US will cast its formal vote on approving the CD after SC22 distributes the ballots to all national bodies. Plum said he needs a US TAG meeting on Thursday so he can poll the TAG to determine how he should vote on Friday on submitting the WP as a CD. He asked Clamage to add this agenda item:

> US TAG meeting at 7:30 pm Thursday.

Motion by Lajoie/Ball:

> Move we accept N0825 = 96-0007 with this addition as the agenda for this meeting.

Motion passed X3J16: lots yes, 0 no, 0 abstain.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

1.8 Liaison reports

1.8.1 WG14+X3J11 (C)

Plum reported that WG14 and X3J11 met in February in Irvine, CA hosted by Unisys. They are working on the revision of the C standard, which they refer to as C9X.

Plum said WG14+X3J11 adopted the following for inclusion in C9X:
-- Variable-length arrays (VLAs, as per Cray and maybe GNU)

-- Compound literals (as per Plan 9)
-- <inttypes.h> header which defines, for example, int32 (as per HP)
-- two-slash comments
-- empty macro arguments (for macro M, M() used to be undefined; now it means one argument whose value is the empty string)
-- rules for tag compatibility across translation units

They are also considering, but have not adopted:
-- deprecating implicit int
-- inline functions (as in C++)
-- the restrict keyword
-- extended characters in identifiers and literals
-- initializer repetition counts
-- bool, true and false (compatible with C++, but as macros, not keywords)
-- low-level I/O functions for embedded systems (as per Kristofferson)
-- anonymous structs and unions
-- long long types
-- classes (as per Jervis from Sun Microsystems)
-- signed integer division specification (compatible with Fortran?)
-- floating-point extensions (FPCE from NCEG)
-- complex arithmetic (as per Cray, with IEEE in an appendix)

Plum explained that C currently allows two models for signed integer division. WG14+X3J11 wants to pick one, and it looks like they'll pick the one compatible with Fortran.

Plum also explained that WG14+X3J11 is considering two models for complex arithmetic. The Cray proposal uses a simpler model and is more like C++. The alternate model attempts to get every corner case compatible with the IEEE specification. WG14+X3J11 will probably accept the simpler model as the minimum requirement for C9X, but add an appendix allowing the other model.

Stroustrup said that, having invented anonymous unions, he now thinks it was a mistake. Anonymous structs are even worse.

1.9 New business requiring actions by the committee

Lajoie announced she's resigning as vice-chair of X3J16. Clamage thanked her for doing such an outstanding job. Resounding applause. Clamage asked for volunteers to be the new vice-chair.

1.10 Drafting committee

Saks briefly explained that the drafting committee is responsible for preparing the formal motions in advance of the voting so that all committee members have the opportunity to understand the issues and consider how they will vote. Lajoie reminded the WG chairs that someone from each WG must bring that WG's motions to, and participate in, the drafting committee.

1.11 Organization of WGs

WG21+X3J16 prepared to break into WGs.

The committees recessed to WGs at 10:35 and reconvened on Wednesday at 8:40.

2 WG sessions

3 Technical session

4 WG sessions

5 Working Paper for Draft Proposed Standard

Koenig presented the project editor's report. He said there was no editing session immediately after the Tokyo meeting, but plenty of people (Dawes, Lajoie, Myers, Podmolik, and Wilhelm) helped with the editing in the weeks that followed.

Koenig said he made numerous "bold changes" since the last WP. (A bold change is one that incorporates the effect of an editorial box without a specific vote by the committees.) He listed the bold changes, using the notation B(C) to mean "editorial box B in clause C": 11(3.9), 12(4.10), 31(14.10.2), 48(21.1.1.10.8), 49(21.2), 50(21.1.1.1.1), 54(21.2.1.1), 55(22.2.1.5.1), 56(22.2.1.5.2), 58(22.2.3.1.1), 59(22.2.5.1.2), 61(24.1.6), 68(24.4.3), 70(24.4.3.2), 71(24.4.3.2), 82(27).

6 General session I

6.1 Core Language WG

==== Lajoie ====

Lajoie presented a proposal to clarify lookup for a name appearing immediately after a . or -> (issue 452a from N0863 = 96-0045). (See Motion 2 under agenda item 11.1 for the precise wording.)

Gibbons asked if this just confirms a previous decision. Lajoie said it does, but the words in subclause 5.2.4 don't account for namespaces. She said we could consider this as an editorial change.

Lajoie presented a proposal to clarify the syntax and semantics for pseudo-destructors for scalar types (issue 433 from N0863 = 96-0045). The proposal clarifies that T in p->~T() can be a typedef-name (as well as a class-name). The translator looks up T both in the context of the full expression and in the scope of the class of *p. It must find T in either scope. If it finds T in both scopes, both declarations must refer to the same type.

The proposal also clarifies name lookup for an expression such as p->T1::~T2(). In this case, the translator looks up T1 and T2 in the scope of the full expression. T1 and T2 must refer to the same type, namely, the type of *p. (See Motion 3. A later proposal from Pennello also contributed to Motion 3.)

Lajoie then presented a proposal to clarify the semantics for members defined within anonymous unions (issues 570 and 105 from N0863 = 96-0046). (See Motion 4.) One consequence of the proposal is that an anonymous union member name must be different from every other name declared in the scope enclosing the anonymous union. For example,

```
class C { };
union {
    int C;  // error
};
```

Another consequence is that, after an anonymous union definition, members of that union appear as if declared in the scope enclosing the anonymous union. For example,

```
namespace N {
    static union {
        int i;
    }
}
N::i;       // ok
```

Next, Lajoie presented a proposal to clarify the use of incomplete types in class templates (item 3.1 from N0851 = 96-0033). Specifically, she proposed that members of a class template may have incomplete type at

the point where the template is defined. For example,

```
class A;
template <class T> class X {
    A a;    // ok, even if A does not depend on the template argument
};
```

Unruh noted that, at present, a template for which no specialization can be generated, is ill-formed. Thus, the template above is ill-formed.

Lajoie then presented a proposal to clarify lookup of class members inside a class definition (item 5.1 from N0851 = 96-0033). (See Motion 5.)

Lajoie presented a proposal to allow arrays of incomplete class type (from discussion raised by Merrill over the Core e-mail reflector). She gave this example:

```
class X;

extern X x;     // ok

extern X ax[3]; // currently an error; recommend it should be ok
```

(See Motion 6 for details.)

Lajoie also explained that, you can write

```
struct S;
S *p;
free(p);
```

in C. Hence, the WG also wants to allow

```
delete p;
```

Next, Lajoie presented a proposal to disallow

```
const class { };
```

(See issue 116 from N0865 = 96-0047). She explained that

```
static class C { };
```

is already ill-formed according to subclause 7.1.1. (See Motion 7.)

Then Lajoie presented a proposal to constrain the width of bit-fields (issue 47 from N0865 = 96-0047). She explained the proposal with this example:

```
struct S {
    char bit : 16;  // error: too many bits
    bool b2 : 32;   // ok
};
```

The WG proposed that the number of bits in a bit-field declaration should not be greater than the number of bits needed for the object representation of the bit-field's type. However, if the bit-field has bool or enumeration type, the number of bits should not be greater than the number of bits needed for the object representation of the underlying type.

Ball and Gibbons thought this was too restrictive. Koenig and others countered that limiting the size of enum bit-fields is good because it prevents bit-fields from holding undefined values.

Straw vote: Who agrees with this proposal? 18 yes, "about the same" no.

Lajoie agreed to take the issue back to the WG.

Stump asked Lajoie to describe the status quo regarding bit-field sizes. Lajoie said it's unclear.

Lajoie then presented a proposal to specify the representation of bit-fields that have bool or enumeration type (issues 623 and 458 from N0865 = 96-0047). She explained that, under this proposal, given:

```
enum B { f = 0, t = 1 };
struct S {
    B b1: 1;
    bool b2 : 1;
};
S s;
s.b1 = t;
s.b2 = true;
```

then both (s.b1 == t) and (s.b2 == true) yield true. (See Motion 8.)

Gibbons offered this informal summary of the proposal: "bools and enums are neither signed nor unsigned, and they must be able to store the values they are required to store."

Finally, Lajoie presented a proposal to allow static members in POD structs and unions (issue 568 from N0865 = 96-0047). (See Motion 9.)

Lajoie said she assumed there was no objection to those issues for which we didn't hold straw votes.

6.2 Syntax WG

Pennello presented four issues regarding C++ syntax.

Pennello explained that the operand of prefix ++ and -- must be an lvalue. However, the grammar in the current WP does not allow expressions such as ++(int &)x (see issue 593 from N0862 = 96-0044). He presented a proposal to change the grammar so that a cast-expression (not a unary-expression) is the operand of prefix ++ and --. (See Motion 10.)

Pennello said this change "raises" right recursion from unary-expression to cast-expression everywhere in the production for unary-expression except in sizeof expressions. Pennello said the WG considered merging cast-expression with unary-expression; however, doing so introduces an ambiguity in expressions such as sizeof (T) * x. Currently that expression means (sizeof (T)) * x; merging the non-terminals would also allow sizeof ((T) * x) as a valid parse.

No one objected to this proposal.

Pennello explained that ~X() is currently ambiguous (issue 512 from N0862 = 96-0044). It means either

```
this->~X()      // Destructor call
```

or

```
~(X())          // operator~ applied to X()
```

He said the WG compiled this expression using several compilers (Borland, cfront, IBM, MetaWare, Watcom), and all interpreted it as the latter. The WG recommended adding semantics to resolve the ambiguity in favor of ~(X()). (See Motion 11.) This resolution has the advantage that all explicit destructor calls must be preceded by naming the object

to be destroyed.

Next, Pennello explained that the current WP has no productions to specify syntax for "destructors" for non-class types (issue 466 from N0862 = 96-0044). However, some parts of the WP suggest that the following are valid:

```
p->int::~int()
p->T::~T()
r.A::B::C::~C()
```

Pennello said the WG recommended prohibiting the first and allowing the second two by specifying the syntax in terms of a new non-terminal called pseudo-destructor-name. (See Motion 3.). He also recommended these constraints:

-- everything before the rightmost :: and the type-name following that :: must refer to the same type after lookup.

-- the type-names must be textually identical.

He explained that this syntax supports the needs of templates; templates do not need expressions such as p->int::~int().

Koenig said the constraint that the type-names must be textually identical conflicts with Lajoie's earlier proposal regarding the semantics of p->T1::~T2(). Pennello said it would be easy to amend this proposal to say that the type-names must simply refer to the same types after lookup.

Finally, Pennello presented a proposal to clarify the disambiguation of declarations and statements (issue 424 from N0862 = 96-0044). The WG considered two possible resolutions:

-- Assume it's a declaration and enter names into the symbol table during disambiguation; remove the entries if it turns out not to be a declaration.

-- Disambiguate before parsing.

Pennello said the WG recommended the latter. (See Motion 12.)

No one objected to any of Pennello's proposals.

6.1 Core WG (continued)

==== Adamczyk ====

Adamczyk explained that the current WP requires that, if declared, operator-> must be a unary operator, but operator->* must be a binary operator. He presented a proposal to allow declarations for operator->* as a unary operator as well as a binary operator (N0831 = 96-0013). Under this proposal, unary operator->* would have "turn the crank" semantics similar to operator->. That is, x->*m would be interpreted as (x.operator->*())->*m. (See Motion 18.)

Adamczyk said this proposal is a pure extension; it does not compete with the built-in operator ->* or any existing user-defined binary operator->*. It would allow "smart pointer" classes that use ->* as well as ->.

Adamczyk said the WG backed the proposal, but did not feel strongly about it. They wanted the full committee to decide. Stroustrup asked why this is needed. Dodgson replied that this change allows operator->* to work for iterators and smart pointers.

Gibbons suggested deprecating the binary form of operator->*. Adamcyzk said no one in the WG thought it was used, so no one cared about it. Stroustrup said he'd seen uses, and thought we shouldn't deprecate it.

Ball said he knows of people who use ->* as a binary operator to implement smart pointers. He argued that this extension isn't necessary, and we must not remove binary operator->*.

Adamczyk explained that, with this proposal, if your program has both unary and binary operator->* for a given class type, it's almost impossible to call the binary operator for operands of that type.

Straw Vote:
    Who doesn't like it? 1.
    Who likes the proposal as is? lots.
    Who wants to see a better proposal at a later date? 4.

Adamczyk presented a proposal to change the argument-matching rules for inherited conversion functions (N0835 = 96-0017). Using the example from that paper, he explained that the WP currently favors conversion operators declared in a derived class over conversion operators inherited from a base class when converting from a derived class type. (See Motion 13.)

Adamczyk later explained that the issue is not whether the conversion functions are available in the derived class -- they are. The issue is their weight in overload resolution. Right now, inherited conversion functions are always considered less desirable than those declared in the derived class. This proposal would make them equally desirable. This is the same approach already adopted for using-declarations.

Koenig suggested that inherited conversion functions should be invisible in the derived class. Stroustrup said he originally agreed with Koenig, but now believes the proposal is right.

Straw vote: Who agrees with this proposal? lots yes, 0 no.

Adamczyk presented a proposal to eliminate the class rvalue standard conversion (N0839 = 96-0021). He said the change is editorial -- it doesn't change the interpretation of any currently well-formed program. However, it extends the language to accept some previously ill-formed programs (see "A Related Case" discussed in the paper) that people probably expect to work.

Straw Vote: Who favors this proposal? lots yes, 0 no.

Adamczyk presented a proposal to relax the rules for functions returning void (N0849 = 96-0031). The proposal had two parts. The first allowed a function returning void to have a return statement with a void expression. This is most helpful in writing templates. For example,

```
template<class T>
T call_f(T (*f)())
{
    return f();
}

int f1();
call_f(f1);     // ok
void f2();
call_f(f2);     // currently an error, ok by this proposal
```

Adamczyk said this change would be useful for the mem_fct proposal (N0848 = 96-0030), but it's not essential. It's a pure extension, with no apparent definitional problems. He said the WG did not back this proposal, but if the committee wanted to accept it, the WG had no

objection.

The second part of the proposal allowed passing a void expression to a void parameter. Adamczyk said the WG saw less need for this. They were concerned that it would be difficult to define. Adamczyk called it a "slippery slope" sliding toward full support for void values. The WG opposed this part of the proposal.

Straw Vote: Who favors part 1 of this proposal? 17 yes, 14 no.

Koenig advised against bringing this for a formal vote because the straw vote was so close.

Adamczyk presented a proposal to clarify access control (N0852 = 96-0034). He explained the issue using the example in the paper. He said the change may be just editorial. The WG also recommended extending the clarification to include static members as well as non-static members. (See Motion 15.)

Straw Vote: Who favors this proposal? lots yes, 0 no.

Adamczyk said the WG agreed with the recommendation on issue 586 from N0864 = 96-0046, namely, that access checking for default arguments is done on name lookup. Therefore, access is checked at the declaration of a default argument, not at its implied use at a function call. Adamczyk said the WG considered this merely editorial. No one objected.

Adamczyk explained another access control issue. Presently, members of a nested class have no special access to the class containing the nested class. Adamczyk asked if the committee wanted to grant any such special access (as suggested in N0615 = 95-0015). He pointed out that this proposal isn't really necessary because enclosing classes can grant access to nested classes via friend declarations, as in

```
class A {
    class B;
    friend class B;
    class B { ... };
};
```

Stroustrup remarked that this is not a coherent way to handle extensions, and he advised the committee to drop this proposal. Adamczyk agreed.

Adamczyk said the WG agreed with the recommendation on issue 532 from N0864 = 96-0046, namely, that a friend declaration may not define a class. For example,

```
class A {
    friend class B { ... };     // ill-formed
};
```

Adamczyk said the WG considered this merely editorial. No one objected.

Adamczyk then presented a proposal to eliminate the words from clause 5 that say user-defined conversions apply wherever necessary (N0853 = 96-0035). He said the WP now has statements detailing where specific conversions apply, so it no longer needs this "blanket" rule to cover those cases. There may be other cases where this rule still applies, but they are just "nits" (except for ?: discussed below). The WG recommended that we remove the "blanket" rule and, if needed in the future, add specific statements to restore conversions we might have lost.

Adamczyk added that the ?: operator requires specific words to clarify the allowable conversions on its second and third operand. He said the

WG recommended modeling the behavior of ?: on the == operator. (See Motion 17.) This part of the proposal reverses a previous vote that disallowed conversions to reference type in lvalue cases. For example,

```
struct A {
    operator int &();
} a;
int i;
a = i;
(x ? a : i) = 1;    // propose to allow this
```

Apparently, no one objected to this.

Finally, Adamczyk presented a proposal to disallow conversion sequences that use a standard conversion after a user-defined conversion to bool. He asked if anyone wanted to vote on this at this meeting.

Schwarz explained that the iostreams specification depends on conversions to bool. As the language stands, many of those conversions don't work properly. Myers added that, if we don't accept this proposal, we will have to change the conversions back to void *. Nobody likes this, but it's the only thing that will work.

Ball said he'd rather see a proposal to remove the standard conversion from bool to int. That's the right way to do it. Adamczyk replied that we really need to see a paper on this, rather than rush to a decision at this meeting.

Straw Vote: Who wants something to vote on...
    ... at this meeting? 8
    ... at the next meeting? lots.

==== Gibbons ====

Gibbon's reported that his WG reviewed items from the template issues list (revision 14) (N0841 = 96-0023). They proposed changes to the WP to resolve many of these issues. (See Motion 20.)

Gibbon said the WG recommended accepting Lajoie's rewrite of clause 14 (N0877 = 96-0059). He didn't think it was necessary to vote on it since it's editorial. (It became a formal motion anyway. See Motion 19.)

Gibbons presented a proposal to augment the phases of translation to include template instantiation (N0818 = 95-0218). (See Motion 21.)

No one objected to any of these proposals.

6.3 C Compatibility

Plum presented a proposal to clarify the requirements for floating literals (core issue 506 from N0862 = 96-0044). (See Motion 22.)

Plum presented a proposal to specify the value of the predefined macro __cplusplus. The WG recommended the value 199707L, representing July of 1997, the expected date that the standard will be official. The intent is that the editor will change this value as needed to match the actual date.

Plum presented a proposal to allow extended characters in identifiers and literals (N0886 = 96-0068). The proposal uses universal character names (UCNs) to specify extended characters. A UCN names a character in the ISO 10646 character set and has the form ??uXXXX or ??UXXXXXXXX, where X is a hex digit. For example,

```
wchar_t f??u00FC()
{
```

```
    return L'??u1234';
}
```

uses two UCNs: ??u00FC and ??u1234. The compiler must look up each UCN appearing in an identifier to validate that UCN.. The WG recommended listing the valid UCNs in an annex to the standard; however, Plum said that the editor can list the UCNs wherever he deems appropriate.

Koch asked if the proposal allows a basic source character in a reserved word. That is, will he be able to spell "if" using UCNs? Plum said no.

Hartinger asked if this proposal affects tables used in evaluating character classification functions such as isalpha. Plum replied that this proposal does not affect the locales a compiler must support -- there's no requirement to support anything but the "C" locale.

Adding to Plum's earlier explanation, Nelson said the table in the annex has nothing to do with the execution character set. It's just there to determine if a particular character may to occur in an identifier. In effect, it's there to catch "line noise" in source programs.

Ball asked if WG14+X3J11 is planning to accept much the same thing for C9X. Plum said he believed this proposal is very compatible with what they will accept.

No one objected to the proposal.

6.4 Library WG

==== Dawes ====

Dawes presented a proposal to resolve several issues regarding clause 17 (from N0801R4 = 95-0201R4). The proposal included closing the following clause 17 issues without taking any action:

004 - Divide namespace std:

> The WG decided it would take too much time, and it isn't absolutely necessary

009 - Separate the library from the language:
010 - Too many classes and features in the library:
011 - The library is defined in terms of templates:

> No one in the WG supported any of these.

012 - Decouple libraries:

> Some of the points have already been fixed; no one supported the others.

Dawes pointed out that 009 through 012 were NB (national body) comments, and he wanted it noted that the WG did consider them.

Dawes said the proposal also included accepting the recommendations in N0801R4 = 95-0201R4 for the following issues: 005, 006, 007, 008, 013, 014, and 015.

Ball objected to the proposal regarding issue 006, namely, that extending namespace std should require a diagnostic. Spicer and Stump agreed with Ball. Dawes said the WG would not object if the committee decided that the behavior should be undefined.

Straw Vote: For issue 006, who favors...
... option 1 (requiring a diagnostic): 0.
... option 2 (leaving the behavior undefined): lots.

Dawes agreed to change the proposal regarding issue 006. (See Motion 26.)

Dawes then presented a proposal to add class underflow_error to the library as a standard exception type. The proposed specification was essentially the same as for class overflow_error. (See Motion 27.) He explained that some implementations of complex<> need this class.

Ball asked if the proposal also says when this exception type must be thrown. Dawes said it does not. The type is available to be thrown, but you need not throw it.

Dawes then presented a proposal to resolve numerous issues regarding clause 21. The proposal was to close these following items without taking any action:

014 - The argument order for copy() is incorrect:

The cure was worse than the problem.

059 - string traits are unrelated to iostream_traits:

The i/o subgroup handled this.

080 - Allow template specializations of basic_string and traits:

The clause 17 changes fix this.

081 - Portions of clause 21 are redundant with 23:

The cure was worse than the problem.

083 - eos() may return different values:

Change to traits fixed this.

089 - basic_string needs a release() function:

21-091 (below) fixes this.

The proposal was also to amend the WP as recommended in the following issues: 082, 084, 086, 087, 088, and 091.

Dawes presented a proposal to resolve several issues regarding clause 20 (from N0843R3 = 96-0025R3). Each of these became a separate formal motion:

020 - accept the recommendation from N0843R3 = 96-0025R3. (See Motion 28.)

010 - "clean up" auto_ptr copy semantics by adopting the recommendations in N0830 = 96-0012. Dawes said the WG considered this the least painful fix. (See Motion 29.)

014 - "clean up" allocator as described in N0790R1 = 96-0190R1. (See Motion 30.)

023 - improve the consistency of library member typedefs as described in N0845 = 96-0027. (See Motion 31.)

Dawes also presented Stroustrup’s proposal to allow member functions to be used as STL function objects (N0848 = 96-0030).

No one objected to any of these proposals.

==== Schwarz ====

Schwarz presented various proposals regarding i/o and locales.

Schwarz began by presenting a proposal to eliminate transparent locales as recommended in option 1 of N0860 = 96-0042. (See Motion 34.)

Schwarz explained that there's considerable overlap in string_traits and ios_traits as specified in the current WP. He presented a proposal to consolidate them into char_traits (N0854R1 = 96-0036R1). (See Motion 35.)

Schwarz explained that the original library had bidirectional streams. However, the library in the current WP does not. He presented a proposal to put bidirectional streams (class iostream) back into the library (N0828 = 96-0010). (See Motion 36.)

Bruck noted that we've gone for several years, and through a CD ballot, without class iostream. He wondered if we really need this. Clamage replied that many users are astounded to find that it's not in the standard. Schwarz and others observed that users could write the class themselves with about five lines of code.

Straw Vote: Who favors this proposal? lots yes, 0 no.

Schwarz then presented a proposal to allow registering of callback functions in class ios_base (N0842R1 = 96-0024R1). (See Motion 37.)

Next, Schwarz explained that conversions between char_type and int_type (which allows EOF) must be done explicitly. The current WP is careless about these conversions. He presented a proposal the clean this up by clarifying the not_eof and to_int_type traits (N0885 = 96-0069). In particular:

-- to_int_type preforms an invertible conversion, guaranteed not to produce EOF

-- not_eof takes an int and returns something that isn't EOF

(See Motion 38.)

Finally, Schwarz presented a proposal to resolve various iostream issues (from N0827 = 96-0009). (See Motion 39 for the list of issues.) He also presented a proposal to resolve various locale issues. (See Motion 40 for that list of issues.)

No one objected to any of the above proposals.

==== Dodgson ====

Dodgson presented a proposal to overhaul iterators (N0833 = 96-0015). He summarized the proposed changes:

1. Make things consistently const-correct
2. Allow operator->* for iterators
3. Add a constraint on distance for reachability from first to last
4. Miscellaneous changes to predefined iterators to fix "thinkos"
5. Other repairs to stream iterators
6. Revise the iterator categories

Austern, Plauger, and Myers expressed concern that the proposed inheritance hierarchy for iterator categories (part 6, above) was not "safe". Plauger said it was probably ok, but he needed to think about it.

Dodgson agreed to make part 6 a separate motion. (Dodgson later decided to also separate part 2 from the rest of the motion. See Motion 42 for

part 2, Motion 43 for part 6, and Motion 40 for the other parts.)

Dodgson explained that the definition for the count() algorithm in the current WP is different from the definition in HP's original STL. He presented a proposal to restore the definition to the way it was (N0847 = 96-0029). (See Motion 44.) He explained the problem and the solution using some examples from the paper.

Straw Vote: Who favors this proposal? lots yes, 0 no.

==== Becker ====

Becker summarized the WG's proposals regarding open issues on algorithms (from N0793R1 = 95-0193R1):

004 - Close this issue with no action.

012 - Remove the ambiguity in search() templates as recommended in the paper. (See Motion 45.)

013 - Correct the Requires clause in the description of search() templates as recommended in the paper. (See Motion 46.)

014 - Leave this open.

Becker summarized the WG's proposals regarding numerics issues (from N0844 = 96-0026):

007 - Close this issue with no action.

008 - Close this issue with no action.

009 - Leave this open.

010 - Accept the paper's recommendation to change the name of template complex to basic_complex. (See Motion 47.)

Schwarz objected to the name change (issue 010) because it would take away users' ability to write

```
complex<float> z;   // allegedly natural
```

They would have to write either

```
basic_complex<float> z;
```

or

```
complex_float z;
```

Straw Vote: Who agrees with changing the template name from complex to basic_complex? 19 yes, 4 no.

Becker continued with the proposals regarding numerics issues (from N0844 = 96-0026):

011 - Accept the paper's recommendations to fix ambiguities for basic_complex and its specializations. (See Motion 48.)

012 - (same see 011)

013 - Accept the paper's recommendations to specify which value to return for multi-valued transcendental functions of complex variables. (See Motion 49.)

014 - Remove the keyword "friend" from operators declared as friends in

class basic_complex, and move the operators to the global scope. (See Motion 50.)

No one objected to these proposals.

Becker presented a proposal regarding a new numerics issue:

015 - (1) Change the name norm() to abs_sq(), and (2) add a Returns clause for abs().

Bruck objected to the first part of this proposal. He said that, if Fortran calls this function norm(), then C++ should call it norm() also. Others agreed. Becker removed that part of the proposal. (See Motion 51.)

Becker presented numerous changes suggested by public review comments on the numeric_limits class. (See Motion 54.)

No one objected to these proposed changes.

Becker then presented three proposals regarding valarrays (N0857 = 96-0039):

1. Assignment to a valarray should not resize it. (See Motion 55.)

2. Perverse self-referential permutations should have undefined behavior. (See Motion 57.)

3. Allow assignment of a scalar to a valarray. (See Motion 56.)

Becker elaborated these proposals using examples from the paper. A brief discussion about item 2 ensued.

Straw Vote:
Who favors item 2 as written? 13.
Who would like to see more work to improve the wording? 2.

No one objected to the other proposals.

6.5 Remarks from the WG21 Convener

Harbison explained what it meant to "vote out" the committee draft (CD) at this point. He said we have to make a judgment call. If we go to our second CD ballot (CD2), but make too many changes by the time we go to DIS ballot, SC22 may insist on yet another CD ballot (CD3). If that happens, the project schedule will slip about one year. On the other hand, if we just wait until the next meeting, the schedule slips by four months, but we increase the chance we will not need CD3.

The committees recessed at 6:05 and reconvened at 08:45 on Thursday.

6.1 Core WG

Plum mentioned that the C Compatibility WG will also introduce a motion to eliminate the "six-character header name" limit. (See Motion 25.) No one objected.

==== Gibbons ====

Gibbons introduced a proposal to remove the "separation" model for template compilation from the WP (N0875 = 96-0057). (See Motion 58.)

He explained that his WG attempted to enumerate the pros and cons of this issue. He summarized their conclusions:
-- Separation provides superior name hiding, especially for macros.
-- If separation were removed, it will become a common extension, and

that will hinder the portability of C++ programs.
-- Separation makes source file organization easier.
-- Inclusion is probably more efficient at building programs, i.e., it probably compiles and links faster.
-- Inclusion places fewer constraints on future language changes and implementations, especially integrated development environments (IDEs).
-- Inclusion is easier to specify correctly in the WP.
-- Inclusion is easier to implement.
-- Inclusion may yield better diagnostics.

Ball spoke in favor of the proposal. He said he's the chief implementor of Sun's compiler, and Sun attempted to implement the separation model several years ago. After running code through it, they found major problems with the template specification in the WP. Ultimately, they produced an implementation that lets users organize sources along the lines of the separation model, but binds names as in the inclusion model. Ball said Sun is not worried about the extra work involved in implementing the separation model. They already have almost all the pieces in place.

Ball explained that Sun has lots of customers using templates. Not one has complained about name leakage being a problem. Rather, customers are concerned about compilation speed. Users can use a separation-like model by organizing sources so that the compiler can find template declarations, but not definitions. However, doing so slows build times quadratically. Thus, people rarely use it.

Ball argued that most people who write templates want their code to be reused. Therefore, they tend to write so there's a minimum dependence on names from surrounding context. Thus, name leakage is not a practical problem.

Ball said he wants to remove separate compilation for templates because he's been through the implementation before and there were problems in the specification. Some have been fixed, but others have not. Why take a risk for something that offers little advantage?

Plum asked Ball for a simple statement of the relative costs of building programs with and without their separation-like model. Ball replied that, if a build without "separation" takes 1-2 minutes, then a rebuild with "separation" takes 20-30 minutes.

Bruck then spoke in favor of keeping the separation model. He said he spoke from a user's, rather than an implementor's, perspective. (Some of his remarks also appeared in N0882 = 96-0064).

Bruck began by summarizing what he believed would be the consequences of removing the separation model:
-- The file interference (name leakage) problems will get much worse.
-- Users will run into problems with the one-definition rule (ODR).

Bruck explained that header files can interfere with each other when used in the same compilation. In particular macro names and overloaded function names in one header can conflict with those in another. He added that removing the separation model affects every template library implementation. Without separation, every identifier in a library becomes part of its interface, and therefore needs to be documented.

Bruck explained the difficulties programmers already have in avoiding name conflicts:
-- Library writers must use "funny" names to reduce the risk of conflicts.
-- Users must ensure that no conflicts arise between libraries and with system header files.
He also explained that non-macro names can be encapsulated in name-

spaces. However, anonymous namespaces and "using namespace" do not "work"; explicit qualification and local using declarations are "ok".

Bruck again recommended voting against the proposal to ban separate compilation of templates.

Stroustrup said that separate template compilation appears to be a lightning rod for all template problems. If there's a problem with name binding, or something else, some people respond by saying that if we get rid of separate compilation, the problem will go away.

Bruck reiterated that, although you can hide non-macro names in name-spaces, the only way to hide macro names is by separate compilation.

Scian asked if Bruck would accept a separation model that binds names as in the inclusion model. Bruck said he wouldn't because it doesn't solve the name leakage problem.

Stroustrup said this proposal is unique in the history of this committee in that it would ban a major language feature. This feature affects the way we organize programs and designs. The proposal would overturn a compromise struck in Valley Forge (November, 1994). That compromise allows two models, and so that users have a choice of techniques. The compromise had overwhelming support at the time. Stroustrup said he didn't think we had any significant new technical information on the issue since the compromise. He said anyone who voted for the compromise then should vote now to keep separation, to preserve trust and a good working environment within the committee.

Gibbons spoke in favor of the proposal. He said his experience with advanced IDEs is that files don't mean much in those environments. He argued that hiding names by separate compilation is an antiquated style:
-- Separation is a solution in search of a problem.
-- By the time it is practical, it will be obsolete.

Ball remarked that debuggers pose similar problems; they ignore file boundaries just like IDEs do.

Stroustrup countered that the technique of organizing programs into files will last well beyond the next version of the standard.

Responding to Stroustrup's earlier remarks, Plum said he doesn't believe we had a policy against overturning decisions. He said the committees have already reversed a compromise on transparent locales. Unruh said the committees have also changed prior decisions on name injection and implicit int.

Adamczyk said he's been on other standards committees, and there is ample precedent for reversing previous decisions.

Stroustrup said the reasons for banning separation keep changing. He also said he sees separation as more than just a feature, but part of the fabric of the language.

Spicer presented an example to show that, if you have an implementation that supports both inclusion and separation (as does the current WP), and you write using inclusion model, compilers and linkers must still do all the work as if you were using separation model. Compilers can't tell which model a program uses by just examining the source code. They need to do extra work or use extra-lingual mechanisms.

Koenig said he agreed with Stroustrup that he will accept any compromise that allows people to write templates in separately-compiled files. He also agreed with Colvin in believing that some technical solution is possible.

Straw Vote: Who favors removing the separation model for templates?
X3J16+WG21: 25 yes, 12 no, 6 abstain.
WG21 only: 3 yes, 2 no, 1 abstain.

6.4 Library WG

==== Dawes ====

Dawes asked in there were any objections to a proposal to modify the library to allow

new (nothrow)

instead of

new (nothrow())

There were none. (See Motion 59.)

The committees recessed to WGs at 11:15 and reconvened at 15:40 that same day.

9 General session II

9.1 Core Language WG

Lajoie reported that her Core WG still had 15 open issues regarding memory model, object model and linkage.

Adamczyk said that Unruh had raised strong objections to the proposal to allow unary operator->*. The Core WG would not submit that motion after all. Adamczyk said they didn't want to do any more work on it. He said that if anyone had concerns about this, they should see him.

Gafter reported that Adamczyk's Core WG still had about 22 open issues.

Gibbons reported that his Core WG still had some open issues regarding exceptions and name lookup. He said he was not sure that all decisions regarding templates had been incorporated into WP; they needed to review the situation. He also said there were about 10 open issues regarding templates, and he expected there would be 10 more by the next meeting.

9.2 Library WG

Schwarz reported that there were still 74 open issues regarding i/o. Three were significant enough to require a paper:

1. The following

```
basic_ostream<jchar> out;
out << 'a';
```

prints 'a' as a decimal integer.

2. imbuing codecvt in filebuf

3. There's a problem with

```
unsigned char c;
cout << c;
```

because cout is a stream of char.

Schwarz appealed for help with iostreams issues. He suggested meeting on the Sunday before the Stockholm meeting to address these issues. Stroustrup suggested a meeting sometime before then. Harbison seconded that idea. Schwarz asked how many would attend an iostream meeting in

San Francisco or San Jose area about six weeks after this meeting. Four or five said they would.

Dawes reported that there were no more than 10 open issues in clauses 17 through 21 and clause 23. The library introduction (clause 17) had one serious open issue -- the header inclusion policy.

Becker said there was one small open issue on algorithms, 14 open issues on numerics, and 12 open issues on iterators, mostly small.

Dodgson said his group would like to see the proposal for a unary operator->* ratified.

9.3 ANSI C compatibility WG

Plum said there are no outstanding C compatibility issues.

9.4 Edit WG and other general session business

Straw Vote: Who thinks we're ready to submit the WP for another CD ballot? 0 yes, lots no.

Harbison agreed that delaying the CD ballot was wise, but he cautioned against slipping any more.

He said he sensed the minority in template compilation model had strong feelings of dissatisfaction. He didn't think the issue was settled. We had not reached consensus. He said we need to work something out before the Stockholm meeting.

Plum explained what US needs to do to insure that all public comments have been replied to. He asked for volunteers to participate in a review committee for public responses. Clamage and Plum volunteered.

Plum announced that, since we would not be voting on CD submission, there's no need for the scheduled US TAG meeting.

Harbison thanked Lajoie for handling all the public comments. Applause.

10 WG sessions (if any time is left)

The committee recessed at 16:40 and reconvened at 8:50 on Friday.

11 Review of the meeting

Clamage explained that we will conduct the X3J16 votes first. The results of each vote will determine how the US IR (International Representative), namely Plum, will cast his vote in WG21.

Clamage counted 41 X3J16 voting members present.

11.1 Formal motions

1) Motion (to accept the WP) by Koenig/Dawes:

Move we accept N0836 = 96-0018 as the current WP.

Motion passed X3J16: 42 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

==== presented by Lajoie ====

2) Motion (to clarify name look up after . and -> operator) by Lajoie/Hartinger:

Move we amend the WP as follows:

-- replace the second and third sentences of 5.2.4 (_expr.ref_) paragraph 2 with:

For the second option (arrow), the type of the first expression (pointer expression) shall be "pointer to class object" (of complete type) in which case the id-expression shall name (_basic.lookup.classref_) a member of the class.

-- move 5.2.4 [expr.ref] paragraphs 3 and 4 to 3.4.4 [basic.lookup.classref] with the following changes:

-- replace the first 3 sentences of paragraph 3 with:

If the id-expression in a class member access (_expr.ref_) is an unqualified-id, and the type of the object-expression is of a class type C (or of pointer to a class type C), the unqualified-id is looked up in the scope of the class C. If the type of the object-expression is of pointer to scalar type, the unqualified-id is looked up in the scope of the object-expression.

If the id-expression in a class member access (_expr.ref_) is a qualified-id of the form

class-name-or-namespace-name::...

the class-name-or-namespace-name following the . or -> operator is looked up both in the context of the entire postfix-expression and in the scope of the class of the object-expression. If the name is found only in the scope of the class of the object-expression, the name shall refer to a class-name. If the name is found only in the context of the entire postfix-expression, the name shall refer to a class-name or namespace-name. If the name is found in both contexts, the class-name-or-namespace-name shall refer to the same entity. If the qualified-id has the form

::class-name-or-namespace-name::...

the class-name-or-namespace-name is looked up in global scope as a namespace-name or a class-name.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

3) Motion (to clarify name look up for a scalar type destructor name) by Lajoie/Hartinger:

Move we amend the WP as follows:

-- add the following as a new paragraph after 3.4.2 [basic.lookup.qual] paragraph 4:

A nested-name-specifier that names a scalar type, followed by ::, followed by ~type-name is a pseudo-destructor-name for a scalar type (_class.dtor_). The type-name is looked up as a type in the scope of the nested-name-specifier. [Example:

```
struct A {
    typedef int I;
};
typedef int I1, I2;
extern int* p;
extern int* q;
p->A::~I();     // ok: I is looked up in the scope of A
```

```
        q->I1::~I2();   // ok: I2 is looked up in the scope of the
                        // postfix-expression

        -- end example]
```

-- add the following productions to 5.2 [expr.post] and A.4 [gram.expr]:

```
        postfix-expression:
            postfix-expression . pseudo-destructor-name
            postfix-expression -> pseudo-destructor-name

        pseudo-destructor-name:
            ::-opt nested-name-specifier-opt type-name :: ~ type-name
            ::-opt nested-name-specifier-opt ~ type-name
```

-- add the following as a new subclause [expr.pseudo] before 5.2.4 [expr.ref] creating paragraph 8:

The use of pseudo-destructor-name after a dot . or arrow -> represents the "destructor" for the non-class type named by type-name. The result can be used only as the operand for the function call operator (), and the result of such a call has type void.

The only effect is the evaluation of the postfix-expression before the dot or arrow. The left hand side of the dot operator shall be a scalar type; the left hand side of the arrow operator shall be a pointer to scalar type. This scalar type is the object type. The type designated by the pseudo-destructor-name shall be the same as the object type. Furthermore, the two type-names in a pseudo-destructor-name of the form

```
::-opt nested-name-specifier-opt type-name :: ~ type-name
```

shall designate the same scalar type.

-- add at the end of 5.2.4 [expr.ref] paragraph 4:

If the id-expression is ~type-name, and the type of the object-expression is of a class type C (or of pointer to a class type C), the type-name is looked up in the context of the entire postfix-expression and in the scope of class C. If the type-name is found in one of these two contexts, it shall be a class-name. If type-name is found in both contexts, the name shall refer to the same entity. If the type of the object-expression is of scalar type, the type-name is looked up in the scope of the object-expression.

-- change the last example in 12.4 [class.dtor] to be:

```
        typedef int I;
        I *p;
        // ...
        p->I::~I();
```

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

4) Motion (to clarify name look up of anonymous union members) by Lajoie/Wilcox:

Move we amend the WP as follows:

-- insert after the first sentence of 9.5 [class.union] paragraph 2:

The member-specification in an anonymous union shall only define data members.

-- replace the text after the ; in the second sentence of 9.5 [class.union] paragraph 2 with:

for the purpose of name lookup, the members of the anonymous union are considered to have been defined in the enclosing scope.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

5) Motion (to clarify class member look up) by Lajoie/Hartinger:

Move we amend the WP as follows:

-- add the following as a new paragraph after 3.3.1 [basic.scope.pdecl] paragraph 3:

After the point of declaration of the class member, the member name can be looked up in the scope of its class. [Note: this is true even if the class is an incomplete type. For example:

```
struct X {
    enum E { z = 16 };
    int b[X::z]; // ok
};
```

-- end note]

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

6) Motion (to allow arrays of incomplete class type) by Lajoie/Stump:

Move we amend the WP as follows

-- delete "an incomplete type, " from the second sentence of 8.3.4 [dcl.array] paragraph 1.

-- replace the first sentence of 3.9 [basic.types] paragraph 6 with:

A class that has been declared but not defined or an array of unknown size or of incomplete element type is an incomplete type.

-- replace the following text at the beginning of the second sentence of 3.9 [basic.types] paragraph 7:

The declared type of an array might be incomplete...

with:

The declared type of an array object might be an array of unknown size and therefore incomplete...

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

7) Motion (to disallow const class { };) by Lajoie/Hartinger:

Move we amend the WP by adding the following as a new paragraph after 7.1.5.1 [dcl.type.cv] paragraph 3:

If a cv-qualifier appears in a decl-specified-seq, the
init-declarator-list of the declaration shall not be empty.

Motion passed X3J16: lots yes, 1 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

8) Motion (to impose requirements on the representation of bitfields)
by Lajoie/Hartinger:

Move we amend the WP by adding the following as a new paragraph
after 9.6 [class.bit] paragraph 3:

If the value true or false is stored into a bit-field of type bool
of any size (including a one bit bit-field), the original bool value
and the value of the bit-field shall compare equal. If the value of
an enumerator is stored into a bit-field of the same enumeration
type and the number of bits in the bit-field is large enough to hold
all the values of that enumeration type, the original enumerator
value and the value of the bit-field shall compare equal.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 5 yes, 0 no, 1 abstain.

9) Motion (to relax the requirements on static members of POD classes)
by Lajoie/Hartinger:

Move we amend the WP by inserting "non-static data" into the
following sentences of 9 [class] paragraph 4:

A POD-struct is an aggregate class that has no non-static data
----
members of ...

A POD-union is an aggregate union that has no non-static data
----
members of ...

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

==== presented by Pennello ====

10) Motion (to allow ++(int&)x) by Pennello/Stump:

Move we amend the WP as follows:

Change the grammar rules in 5.3 [expr.unary] and A.4 [gram.expr] for
++/-- to:

unary-expression:
    ++ cast-expression
    -- cast-expression

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

11) Motion (to resolve ~X() ambiguity) by Scian/Charney:

Move we amend the WP as follows:

Append to 5.3.1 [expr.unary.op] paragraph 9:

There is an ambiguity in the unary-expression ~X(), where X is a
class-name. The ambiguity is resolved in favor of treating ~ as
unary complement rather than treating ~X as a reference to a

destructor.

Motion passed X3J16: lots yes, 1 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

12) Motion (to clarify statement disambiguation) by Pennello/Stump:

Move we amend the WP as follows:

Change 6.8 [stmt.ambig] paragraph 3 to:

The disambiguation is purely syntactic; that is, the meaning of all names occurring in such a statement, beyond whether they are type-ids or not, is not used in or changed by the disambiguation. Disambiguation precedes parsing, and a statement disambiguated as a declaration may be an ill-formed declaration.

[Example:

```
struct T1 {
    T1 operator()(int x) { return T1(x); };
    int operator=(int x) { return x; };
    T1(int) {};
};
struct T2 { T2(int) {}; };
int a, (*(*b)(T2))(int), c, d;
void f() {
    // Disambiguation requires this to
    // be parsed as a declaration
    T1(a) = 3,
    T2(4),                  // T2 will be declared as
    (*(*b)(T2(c)))(int(d)); // a variable of type T1
                            // but this will not allow
                            // the last part of the
                            // declaration to parse
                            // properly since it depends
                            // on T2 being a type-name
}
```

end example]

Koch opposed this motion because it might invalidate some fairly simple declarations in currently valid programs. Pennello said this allows application of Early's algorithm in context-free grammars (which has essentially linear behavior). Without this proposal, the parser must operator in a context-sensitive setting, which is hard to do.

Motion to table by Koenig/Koch:
Motion failed X3J16: 17 yes, 17 no.

Motion passed X3J16: lots yes, 4 no.
Motion passed WG21: 4 yes, 0 no, 2 abstain.

==== presented by Adamczyk ====

13) Motion (to change the handling of inherited conversion functions in overload resolution) by Adamczyk/Lajoie:

Move we amend the WP as indicated in N0835 = 96-0017.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

14) Motion (to eliminate the class rvalue standard conversion) by Adamczyk/Lajoie:

Move we amend the WP as described in N0839 = 96-0021, excluding the final edit in that document. (That is, the deletions in 13.3.3.2 [over.ics.rank] are not to be done).

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

15) Motion (to clarify access control) by Adamczyk/Lajoie:

Move we amend the WP as indicated in N0852 = 96-0034, with the phrase "For non-static members," and the word "non-static" removed from the changes (the text applies to all members and not just non-static members).

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

16) Motion (to eliminate the rule that implicit conversions can be done wherever necessary in expressions) by Adamczyk/Lajoie:

Move we amend the WP as indicated in N0853 = 96-0035. Specifically, delete clause 5 [expr] paragraph 9.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

17) Motion (to rework the description of the "?" operator) by Adamczyk/ Lajoie:

Move we amend the WP as indicated in N0890 = 96-0072.

Gibbons asked members to approve this, even though it needs more work.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

18) Motion (to add a single-parameter version of operator->*) by Adamczyk/Dodgson:

Move we amend the WP as follows:

-- Add to Table 8 in 13.3.1.2 [over.match.oper] after the item for 13.5.6:

13.5.7 a->* (a).operator->*()
a->*b (a).operator->*(b) operator->*(a, b)

and renumber the item for 13.5.7 as 13.5.8.

-- Add as a new paragraph after paragraph 10 of 13.3.1.2 [over.match.oper]:

As described in _over.arrowstar_, when an operator->* function taking no parameters is found, ->* is interpreted by ignoring the second operand and calling the operator->* function on the first operand. When that operator->* returns, the operator ->* is applied to the value returned, with the original second operand.

-- Add a new section after 13.5.6 [over.ref], calling this new section [over.arrowstar]:

operator->* shall be a function as described under _over.binary_, or a non-static member function taking no parameters. It implements pointer to member access using ->*

pm-expression ->* cast-expression

An expression x->*m is interpreted as (x.operator->*())->*m for a class object x of type T if a T::operator->* taking no parameters exists. Otherwise, the operator is interpreted as a normal binary operator according to _over.match.oper_, with the built-in operator and any overloaded operator->* functions (other than operator->* member functions taking no parameters) taken as the set of candidate functions.

Unruh opposed this. He said it had not been analyzed thoroughly. Corfield called this a "gratuitous extension". Stroustrup said we have not considered all the implications of this extension. Gibbons said this is not an extension, it's a correction. Colvin added that, without this change, users can't create a smart pointer-to-member template without extraordinary effort; with this addition, it's much easier.

Adamczyk explained that his Core WG was not in a position to evaluate the merits of this as an extension. They simply did this work at the request of Dodgson's Library subgroup.

Several others agreed with Stroustrup that more analysis was needed. Colvin said users are puzzled by the absence of smart pointers-to-members.

Motion failed X3J16: 16 yes, 23 no.
Motion failed WG21: 1 yes, 3 no, 2 abstain.

Straw Vote: Who wants someone to continue exploring this issue? 16 yes, 10 no.

Adamczyk said his Core WG has done its work; someone else would have to continue the work.

==== presented by Gibbons ====

19) Motion (to clean up the template clause) by Lajoie/Spicer:

Move we amend the WP by replacing 14 [temp] with the replacement text from N0877 = 96-0059.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

20) Motion (to resolve various template issues) by Spicer/Corfield:

Move we amend the WP as specified in N0892 = 96-0074. (N0892 = 96-0074 describes changes relative to N0836 = 96-0018. If Motion 19 passes, the intent is to apply the equivalent changes to the new clause 14 text.)

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

21) Motion (to clarify the interaction of template instantiation and the phases of translation) by Corfield/Bruck:

Move we amend the WP as specified in N0818R1 = 95-0218R1.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

==== presented by Plum ====

22) Motion (to clarify the requirements for floating literals) by Plum/Benito:

Move we amend the WP by adding the following sentence to 2.10.3 [lex.fcon], paragraph 1:

If the scaled value is not in the range of representable values for its type, the program is ill-formed.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

23) Motion (to specify the value of __cplusplus) by Plum/Lajoie:

Move we amend the WP by modifying the last sentence of 16.8 [cpp.predefined], paragraph 1, to read as follows:

__cplusplus The name __cplusplus is defined to the value 199707L when compiling a C++ translation unit.

Footnote: It is intended that future versions of this standard will replace the value of this macro with a greater value. Non-conforming compilers should use a value with at most five decimal digits.

[Editorial Box: The date is intended to specify the expected official date of the standard. The Project Editor will revise the date as needed.]

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

24) Motion (to incorporate extended identifiers and extended literals) by Plum/Bruck:

Move we amend the WP by adding the changes specified in N0885B = 96-0068.

Kiefer said the German delegation discussed this, they favored it, but they thought it needed more work.

Motion passed X3J16: lots yes, 1 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

25) Motion (to eliminate the six-character restriction on header names) by Plum/Benito:

Move we amend the WP, 16.2 [cpp.include], paragraph 5, by striking the words "and restrict the mapping to six significant characters before the period", and eliminating the Editorial Box which asks "Does this restriction still make sense for C++?"

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

==== presented by Dawes ====

26) Motion (to resolve several issues from the Clause 17 [Library Introduction] Issues List) by Dawes/Rumsby:

Move we amend the WP as described in N0801R5 = 95-0201R5 by adopting the proposed resolutions for issues 005, 006, 007 Option 2, 008, 013 Change 1 and 4 only, 014, and 015. In addition, close without taking any action, issues 004, 009, 010, 011 and 012.

Dawes reminded everyone that 007 Option 2 is that extending namespace std yields undefined behavior.

Motion passed X3J16: lots yes, 0 no.

Motion passed WG21: 6 yes, 0 no, 0 abstain.

27) Motion (to add class underflow_error to the library) by Dawes/Rumsby:

Move we amend the WP by:

-- adding the following as clause 19.1.9:

19.1.9 Class underflow_error [lib.underflow.error]

```
namespace std {
    class underflow_error : public runtime_error {
    public:
        underflow_error(const string& what_arg);
    };
}
```

The class underflow_error defines the type of objects thrown as exceptions to report an arithmetic underflow error.

underflow_error(const string& what_arg);

Effects: Constructs an object of class underflow_error.

Postcondition: what() == what_arg.data().

-- adding a declaration for

class underflow_error;

to the synopsis in 19.1 [lib.std.exceptions] paragraph 3.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

28) Motion (to resolve several issues from the Clause 20 [Utilities] Issues List) by Dawes/Rumsby:

Move we amend the WP as described in N0843 = 96-0025 Revision 3 issue 020.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

29) Motion (to cleanup auto_ptr copy semantics) by Dawes/Colvin:

Move we amend the WP as described in N0830R1 = 96-0012R1.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

30) Motion (to cleanup allocator) by Dawes/Rumsby:

Move we amend the WP as described in N0790R1 = 95-0190R1, after replacing the text "x.rebind<U>::other" in the proposal with "X::rebind<U>::other". (Clarification for the editor: In the table of replacement text forms in the proposal, the patterns are meant to be substituted syntactically rather than textually.) Also, in 20.1.4 [lib.allocator.requirements] Table 41, replace the description of variable u with:

a value of type X::rebind<U>::other::const_pointer for some type U, obtained by calling member a1.allocate for some a1 where a1 == a.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

31) Motion (to make library member typedefs consistent) by Dawes/Rumsby:

    Move we amend the WP as described in N0845 = 96-0027.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

32) Motion (to allow member functions to be used as STL function objects) by Dawes/Rumsby:

    Move we amend the WP as described in N0848 = 96-0030, except change the spelling of the following names:

From:	To:
Mem_fct	mem_fun_t
Mem_fct1	mem_fun1_t
mem_fct	mem_fun
mem_fct1	mem_fun1
Mem_fct_ref	mem_fun_ref_t
Mem_fct1_ref	mem_fun1_ref_t
mem_fct_ref	mem_fun_ref
mem_fct1_ref	mem_fun1_ref

    Also, add the declarations for the adapters in the N0848 = 96-0030 to the synopsis in 20.3 [lib.function.objects].

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

33) Motion (to resolve several issues from Clause 21 [Strings] Issues List) by Dawes/Rumsby:

    Move we amend the WP as described in N0876 = 96-0058 Revision 14 by adopting the proposed resolutions for issues 082, 084, 086, 087, 088, 091. In addition, close without taking any action, issues 014, 059, 080, 081, 083 and 089.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

==== presented by Schwarz ====

34) Motion (to eliminate transparent locales and close various locale issues) by Schwarz/Smithey:

    Move we amend the WP as described in option 1 of the proposed resolutions of N0860 = 96-0042, thus closing issues 22-034, 22-037, 22-039, and 27-311.

Motion passed X3J16: lots yes, 6 no.
Motion passed WG21: 5 yes, 1 no, 0 abstain.

35) Motion (to consolidate ios_traits and string_traits into char_traits) by Schwarz/Smithey:

    Move we amend the WP as described in section 4 of N0854R1 = 96-0036R1.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

36) Motion (to reincorporate bidirectional streams into the library) by

Schwarz/Smithey:

Move we amend the WP as proposed in N0828 = 96-0010, thus closing issue 27-905.

Motion passed X3J16: lots yes, 5 no.
Motion passed WG21: 5 yes, 0 no, 1 abstain.

37) Motion (to add ios_base::register_callback and related functions) by Schwarz/Smithey:

Move we amend the WP as proposed in N0842R1 = 96-0024R1, thus closing iostream issues 27-105 and 27-303.

Motion passed X3J16: lots yes, 1 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

38) Motion (to clarify use of not_eof and to_int_type) by Schwarz/Smithey:

Move we amend the WP as proposed in N0885 = 96-0069, thus closing iostream issues 27-105 and 27-303.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

39) Motion (to close various iostream issues) by Schwarz/Smithey:

Move we:

-- amend the WP (to incorporate basic_istream::sentry and basic_ostream::sentry) as proposed in issue 27-903 of N0827 = 96-0009, but modify the description of basic_ostream::sentry::~sentry to be:

   Effects: For the constructor argument value os, if ((os.flags() & os.unitbuf) && !uncaught_exception()) is true, then calls os.flush().

-- amend the WP as described in N0827 = 96-0009 issues 27-104, 27-306, 27-307, 27-308, 27-903, and delete a sentence as proposed in issue 27-101.

-- draw the editor’s attention to the following issues raised in N0827 = 96-0009: 27-102, 27-309, 27-310, 27-402, 27-403, 27-504 and 27-914.

-- close the following issues in N0827 = 96-0009 without change to the WP: 27-103, 27-305, 27-309, 27-310, 27-402, 27-504 and 27-914.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

40) Motion (to close various locale issues) by Schwarz/Smithey:

Move we:

-- amend the WP (to specify actions of locale constructors on null arguments) as proposed in N0887 = 96-0071, thus closing locale issue 22-058.

-- amend the WP (to eliminate a humungously large array from the library) as proposed in N0886 = 96-0070, thus closing locale issue 22-062.

-- draw the editor's attention to the following issues raised in N0891 = 96-0073: 22-016, 22-030, 22-017.

-- close, without change to the WP, the following issues from N0891 = 96-0073: 22-038, 22-050, 22-054, 22-056, 22-060, 22-067, 22-068 and 22-069.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

==== presented by Dodgson ====

41) Motion (to accept iterator issues) by Dodgson/Dawes:

Move we amend the WP as described in N0833R1 = 96-0015R1 excluding Section 2 and Table 6B.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

42) Motion (to accept iterator operator->*):

Move we amend the WP as described in Section 2 of N0833R1 = 96-0015R1.

No one moved this motion.

43) Motion (to accept derivation for forward_iterator_tag) by Dodgson/Dawes:

Move we amend the WP as described in Table 6B of N0833R1 = 96-0015R1, except that forward_iterator_tag is derived only from input_iterator_tag.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

44) Motion (to accept iterator_traits) by Dodgson/Swan:

Move we amend the WP as described in N0847 = 96-0029, except replace all occurrences of 'iterator_trait' with 'iterator_traits'.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

==== presented by Becker ====

45) Motion (to remove ambiguity in search() templates) by Becker/Dodgson:

Move that we amend the WP by changing the third and fourth template function names of clause 25.1.9 [lib.alg.search] from search to search_n, and make the corresponding change in the synopsis for clause 25 [lib.algorithms].

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

46) Motion (to correct the Requires clauses in the descriptions of search() templates) by Becker/Swan:

Move that we amend the WP in accordance with issue #013 in the Clause 25 (Algorithms Library) Issues document, N0793 = 95-0193.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

47) Motion (to close one clause 26 issue) by Becker/Saini:

Close issue 26-010 from N0844 = 96-0026 with no editorial action.

Motion passed X3J16: lots yes, 2 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

48) Motion (to eliminate ambiguities when assigning a scalar to a complex) by Becker/Schwarz:

Move that we amend the WP by adding the following member functions to the template complex in clause 26.2.1 [lib.complex]:

```
complex<T>& operator = ( T );
complex<T>& operator += ( T );
complex<T>& operator -= ( T );
complex<T>& operator *= ( T );
complex<T>& operator /= ( T );
```

and adding the following member functions to the template specialization complex<float> in clause 26.2.2 [lib.complex.special]:

```
complex<float>& operator = ( float );
complex<float>& operator += ( float );
complex<float>& operator -= ( float );
complex<float>& operator *= ( float );
complex<float>& operator /= ( float );
```

and adding the following member functions to the template specialization complex<double> in clause 26.2.2 [lib.complex.special]:

```
complex<double>& operator = ( double );
complex<double>& operator += ( double );
complex<double>& operator -= ( double );
complex<double>& operator *= ( double );
complex<double>& operator /= ( double );
```

and adding the following member functions to the template specialization complex<long double> in clause 26.2.2 [lib.complex.special]:

```
complex<long double>& operator = ( long double );
complex<long double>& operator += ( long double );
complex<long double>& operator -= ( long double );
complex<long double>& operator *= ( long double );
complex<long double>& operator /= ( long double );
```

and adding the following text to clause 26.2.4 [lib.complex.member.ops]:

```
complex<T> operator += ( T rhs );
```

Effects: Adds the scalar value rhs to the real part of the complex value *this and stores the result in the real part of *this, leaving the imaginary part unchanged.

Returns: *this.

```
complex<T> operator -= ( T rhs );
```

Effects: Subtracts the scalar value rhs from the real part of the complex value *this and stores the result in the real part of *this, leaving the imaginary part unchanged.

Returns: *this.

complex<T> operator *= (T rhs);

Effects: Multiplies the complex value *this by the scalar value rhs and stores the result in *this.

Returns: *this.

complex<T> operator /= (T rhs);

Effects: Divides the complex value *this by the scalar value rhs and stores the result in *this.

Returns: *this.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

49) Motion (to specify which value to return for multi-valued transcendental functions of complex variables) by Becker/Dodgson:

Move we amend the WP by adding the following text to the end of the final paragraph of clause 26.2.7 [lib.complex.transcendentals]:

The sqrt function returns the complex value whose phase angle is greater than -pi/2 and less than or equal to pi/2. All other functions which can produce multiple values return a complex value whose imaginary part is greater than -pi and less than or equal to pi.

Becker explained that the proposed wording isn't quite right, but it was better than nothing, which is what's in the WP. He recommended passing the motion. The WG would fix the problem by the next meeting.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

50) Motion (to eliminate unnecessary 'friend' declarations from complex) by Becker/Dodgson:

Move we amend the WP by removing the word "friend" from the operators declared as friends of the template complex in clause 26.2.1 [lib.complex] and moving the declarations of all of those operators so that they come after the closing curly brace in the definition of complex.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

51) Motion (to provide a definition of the abs() function for complex types) by Becker/Swan:

Move we amend the WP by adding the following text to clause 26.2.6 [lib.complex.value.ops]:

template <class T> T abs(const complex<T>& x);

Returns: the magnitude of x.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

52) Motion (to close one open issue in the Algorithms issues list) by Becker/Dodgson:

Move we close issue #4 in N0793 = 95-0193 with no change to the WP.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

53) Motion (to close two open issues in the Numerics issues list) by Becker/Dodgson:

Move we close issues #7 and #8 in N0844 = 96-0026 with no change to the WP.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

54) Motion (to amend numeric_limits) by Becker/Dodgson:

Move we amend the WP as follows:

-- remove 18.2.1.1 [lib.numeric.limits] paragraph 2 and paragraph 3 (this information already appears in 18.2.1.2).

-- change 18.2.1.1 [lib.numeric.limits] paragraph 4 to replace "meaningless (0 or false)" with "0 or false".

-- change 18.2.1.2 [lib.numeric.limits.members] paragraph 2 to remove ", denorm_min()".

-- change 18.2.1.2 [lib.numeric.limits.members] paragraph 22 to include the description for rounding error from the LIA-1 standard (either directly or by reference).

-- change 18.2.1.2 [lib.numeric.limits.members] paragraph 23 (min_exponent) to

Minimum negative integer such that radix raised to the power of 1 less than that integer is a normalized floating-point number.

-- change 18.2.1.2 [lib.numeric.limits.members] paragraph 25 (min_exponent10) to

Minimum negative integer such that 10 raised to that power is in the range of normalized floating-point numbers.

-- change 18.2.1.2 [lib.numeric.limits.members] paragraph 27 (max_exponent) to

Maximum integer such that radix raised to the power of 1 less than that integer is a representable finite floating-point number.

-- change 18.2.1.2 [lib.numeric.limits.members] paragraph 29 (max_exponent10) to

Maximum integer such that 10 raised to that power is in the range of representable finite floating-point numbers.

-- change 18.2.1.2 [lib.numeric.limits.members] paragraphs 21, 24, 26, 28, 30, 32, 35, 38, 41, 53, and 62 to:

Meaningful for all floating point types.

-- change 18.2.1.2 [lib.numeric.limits.members] paragraphs 44, 46, and 48 from "only in specializations" to "for all specializations".

-- change 18.2.1.3 [lib.round.style] (flt_round_style) to include the following text:

The rounding mode for floating-point addition is characterized by the values:

round_indeterminate is indeterminable
round_toward_zero is toward zero
round_to_nearest is to nearest
round_to_infinity is toward positive infinity
round_toward_neg_infinity is toward negative infinity

-- change the example in 18.2.1.4 paragraph 2 by replacing the declaration for is_iec559 with:

static const bool is_iec559 = false;

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

55) Motion (to disassociate assignment of valarrays from resizing valarrays) by Becker/Swan:

Move we amend the WP as follows:

-- In [lib.valarray.assign] 26.3.1.2 change paragraph 1 to:

Each element of the *this array is assigned the value of the corresponding element of the argument array. The resulting behavior is undefined if the length of the argument array is not equal to the length of the *this array.

-- In [lib.valarray.access] 26.3.1.3 change paragraph 5 to:

The reference returned by the subscript operator for a non-constant array is guaranteed to be valid until the member function resize(size_t, const T&) (_lib.valarray.members_) is called for that array or until the lifetime of that array ends, whichever happens first.

-- In [lib.valarray.access] 26.3.1.3 remove paragraph 6

-- In [lib.valarray.members] 26.3.1.7 add after paragraph 11 (member function free()):

void resize(size_t sz, const T& c = T());

This member function changes the length of the *this array to sz and then assigns to each element the value of the second argument. Resizing invalidates all pointers and references to elements in the array.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

56) Motion (to allow scalar assignment to valarrays) by Becker/Swan:

Move we amend the WP as follows:

-- In [lib.template.valarray] 26.3.1 and in [lib.valarray.members] 26.3.1.7, remove the fill member function and its annotations.

-- In [lib.template.valarray] 26.3.1, following the copy-assignment operator, include a scalar assignment operator prototype:

valarray<T>& operator=(const T&);

-- In [lib.valarray.assign] 26.3.1.2 after paragraph 1, add:

valarray<T>& operator=(const T&);

The assignment operator causes each element of the *this array to be assigned the value of the argument.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

57) Motion (to ban order dependencies in valarray expressions) by Becker/Dawes:

Move we amend the WP by adding the following paragraph at the end of both [lib.valarray.assign] 26.3.1.2 and [lib.valarray.cassign] 26.3.1.6:

If the value of an element in the left hand side depends on the value of another element in the left hand side, the resulting behavior is undefined.

Vandevoorde explained that motions 55, 56 and 57 allow a C++ implementation to implement valarray as an intrinsic class which can take advantage of array-processing hardware.

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

==== presented by Gibbons ====

58) Motion (to specify the template compilation model) by Spicer/Ball:

Move we amend the WP as specified in section 3.1 of N0875 = 96-0057.

Motion passed X3J16: 28 yes, 13 no.

58a) Motion by Lajoie/Plum:

Move we reconsider Motion 58.

Motion passed X3J16: lots yes, 0 no, 6 abstain.

58b) Motion by Lajoie/Plum:

Move we table Motion 58, to allow more time for investigation of related technical issues, and to allow more time for WG21 and X3J16 to form consensus; and direct the Project Editor to incorporate the N0875 = 96-0057 wording into Editorial Boxes in the WP, noting that WG21 has no consensus on compilation model.

Motion passed X3J16: lots yes, 0 no, 7 abstain.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

WG21 asked that the minutes note that WG21’s prior straw vote on this proposal was: 3 yes, 2 no, 1 abstain.

==== presented by Dawes ====

59) Motion (to allow writing "nothrow" instead of "nothrow()") by Dawes/Corfield:

Move we amend clause 18.4 [lib.support.dynamic] as follows:

-- in header <new> synopsis, replace:

struct nothrow{};

with:

```
        struct nothrow_t{};
        const nothrow_t nothrow;

    --  in 17.3.1.1 [lib.contents] Table 17 - Standard Structs, change
        "nothrow" to "nothrow_t".

    --  throughout the WP, change "nothrow&" to "nothrow_t&".
```

Motion passed X3J16: lots yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

11.2 Issues delayed until Friday

Koenig thanked the people who would be staying over the weekend to edit the draft. He apologized that he could not stay because of a prior commitment. Clamage thanked Koenig for all his work as editor. Applause.

12 Plans for the future

12.1 Next meeting

No discussion.

12.2 Mailings

Lajoie said she must receive documents by March 29 for them to make the post-meeting mailing.

12.3 Following meetings

Harbison listed the dates and locations for the upcoming meetings:
- 7-12 July 1996 in Stockholm, Sweden, hosted by Ericsson
- 10-15 November 1996, Kona, Hawaii, USA hosted by Plum Hall
- 9-14 March 1997, Nashua, New Hampshire, USA hosted by Digital
- 13-18 July 1997, somewhere near London, UK, hosted by Programming Research
- 9-14 November 1997 (location and host to be determined)

Harbison said he would inform SC22 that our project schedule has slipped (because we did not submit the WP for a second CD ballot as planned). However, he would not submit a new project schedule to SC22 until after the next WG21+X3J16 meeting.

Clamage said Dawes would continue coordinating the Library WGs.

Scian thanked Lajoie (again) for her work as Vice-chair of X3J16.

13 Adjournment

Motion to adjourn passed WG21+X3J16: lots yes, 0 no.

The meeting adjourned at 11:15 on Friday.

Appendix A - Attendance

Name	Affiliation	M	Tu	W	Th	F
Landauer, Doug	Apple Computer	A	A	A	A	A
Winder, Wayne	Asymetrix	V	V	V	V	V
Koenig, Andrew	AT&T Research	V	V	V	V	V
Stroustrup, Bjarne	AT&T Research	A	A	A	A	A
Becker, Pete	Borland	V	V	V	V	V
Swan, Randall	C-Team	V	V	V	V	V

Corcoran, Marian	Center for Adv Tech	A	A	A	A	A
Allin, Glenn	Centerline Software	V	V	V	V	V
Charney, Reg	Charney & Day	V	V	V	V	V
Pagel, Mark	Cray Research	V	V	V	V	V
Knuttila, Kim	Cygnus Support	A	A	A	A	
Merrill, Jason	Cygnus Support	A				
Stump, Mike	Cygnus Support	V	V	V	V	V
Schwarz, Jerry	Declarative Systems	V	V	V	V	V
Meyers, Randy	Digital	V	V	V	V	V
Phillimore, Coleen	Digital	A	A	A	A	
Ward, Judy	Digital	A	A	A	A	A
Whitman, Sandra	Digital	A	A	A	A	A
Bruck, Dag	Dynasim AB	V	V	V	V	V
Millard, Andy	Edinburgh Portable Compilers	V	V	V	V	V
Adamczyk, Steve	Edison Design Group	V	V	V	V	V
Anderson, Mike	Edison Design Group	A	A	A	A	A
Spicer, John	Edison Design Group	A	A	A	A	A
Jonsson, Fredrik	Ericsson	V	V	V	V	V
Coha, Joseph	Hewlett-Packard				V	
Feng, Yinsun	Hewlett-Packard	A	A	A		V
Lenkov, Dmitry	Hewlett-Packard	V				
Lajoie, Josee	IBM	V	V	V	V	V
Colvin, Greg	IMR	V	V	V	V	V
Nelson, Clark	Intel	V	V	V	V	V
Suto, Gyuszi	Intel	A	A	A	A	A
Andersson, Per	Ipso Object Software	V	V	V	V	V
Kamimura, Tsutomu	Japan		A	A	A	
Koshida, Ichiro	Japan	A	A	A	A	A
Allison, Chuck	LDS Church	A	A	A	A	
Munch, Max	Lex Hack & Associates	A	A	A	A	A
Dum, Stephen	Mentor Graphics	V	V	V	V	V
Pennello, Tom	MetaWare	V		V		V
Reeves, Jack	Microcybernetics		A			
Eager, Michael	Microtec Research	A	A	A	A	A
Saini, Atul	Modena Software	V	V	V	V	V
Corfield, Sean	Object Consultancy Services	V	V	V	V	V
Benito, John	Perennial	V	V	V	V	V
Krit, Habib	Perennial	A	A	A		
Kumoluyi, Akin	Pilkington Microelectronics	A	A	A	A	A
Plum, Tom	Plum Hall	V	V	V	V	V
Southworth, Mark	Programming Research	V	V	V	V	V
Wilcox, Thomas R.	Rational Software	V	V	V	V	V
Glassborow, Francis	Richfords	A	A	A	A	A
Kim, Tom	Rogue Wave Software	A				A
Langer, Angelika	Rogue Wave Software		A	A	A	
Le Mouel, Philippe	Rogue Wave Software	A	A	A	A	A
Smithey, Randy	Rogue Wave Software	V	V	V	V	V
Vandevoorde, David	RPI	A	A	A	A	A
Saks, Dan	Saks & Associates	V	V	V	V	V
Koch, Gavin	SAS Institute	V	V	V	V	V
Schilling, Jonathan	SCO	V	V	V	V	V
Kiefer, Konrad	Siemens AG	V	V	V	V	V
Hartinger, Roland	Siemens Nixdorf	V	V	V	V	V
Unruh, Erwin	Siemens Nixdorf	A	A	A	A	A
Austern, Matthew	Silicon Graphics	A	A	A	A	A
Kung, Michael	Silicon Graphics	A				
Wilkinson, John	Silicon Graphics	V	V	V	V	V
Miller, William M.	Software Emancipation Tech	V	V	V	V	V
Ball, Mike	Sun Microsystems	V	V	V	V	V
Clamage, Steve	Sun Microsystems	A	A	A	A	A
Gafter, Neal	Sun Microsystems	A	A	A	A	
Harbison, Sam	Tartan	V	V	V	V	V
Rumsby, Steve	UK	A	A	A	A	A
Dodgson, Dave	Unisys	V	V	V	V	V
Hauer-Lowe, Keith	Unisys	A	A	A	A	A
Scian, Anthony	Watcom	V	V	V	V	V

Welch, Jim	Watcom	A	A	A	A	A
Plauger, P. J.	WG14	A	A	A	A	A
Hedquist, Barry	X3 / OMC					A
Crowfoot, Norm	Xerox	V	V	V	V	V
Swaminathan, Kumar	Xerox	A	A			
Dawes, Beman		V	V	V	V	V
Eckel, Bruce		A				
Gibbons, Bill		A	A	A	A	A
Myers, Nathan		A	A	A	A	A
Tooke, Simon		A	A	A	A	A
Total Voting		41	39	40	40	41
Total Attending (including Voting)		77	74	73	71	68