Document Number: WG21/N0817
X3J16/95-0217
Date: 30 January 1996
Project: Programming Language C++
Reply to: Dan Saks
dsaks@wittenberg.edu

X3J16 Meeting No. 19
WG21 Meeting No. 14
5 - 10 November 1995

Kikai-Shinko-Kaikan Bldg.
Tokyo, Japan

1 Opening activities

Clamage convened the meeting as chair at 09:20 (JST) on Monday, 6 November 1995. Lajoie was the vice-chair, and Corfield was the secretary.

IBM Japan (represented by Kamimura) hosted the meeting.

1.1 Opening comments

1.2 Introductions

Clamage said that applications for chair of X3J16 can be submitted until Friday.

Corfield 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

Lajoie noted that X3J16 had only two members above quorum. She asked voting members to be sure to be present on Friday. Lajoie also explained that STR and Object Consultancy Services did not have voting rights because this was their first meeting.

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

Kiefer said he brought a new version of a proposal to add ’long long’ integral types to C++. Plauger brought a new version of the library issues. Others had revised issues lists for other WGs. Lajoie brought a paper on program start and termination issues. Harbison provided a paper summarizing the CD Ballot votes.

Plum said he would have a paper on extended identifiers.

1.5 Approval of the minutes of the previous meeting

Corfield submitted the minutes from the previous meeting (N0734 = 95-0134) for approval with the following corrections (posted to the -all reflector by Saks):

-- Under item 1.6, paragraph 1, add a new second sentence:

The agenda was not available electronically and so some (possibly many) of those present had not seen the agenda yet.

-- Under item 6.1 "Core (Adamczyk)", after the paragraph:

Plum recalled that we earlier proposed to disallow references in

unions, but Skaller objected because it would preclude an extension he wanted to propose. But we didn't approve that proposal.

Insert the following new paragraph:

Corfield said it was the UK that objected to the previous attempt to disallow references in unions; they objected on the grounds of consistency. He also pointed out that the motion was defeated by WG21 in San Diego (3 yes, 4 no, 1 abstain) and that it was politically unsound to retake the vote in Monterey when three of the four NBs that voted 'no' were not present.

-- Under item 6.1 "Core (Lajoie)", under ingredient 5 of Schwarz's presentation on the ODR, fix the following errors in the example:

```
// in one file

class B {
    B(int);
    B(int, int);
};
B(int = 0) { }              <- replace 'B' with 'B::B'
class D : public B { }
D d1;

// in another file

class B {
    B(int);
    B(int, int);
};
B(int = 0, int = 0) { }     <- replace 'B' with 'B::B'
class D : public B { }
D d1;
```

and fix the same error two examples later:

```
class B {
    B(int);
    B(int, int);
};
B(int = 0) { }      // forbid this  <- replace 'B' with 'B::B'
class D : public B { }
D d1;
```

-- Under item 11, paragraph 2, sentence 2, change '"two-rule"' to '"two-week" rule'.

-- Under item 11.2, under the action items for "Core WG (Lajoie)" change "Glassborough" to "Glassborow".

Motion by Lajoie/Bruck:

Move we approve N0734 = 95-0134 as the minutes of the previous meeting with these corrections.

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

1.6 Agenda review and approval

Clamage submitted the proposed agenda (N0712 = 95-0112) for approval.

Koenig asked if we should hold a technical session on input iterators. The members agreed to deal with this as part of the Library WG (working group) discussions.

Motion by Dawes/Rumsby:

Move we accept N0712 = 95-0112 as the agenda for this meeting.

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

1.7 Report on the WG21 Sunday meeting

Harbison summarized his report. The CD ballot vote was: 10 yes, 6 no with comments. Harbison explained that the comments included the various WG issues lists and he has asked the various NBs (national bodies) to track their own issues so that he can formulate the disposition of comments after the Scotts Valley meeting. He also reported that SC22 approved our revised schedule with a second CD ballot and a two meeting turnaround for the disposition of comments (as in N0755 = 95-0155).

Harbison explained SC22's concern that WG21 changed the draft during the balloting. He also explained that, for the Stockholm meeting (July '96), we should not take any official votes that would change the draft but rather continue to work on issues and resolve problems. Lajoie said we agreed to take straw votes and modify the WP (Working Paper) accordingly, but delay any formal vote(s) on changes until the meeting in Hawaii (November '96). Harbison said he'd draft proposed procedures for the Stockholm meeting.

Koenig wondered if he should produce a revised WP for mailing after the Stockholm meeting. He suggested two alternatives: 1) He could produce an unofficial WP with a cover sheet explaining that it only represents the project editor's working notes, or 2) he could not publish a WP at all and send out a machine-readable copy of his latest version on request. Plum and Stroustrup preferred the latter. In any event, Koenig felt we needed some document to work from in Hawaii. Harbison suggested dropping this discussion this until he has spoken further with Bill Rinehuls of SC22.

Plum said SC22 was not convinced that we should even hold a meeting during the balloting, but Harbison had persuaded them that we could do useful work. Plum also noted that NBs that submit comments early should not get favored treatment over NBs that submit their comments later.

Stroustrup supported Koenig's suggestion not to produce an official WP, but he wanted to have an internal document that incorporates changes made at the Stockholm meeting. Gibbons expressed some concern that we would revisit issues from Stockholm at the Hawaii meeting if we do not have an approved WP. Plauger said there is clearly a tradeoff between getting work done and following the rules. We can't officially close issues at the Stockholm meeting; we must risk that issues seemingly resolved in Stockholm will come up again in Hawaii.

Koenig emphasized the need for a "feature freeze" in order to be able to fix and improve the wording. He suggested that the appropriate point for that is when we vote to submit the final CD. After that the committee should be prepared to work on editorial issues only. Bruck felt there was already consensus on this issue.

Lajoie was concerned that, if we give the impression that we can't do any "real" work in Stockholm, members may be reluctant to attend, even though we certainly need people there. Harbison said we should already be in the frame of mind where we are dealing only with NB comments. Stroustrup agreed, but said that sometimes the best fix is to make a wider change which may also fix other acknowledged problems.

Welch asked if we had to go through NB channels to fix a clear error. Plauger said no, but we cannot invent new stuff.

Harbison explained that ISO rejected the proposed JTC1 DIS procedures. The effect is that we are allowed to edit the draft during DIS ballot to resolve NB comments. Moreover, if we fail that ballot, we will be able to stage another DIS ballot without falling back to another CD ballot. However, he recommended that we proceed under the assumption that we intend to produce a camera-ready DIS anyway.

1.8 Liaison reports

1.8.1 WG14+X3J11 (C)

Plum reported that WG14+X3J11 met in Nashua, NH, USA from 16-20 October 1995. They are working on the five-year revision of C, known informally as "C9x".

Plum conveyed WG14's request that the implicit int "ban" should not allow const and volatile as the only type-specifiers. That is,

```
const N = 9;
void f(const i);
```

should be invalid.

Koenig asked Plum about WG14's stance on 'int main', and Plum said he wasn't sure.

Plum said C9x is moving toward tag compatibility of structures across translation units, and generally tightening up the linkage rules. Plum said he's urged the C committees to adopt the C++ treatment of tagless structs.

Plum emphasized that we should view this as a bidirectional compatibility effort. Koenig wondered if there is a danger of producing a C++ standard that's compatible with C features that C9x has removed.

Plum said WG14 has starting voting on changes to C9x. The preliminary results indicate sentiment to add the following features:
-- designated initializers
-- the keyword 'restrict'
-- classes with member functions, single inheritance, virtual functions, access restrictions

Plum said WG14 intends this subset to be compatible with C++. It appears that C will not have constructors and destructors (see below), but will require operator new to create polymorphic objects.

Plauger said the C committee is advancing very slowly and these preliminary votes are not necessarily representative of what the committee will eventually do. Stroustrup emphasized the importance of remaining compatible in both directions, but noted that not adopting constructors and destructors may lead to noticeably different styles of programming in the otherwise common subset of both languages. Plum said he felt that C++ and C9x differ in that C++ was designed as a leading edge experiment while C9x is building on an established language.

Plum reported that the C committees are also considering adding:
-- bool, true, false (as a typedef/macro in a separate header)
-- tag compatibility requirements
-- variable length arrays (as in Cray and gcc)
-- inline
-- complex
-- extended identifiers and extended literals
-- // comments

and have already rejected:
-- overloading
-- constructors and destructors

Gibbons asked about the numerical extensions that have been proposed. Plum was unsure where they stood.

1.8.2 WG20 (internationalization)

Kung is not the official liaison, but he offered a report. He said WG20 is considering internationalization issues for language design and Unicode support. Both of these are probably too late for C++ but will likely be proposed for C9x.

1.9 New business requiring actions by the committee

None.

1.10 Drafting committee

Corfield said he'd manage the technical aspects of the drafting committee and Rumsby agreed to manage the administrative aspects. Corfield asked the WGs to prepare their motions on Tuesday evening so that the drafting committee can simply merge and edit them. Corfield asked for additional volunteers for the drafting committee on Wednesday evening.

1.11 Organization of WGs

The WG21+X3J16 prepared to break in WGs.

Koenig volunteered to give a talk about input iterators from 17:30 to around 18:30 on Monday night.

The committees recessed to WGs at 11:25 and reconvened on Wednesday at 9:10.

2 WG sessions

3 Technical session

4 WG sessions

5 Working Paper for Draft Proposed Standard

5.1 Changes in the Working Paper

Koenig presented the project editor's report (N0811 = 95-0211). He thanked the volunteers who helped edit the draft at the end of the Monterey meeting and in the weeks thereafter. He observed that the changes in the WP (both the number of lines touched and the number of separate changes) has decreased substantially over the last few WPs. This sign of progress is "good news".

He said he did not make one of the changes approved in Monterey. Issue 21-0646 (from the clause 21 issues list) called for changing some return types from basic_string& to basic_string<charT, traits, Allocator>&. In this context the types are equivalent, so the change wasn't necessary.

Koenig summarized some general editorial changes he made, including
-- changing "processor" to "implementation", and
-- adding text to explain that apparent requirements on programs are abbreviated forms of requirements on implementations.

He also made some "bold changes" -- changes that were editorial proposals appearing in editorial boxes (see N0811 = 95-0211 for the list of changes).

Koenig said he intends to make the following bold changes in the next draft:
-- Add specializations of the swap template for each container that has a 'swap' member. Currently 'swap(a, b)' works for all containers

but may be inefficient. This proposal uses specializations of the global 'swap' template to call the 'swap' member of the standard containers.
-- Reinstate <iostream.h>, which includes <iostream> and contains the appropriate code to match other '.h' headers.

Plauger noted that the WP specifies in detail what the .h headers actually do for the C compatibility headers, but not for the other .h headers. Koenig suggested this should be a Library WG issue.

Koenig said he intends to make a careful editorial pass through the language clauses over the next year. He asked for volunteers to do the same for the library clauses.

Straw Vote: Who approves N0785 = 95-0185 as the current WP? lots yes, 0 no.

(See Motion 1.)

6 General session I

6.1 Core Language WG

==== Adamczyk ====

Adamczyk discussed a proposal to allow declarations of operator void() in templates, but not allow calls to such functions (N0720 = 95-0120). He explained that there is precedent for allowing declarations of unusable entities such as derived-class-to-base-class conversion operators.

Adamczyk said the subgroup agreed to allow declarations and definitions of operator void() everywhere. They also agreed that such operators can be called only using explicit operator call notation such as x.operator void(). They cannot be called implicitly. Stroustrup asked for assurance that (T)x, where T is void, would not invoke an operator void(), which he got.

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

(See Motion 2 for the formal wording.)

Adamczyk presented proposed resolutions to issues regarding function typedefs (N0792 = 95-0192). The subgroup agreed to all points in the paper, but Gibbons had objections to the last issue (regarding const in a function typedef). Gibbons and Schreiber (author of N0792 = 95-0192) proposed to resolve the issue as per the following example:

```
typedef void G();
typedef void F() const;
struct X {
    const G g;  // ill-formed
    F f;        // well-formed
};
```

(See Motion 5.)

Stroustrup expressed surprise that a member function can be declared using a typedef. Gibbons said we had previously decided that this would not be valid inside a template.

Gibbons explained the point of the proposal more-or-less as follows: If you want a cv-qualifier to apply to a function type, you must write it in the function declarator for that type, as in the declaration of F above. You can't declare a cv-qualified function type by combining a cv-qualifier with a function type in the decl-specifier-seq of a later declaration.

Koenig said he supported this decision. He has observed a bug/extension in Cfront which permits declaring typedefs for member function types. This proposal does not preserve that bug.

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

Adamczyk recommended clarifying initialization of call-by-value objects (as per N0780 = 95-0180). Specifically,
-- access checking is performed on the caller side not the callee side,
-- the lifetime of a parameter is well-defined and it is destroyed immediately at the end of the function.
(See Motion 3.)

Adamczyk explained that in an expression such as f(x) + f(y), "temporaries" generated to pass x and y by value to f are in fact parameters, so the destruction of those parameters occurs immediately after each call, prior to the addition.

Koenig asked Adamczyk to ensure that the formal wording for the WP acknowledges the "Schwarz" optimization (allowing elision of copy constructor calls). Expanding on this issue, Adamczyk noted that the lifetime of the copied parameter is likely to be shorter than the lifetime of any temporaries introduced.

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

Adamczyk said the subgroup looked a proposal to allow cv-qualified constructors (N0798 = 95-0198), but rejected it because it's an extension.

Adamczyk presented a proposal to refine the WP's definitions for "scalar type" and "fundamental type" (N0774 = 95-0174). The substance of the proposal is to change:
-- scalar type to include pointer-to-member
-- reclassify enumerated types as compound types (instead of as fundamental types).
Adamczyk said the subgroup agreed to accept this as editorial.

Bruck asked if enumerated types are PODs. Koenig and Lajoie said yes. Clamage wanted to be sure there are no semantic changes implied. Kiefer asked for a vote on the proposal as a formal motion, which he got (see Motion 4).

Unruh noted that the paper implies that C++ would have to allow bitwise copying (e.g., using memcpy) of pointers-to-members. This interacts with an upcoming proposal on pointer-to-member casts.

Adamczyk presented a proposal to clarify how the initialization order of non-local objects depends on whether initialization is static or dynamic (item 555 of N0802 = 95-0202). He summarized the proposal as follows. By the time a non-local object is completely initialized:
-- all preceding non-local objects are also completely initialized
-- following non-local objects may have been initialized statically.
This is a relaxation of the current ordering. (See Motion 6.)

Adamczyk used the following example to clarify the proposal:

```
struct A {
    float x;
    A(float p) : x(p) { }
};
extern A b;
extern A a(b.x);
A b(1.0);
```

He said this is the sort of program that "should not be written" so it's

unspecified whether 'b' is initialized when 'a' is defined. The current WP guarantees that 'b' is zero-initialized prior to initializing 'a'.

He went on to explain that all static initialization is done first. The dynamic initialization is done in the canonical order. This proposal relaxes requirements to allow an implementation to turn some the dynamic initializations into static initializations. This can produce unspecified behavior (as in the example above).

Gibbons said this proposal might break the "as if" rule, but he favored it nonetheless. Adamczyk agreed that this could happen for the initializations involving forward references, which depend on unspecified behavior. Clamage was concerned that the proposal would change the behavior of well-formed programs. Stroustrup felt the optimization potential was more important than the behavior of examples such as the one above.

Unruh asked if an implementation can optimize the initialization of a class object by initializing parts of the object statically and other parts of the same object dynamically. Lajoie said the proposed rule did not address this, but nothing in the draft prevented it either.

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

==== Lajoie ====

Lajoie said her Core subgroup discussed issues regarding program start and termination (from N0802 = 95-0202). In the following, numbers in () are core issue numbers.

Lajoie began with (551). The WG recommended that all C++ programs must have a main function, including those produced by freestanding implementations.

Stroustrup wanted to consider requirements for C++ libraries called from other languages. Unruh suggested an alternative wherein a freestanding implementation need not require 'main'. Koenig agreed that the standard could say the only portable way to provide 'main' is to write it explicitly, but implementations could provide an alternative as an extension. Stroustrup said it was important not to exclude mixed-language programs. Lajoie agreed to try rephrasing this part of the proposal to satisfy Unruh's and Stroustrup's concerns.

Lajoie said that, for (462), the WG recommended that calling 'exit' during destruction of an object with static storage duration has undefined behavior. Also, for (429), they recommended that reference initialization is part of static initialization.

Lajoie presented new terminology for expressing initialization semantics. The new term "reference constant expression" means "an lvalue designating an object with static storage duration". Further details appear in N0802 = 95-0202.

Stroustrup asked if "reference initialization is part of static initialization" is a requirement on implementations. Lajoie said yes. Rumsby asked if there's an interaction between reference constant expressions and operator&. Lajoie said no. Stroustrup expressed concerned that this can't be implemented. Adamczyk explained that the proposed wording directly parallels the words in the C standard about address expression initialization.

Lajoie said that, for (430), the WG recommended that objects with static storage duration are destroyed in the reverse of the order that their constructors completed. This is true for _all_ such objects in a program, including local static objects.

Clamage asked about dynamic libraries. Lajoie explained that dynamic

libraries are already outside the scope of the WP (they're an extension) so it doesn't matter if dynamic libraries cannot follow the proposed rule.

Clamage asked if this implies that an implementation must keep track of the exact order of all constructions in order to satisfy this requirement. Lajoie said yes, the implementors present in the subgroup did not feel this was too onerous. She added that the implementation needs to track the construction only for "outermost" objects. That is, statics in an aggregate will be destroyed only after the whole object is destroyed. Unruh asked if "completion of construction" applies to each array element separately or to the array as a whole. Lajoie replied that it applies to the array as a whole. Therefore, another object constructed within the constructor of an array element will be destroyed after the entire array is destroyed.

Someone asked what happens if a local static is constructed during the destruction of objects with static storage duration. Lajoie explained that it gets constructed, and then it gets destroyed immediately after the "current" destruction completes; however, if the local static object in question had already been destroyed, the behavior is undefined. Corfield asked if this covers a local static that's being constructed for the first time. Lajoie was not sure. Corfield pointed out that this is one of the issues that concerns the UK. Welch said that the subgroup intended to cover only local objects that had already been destroyed.

Clamage felt this proposal attempts to provide too many guarantees. Lajoie said the proposal covers only what several implementations already do. Clamage said that may just indicate that those implementations have not been exercised heavily enough.

Unruh expressed concern about the destructor for an object that is initialized statically. Destructors execute in reverse order of the constructors. But there's no point in time when the constructor for that object executes, so when does the destructor execute? Lajoie said it was a good question but a separate issue.

Lajoie said the, for (484), the WG recommended that destructor calls should interleave with calls to the atexit functions. Using this example:

```
void f() { static T t1(1); }
void g() { static T t2(2); }
main() {
    atexit(f);
    atexit(g);
    f();
    exit(0);
}
```

she explained that, by this proposal, the steps at exit would be:

1. g() called
2. t2 destroyed
3. f() called
4. t1 destroyed

Unruh asked whether destruction was guaranteed to occur between calls to registered functions. Lajoie said that order of construction was the determining factor. Unruh said this implies that the implementation must track atexit functions as well as static object construction. Lajoie said yes.

Welch stepped through the example above showing what the implementation must track. He said that the order was not, in fact, as given above. Actually t1 is destroyed _before_ g is called, so the call to f from the

atexit chain causes undefined behavior. Schreiber suggested moving the call to f ahead of the atexit(f) call to show a well-defined ordering:

```
main() {
    f();
    atexit(f);
    atexit(g);
    exit(0);
}
```

In this case the process at exit would be:

1. call f and construct t1
2. register f
3. register g
4. call exit, which calls g and constructs t2
5. destroy t2
6. call f
7. destroy t1

Gibbons was surprised that atexit affects destruction ordering in this way. Welch confirmed that the WG intended this. Gibbons asked if this means atexit calls look a lot like constructor calls. Welch said yes, you can simulate this process by using static objects that register a function on construction and call it on destruction. Someone suggested deprecating atexit.

(Motion 7 formalizes the recommendations above.)

Lajoie said the working group agreed that the following are editorial corrections (from N0802 = 95-0202):

-- (UK 38): The standard should not say that exit's argument is returned to the program's environment.
-- (552) Confirm that calling exit may not destroy objects with automatic storage duration
-- (527) Confirm that nonlocal objects with static storage duration need not be initialized before entering main.
-- The WP should state explicitly that main's return type is int (N0772 = 95-0172).

Lajoie said the WG rejected an as extension a proposal to specify the initialization order for global objects (N0717 = 95-0117).

Lajoie explained that N0772 = 95-0172 simply prohibits non-int return types for 'main'. Koenig said he wanted an exemption to allow an implicit int return type for 'main'. Lajoie said this is a separate issue. Spicer agreed with the proposal, but asked for a formal vote instead of treating it as editorial. He wanted to make it clear that we consciously decided that 'void main()' is ill-formed.

Lajoie said her Core subgroup discussed the following memory model issues (from N0803 = 95-0203). Again, numbers in () are core issue numbers.

Lajoie explained that, for (554), the WG recommended that if a program attempts to construct an object in storage once occupied by a const object that's been destroyed, the program's behavior is undefined. Thus, an implementation may place const objects in readonly memory even when those objects have constructors.

Gibbons asked how this applies to dynamically-allocated objects. He pointed out two additional cases not yet covered:
1. using new to create a const object, and
2. using new with placement to construct a const object.
Lajoie said she'd add issues for these.

Lajoie said that, for (UK 611), the WG recommended that a program may use memmove to copy objects that do not overlap. (The WP already allows a program copy objects using memcpy.)

Unruh asked if an explicit loop that copies objects byte-by-byte as unsigned chars would also work. Lajoie said no. Plauger explained the problem as follows. Some mem* functions in the C library have parameters of type char * instead of unsigned char *. On a machine that uses sign-magnitude arithmetic and uses a signed representation for plain char, "value collapse" could occur. That is, copying bytes as signed char might convert -0 to +0, thus changing bit patterns as it copies. So memcpy is special because it promises a "transparent" copy (one that preserves the bit pattern) regardless of value collapse.

Lajoie agreed to add an issue for moving objects byte-by-byte as unsigned chars.

The next issue was (417). The WG recommended that, if an operand of [], ++, --, +, -, +=, or -=, has static type T * a dynamic type that is not T *, the behavior is undefined. This restriction does not apply to other operators, such as indirection, relational operators, and equality operators. Hence, these operators may apply to an operand of type T *, where T is an abstract base class.

Lajoie then presented (597). The WG recommended that adding zero to a null pointer produces a null pointer; subtracting two null pointers produces zero. Koenig explained that this allows a program to use two null pointers to represent an empty range. Unruh asked if this applies at run-time, for example, when adding a pointer variable and an integer variable. Lajoie said that was the intent. Clamage asked Lajoie to make the words clear that "zero" need not be just an integral constant expression. She agreed.

Lajoie said that, for (513), the WG recommended that pointer comparisons not described as "well-defined" should have unspecified behavior, not undefined behavior. Such comparisons will yield some sort of truth value rather than a possible machine exception. Unruh asked if repeated comparisons could yield different results. Lajoie said yes, the result is unspecified. Corfield asked whether the subgroup had considered architectures where avoiding a machine fault might be expensive. Lajoie said no one seemed to think this was likely.

(Motion 8 formalizes the recommendations above.)

Lajoie said the working group agreed that the following are editorial corrections (from N0803 = 95-0203):

-- (UK 382) Change the term "unusable value" to "indeterminate value".
-- (UK 388) Change the term "valid storage" to "allocated storage".
-- (557a) Remove the term "well-defined copy operation" because it's meaningless.
-- (557b) Preserve the term "value representation", but define it better.
-- (471.2 and 471.3) Clarify when a program can access the operand of a delete-expression.
-- (93) The behavior of 'delete this' in a member function is undefined. (It need not be diagnosed.)
-- (596) Clarify the meaning of a relational operator when only one operand is the null pointer.
-- (476) Clarify that accessing an object with indeterminate value causes undefined behavior.

Lajoie said her Core subgroup discussed the following object model issues (from N0804 = 95-0204). Again, numbers in () are core issue numbers.

Regarding (569), Lajoie said the WG recommended that, in the storage

mapping for objects, the order of members separated by access specifiers is unspecified rather than implementation-defined. Unruh said an implementation-defined ordering might be useful. Clamage said the standard should not prescribe an ABI (an "application binary interface"). Adamczyk said the subgroup did not want to require implementations to document something that might be very difficult to formulate.

Lajoie then discussed (529). The WG recommended that zero-sized base classes are allowed, but no pair of pointers to base class subobjects can compare equal.

Kiefer asked how a subobject can have zero size. Adamczyk explained that sizeof always returns a non-negative value, even for an object with no members. Thus, the size of an object may be less than the sum of 'sizeof' applied to each of its members and subobjects.

The next issue was (589). The WG recommended that the relationship between the return types of overriding functions must be known at compile time, i.e., the return types must be pointers or references to complete types for derived classes. Lajoie gave this example to illustrate the proposal:

```
class A;
class B;
class C {
    virtual A& f();
};
class D : C {
    virtual B& f(); // ill-formed; A and B are incomplete
};
```

(Motion 9 formalizes the recommendations above.)

Lajoie said the working group agreed that the following are editorial corrections (from N0804 = 95-0204):
-- (OB1) Refine the definition of "object".
-- (533) An anonymous union is neither an object nor a type.

Lajoie said her Core subgroup discussed the following special member function issues (from N0806 = 95-0206). Again, numbers in () are core issue numbers.

Lajoie said that, for (575), the WG recommended that a program can refer explicitly to implicitly-declared special member functions.

Unruh asked if these functions can be declared as friends. Lajoie said it's possible, but probably useless. Corfield asked if the subgroup had looked at his recommendation that such functions should be callable and addressable (N0718 = 95-0118). The paper also suggests that implicitly-declared functions should be callable and addressable for arbitrary types in templates. Lajoie said the subgroup had not looked at this. Corfield said this issue needs to be addressed.

Lajoie said the group discussed (379) regarding destructor names. They agreed that the declaration for a destructor cannot use a typedef name, but an explicit call to a destructor can use a typedef name. Also, the object expression in an explicit destructor call must have the same type as the destructor's class type or (for virtual calls) must have a type derived from the destructor's class type.

Lajoie used this example to illustrate the proposal:

```
struct B {
    virtual ~B();   // must use B
};
```

```
B::~B() { }         // must use B
B* p;
typedef B B_alias;
p->~B_alias();      // okay to use B_alias
```

Unruh observed that ~B() can be parsed as a destructor call, or as the ~ operator applied to a constructor call. He said we could resolve the ambiguity by prohibiting unqualified destructor. Gibbons said the WP already does.

Spicer asked how the name lookup was actually performed after B::. Lajoie said this remains an open issue. Welch said Pennello has a whole list of issues like this which need to be resolved. Gibbons suggested that the lookup should be the same as for B::operator B().

Lajoie explained that, for (95), the WG recommended that a user-declared constructor for T taking volatile T& is a copy constructor. A user-declared assignment operator for T taking volatile T& is a copy assignment operator. Generated constructors and assignment operators in derived classes will call these. Generated constructors and assignment operators are never implicitly declared to take volatile references.

Unruh asked if writing

```
T::T(volatile T&);
```

for class T suppresses the usual copy constructor. Lajoie said yes. Welch said a program that does this for a class T will probably have many errors because it lacks the usual copy constructor.

The next issue was (574). The WG recommended that, if a class has a const or reference member, that member must be initialized either by a ctor-initializer list or by a brace-enclosed initializer list.

Gibbons said this means an implicitly-generated default constructor will be ill-formed (and the generation will fail). Adamczyk emphasized that this retains the existing incompatibility with C wherein C++ _requires_ initialization of const scalars.

Lajoie presented two more issues. For (478), the WG recommended that a union constructor shall initialize only one member of the union. For (534), the WG recommended that members of a nested anonymous union may be initialized by a constructor for its enclosing class.

Corfield asked if the constructor of the enclosing class can only initialize one member of the nested anonymous union. Lajoie said yes. Corfield said this affects the proposed WP wording. Gibbons asked if a const member of a union can be set only during initialization. Lajoie said sje'd take it as a Core issue.

(Motion 10 formalizes the recommendations above.)

Lajoie said her Core subgroup agreed that the following are editorial corrections (from N0806 = 95-0206):

-- (22) A program is ill-formed if it uses a special member function that's inaccessible at the point of use.
-- (576) const and volatile semantics never apply to an object during the execution of a constructor or destructor; const/volatile semantics apply to const/volatile objects only after their initialization has completed and only until their destruction starts.
-- (562) A copy constructor is a conversion function. An implicitly-declared copy constructor is not an explicit conversion constructor; it can be used for implicit type conversion.

Lajoie said the group considered a proposal to add a 'long long int' type to C++ (N0715R1 = 95-0115). They recommended rejecting it until we

know what WG14+X3J11 does about this issue. This means it will not appear in this standard but should be considered for the next revision of C++.

Kiefer was concerned that C would adopt this and C++ would not. Lajoie said that members of WG21+X3J16 who are concerned about this should lobby WG14+X3J11. Harbison said we could urge our liaison to encourage the C committee to adopt this. Plauger said he'd be glad to present the results of our straw vote on this to WG14+X3J16. Unruh requested a formal vote to indicate our support. Clamage preferred a straw vote. Lajoie suggested discussing this in a later session.

==== Gibbons ====

Gibbons reported that his Core subgroup discussed lookup for friend declarations. They proposed to ignore scopes outside the nearest enclosing namespace when looking for a name referred to in a friend declaration. This is to ensure that friends inject into the same namespace that they would be found in.

Gibbons explained the proposal with this example:

```
namespace A {
    void f();
    namespace B {
        class C {
            friend void f();    // refers to A::f
            friend void f(int); // injects A::B::f
        };
    }
}
```

According to the current WP, the two declarations of f do not overload because the first friend refers to A::f but the second is injected as A::B::f. Under this proposal, both f() and f(int) inject into A::B, so the f's declared in A::B::C become overloaded in A::B.

Stroustrup favored the proposal. Plauger expressed some concern about the possible impact on the library. Gibbons said a friend declaration can explicitly specify the namespace in which the name should be declared (i.e., looked up) by using qualified names.

Gibbons then proposed that a friend declaration containing a qualified name shall not be a definition. He also presented a proposal to change the implied meaning of an unnamed namespace from:

```
namespace UNIQUE {
    // body
}
using namespace UNIQUE;
```

to:

```
namespace UNIQUE { }
using namespace UNIQUE;
namespace UNIQUE {
    // body
}
```

The effect is the name x declared in the unnamed namespace can be referenced inside the namespace using ::x as well as just x.

Gibbons then presented a proposal to prohibit useless using-declarations. Under this proposal, the following will be ill-formed:

```
namespace A {
```

```
    typedef int Int;
}
namespace A {
    using A::Int;   // previously allowed as harmless
}
```

Next, Gibbons explained that a constructor does not have a name so it cannot appear in a using-declaration. The WG proposed to prohibit using-declarations that name destructors. Lajoie pointed out that the proposal on destructors from her Core subgroup makes this prohibition unnecessary. Gibbons said it may as well be added for clarity.

Gibbons presented a proposal to change the C++ grammar to allow unary :: (as in ::name) in all declarator-ids. He said this does not introduce any new syntactic ambiguities into C++. (There already is an ambiguity for pointer-to-member declarations that is resolved by a "maximal munch" parse and therefore may require parentheses to disambiguate.) Gibbons said the WG saw no reason to limit this change to friend declarations, so the proposal augments the grammar to allow a leading :: wherever it allows a qualified-name.

Gibbons then presented a proposal to consider only namespace names when looking up names in a namespace-alias-declaration or using-directive. Gibbons claimed that hiding namespace names makes no sense in these contexts. Stroustrup commented that this concept already holds for class names (in lookup for elaborated-type-specifiers). Gibbons agreed that this proposal simply extends the principle to namespace names.

Next, Gibbons said his Core subgroup discussed the semantics of unnamed namespaces. He said the WP currently employs the concept of "unique name", but this is a kludge. He proposed to change the linkage of names declared in an unnamed namespace to "non-external".

Koenig opposed this proposal, arguing that it prevents using private types with library templates. Stroustrup agreed with Koenig that we shouldn't prevent this. Gibbons said you could always use an "implementation" namespace to wrap the private types. Clamage felt that was a bit drastic if the intent was simply to prevent the unique name from "escaping". Gibbons said that unique names are, in fact, a fiction -- they are hard to generate and not really guaranteed to be unique.

After a bit more controversy, Gibbons agreed to take this issue back to the Core WG.

No one objected to the other proposals regarding namespaces. (See Motion 11.)

Gibbons proposed the following minor clarifications for typeid issues:

- typeid ignores top-level cv-qualifiers in its argument.
- typeid cannot have an argument with an incomplete class type.
- type_info is an incomplete type unless <typeinfo> is included (and the name type_info is not visible).
- when appearing as an argument to typeid, array and function types do not decay.
- type_info objects have static storage duration.

Gibbons presented revised text for all of clause 5.2.7 [expr.typeid]. He pointed out that the replacement text removes the discussion of 'typeid(p[N])', which is in line with the Core WG's proposal to prohibit subscripting of pointers to polymorphic types when the static and dynamic type differ.

Clamage asked why typeid should ignore the top-level cv-qualifier in its argument. Gibbons explained that otherwise programs must retain cv-qualifier information at run-time, and the WG thought this would incur too much run-time overhead.

Plum asked if the name std::type_info is available on demand. Gibbons said no. Clamage asked if this was like size_t. Gibbons replied no, because size_t is not a new type name, only a synonym.

Gibbons then introduced a proposal regarding pointer-to-member casts. He said the WG agreed on this proposal in Monterey (July '95) but did not submit it for a formal vote. In summary, the proposal:
-- disallows casts of pointers-to-members across virtual inheritance,
-- allow upcasts beyond the original member's class.
(See Motion 13 for details.)

Gibbons used the following example to illustrate the proposal:

```
class A { };
class B : public A {
public:
    virtual void f();
};
void g() {
    B* b = new B;
    void (B::*pf)() = &B::f;
    (b->*pf)();
    A* a = b;
    void (A::*pg)() = (void(A::*)())pf; // 1
    (a->*pg)();                         // 2
}
```

According to the current WP, the cast on // 1 has undefined behavior. This proposal sanctions the cast. In the dereference a->*pg on // 2, 'a' points to an object whose static type, 'A', that does not contain the member addressed by pointer-to-member 'pg'. However, 'a' points to an object whose dynamic type, 'B', does contain that member. This proposal requires that a pointer-to-member carry enough information to make this work.

Unruh argued that there is an implementation technique that cannot allow pointer-to-member dereferencing unless the static type of the object contains the actual member. Gibbons replied that existing commercial compilers already have support for the dereference, but currently perform optimizations that lose some of the necessary information. He later explained that the dereference is well-behaved only when the dynamic type of the object pointed to is actually a type derived from its static type.

Welch opposed the prohibition on conversions across virtual inheritance. Gibbons said no vendors that failed to implement this had users complain about it.

Straw vote: Who favors this proposal? lots yes, 2 no, some abstain.

6.2 Library WG

==== Dawes ====

Dawes presented the WG's proposals to resolve clause 17 [library introduction] issues (from N0801 = 95-0201). He began with issue 17-001, using the following example:

```
#include <string>
using namespace std;
int main() {
    cout << "Portable C++ code?" << endl;
    return 0;
}
```

The program refers to names declared in <ostream>, but does not include <ostream>. <string> might include <ostream>, or it might not; the WP does not say. Plauger pointed out the program is not portable but some implementations might successfully translate and execute it. The WG recommended that the WP should specify that headers cannot include other headers, or rather that they make visible only those names that they are specified to declare. (See Motion 15.)

Dawes then discussed issue 17-002, which raised a concern that the WP currently says "A C++ program shall not extend the namespace std." However, a C++ program must be allowed to extend the namespace std, if only to specialize class numeric_limits. The WG recommended adding "unless otherwise specified" to the prohibition on extending namespace std. This is specifically to allow specializations for numeric_limits.

Dawes said he'd add traits for the string and stream classes to the library issues list for review. Myers suggested adding the algorithms templates to the list. Dawes said there are templates in namespace std that users might want to specialize. Specializations must be in the same namespace as the original template, so user can't write such specializations unless that can extend namespace std.

Unruh asked whether the WP would provide an explicit list of allowable extensions. Dawes said the WG decided to use the "unless otherwise specified" form and that additional forms were up to the editor.

Gibbons asked about the is_specialized member of numeric_limits. Plauger explained that the is_specialized flag is always set true in any specialization. Only the template sets this flag false to warn that it is probably not supplying any useful information.

Spicer asked whether the prohibition on extending namespace std is diagnosable. Rumsby said the WG intended so. Spicer said there doesn't seem to be a distinction between user source and library source (making diagnosis harder). Plauger pointed out that library source was an implementation detail and so, in effect, it does have special status.

Unruh said this could be an expensive check for implementations. Koenig said an implementation could use "magic". Plauger said that making this diagnosable pretty much requires compilers to check all the names. Koenig responded that broken implementations are not required to diagnose themselves.

Dawes said that "shall not" has been in clause 17 for a long time; he suggested adding an issue to the list and then moving on. Plauger said the proposal reads as if implementations cannot extend their own headers. Rumsby said the "shall not" refers to C++ programs, not to the implementation itself. Koenig said this is the same situation as attempts by the user to extend the library in C programs.

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

Spicer said the definition of "extending the namespace" needs to be very clear. He asked why adding a specialization is an extension; it doesn't add any new names. Rumsby said we should just log the issue and move on.

Dawes went on the the next issue, that the WP does not describe the effect of a program that violates a library "Requires" paragraph. Dawes said the WG recommended that the effect is "undefined behavior unless otherwise specified".

(See Motion 16 for the formal wording of the previous proposals.)

Dawes said the WG agreed to resolve issue 18-013 by accepting the recommendation from the issues list (N0784R2 = 95-0174R2). Specifically, the

recommendation is that deleting a pointer obtained by nothrow operator new has well-defined behavior.

Corfield asked if there is a library issue asking if the form of nothrow operator new should be:

new (nothrow) T

or:

new (nothrow()) T

Dawes said yes. Myers said it was too late to change this.

Dawes proposed to resolve issue 18-014 by accepting the recommendation from the issues list (N0784R2 = 95-0174R2). The recommendation specifies a corresponding delete for the nothrow operator new. Dawes explained that we need this to account for an exception thrown during a nothrow new expression.

(See Motion 17 for the formal wording of the previous proposals.)

Dawes proposed accepting the recommendation for issue 20-018 (from N0789R2 = 95-0189R2). The proposal specifies the semantics of auto_ptr<>.reset().

Dawes then presented the WG's proposal to resolve issue 20-021 (from N0789R2 = 95-0189R2) by specifying a default constructor for pair<>. The paper offered alternative recommendations; the WG proposed the latter after cleaning it up a little. They intended the specification to conform to HP's implementation of the STL.

Dawes said the WG recommended closing the following issues (from N0789R2 = 95-0189R2) with no action:
-- 20-014: add an allocator template
-- 20-017: add an implicit_cast template
-- 20-019: add default constructors to many library classes
-- 20-020: remove make_pair
-- 20-022: add unary_compose and binary_compose

Corfield asked if this means we will keep make_pair, and not add unary_compose and binary_compose. Dawes said yes.

(See Motion 18 for the formal wording of the previous proposals.)

Dawes said the WG agreed to accept the recommendation for the following clause 25 [algorithms] issues (from N0793 = 95-0193):
-- 25-001: change the behavior of find_end so it can be implemented
-- 25-002: change the behavior of find_first_of so it can be implemented
-- 25-003: change the behavior of adjacent_find, min_element, and max_element so they can be implemented
-- 25-005: remove an extraneous footnote from clause 25.1.9
-- 25-007: add a constraint against overlapping ranges for copying algorithms
-- 25-009: revise effects of partial_sort to say remaining elements are in unspecified (not undefined) order
-- 25-010: describe effects of set_difference more precisely
-- 25-011: describe effects of set_symmetric_difference more precisely
They also recommended closing the following issues with no action:
-- 25-006: state explicitly that 'copy' requires forward copying
-- 25-008: change return type for stable_partition
(See Motion 19.)

==== Myers ====

Myers summarized the WG's proposals regarding various clause 22 [locale] issues (from N0788R1 = 95-0188R1). The first proposal addressed issue 22-009. The WG recommended divorcing the global C locale from the global C++ locale. That is, calls to setlocale() should not affect the C++ library functions, but calls to locale::global() _may_ affect the C locale. (See Motion 26, which includes other recommendations.) This allows the C++ locale machinery to be layered on top of the C locale machinery.

Plum asked how the C++ locale machinery affects the C locale. Plauger explained that using only named locales in C++ will keep the C locale in step; however, if you use other features of the C++ locale, all bets are off.

Myers elaborated the second locale proposal, which cleans up facet put and get error handling (as per issues 22-017, 22-044, 22-063, 22-064, 22-065 from N0788R1 = 95-0188R1, and per N0791 = 95-0191):
-- clean up streambuf iterators
-- add failed() to ostreambuf_iterator
-- cleanup argument lists to provide communication of errors via an ios_base::iostate& argument.
(See Motions 20 through 23.)

Myers presented details of the third proposal, to extend the codecvt facet to support filebuf extensions (as per issues 22-042 and 22-043):
-- define length() analogous to mblen()
-- define max_length() analogous to MB_CUR_MAX
-- add do_always_noconv() to indicate vacuity (to do with run-time optimizations)
-- clarify the meaning of do_convert "partial" result
(See Motion 26, which includes other recommendations.)

Myers then presented a few more specific proposals regarding locales, monetary representation, and input/output. (See Motions 24, 25, 27, 28 and 29.)

Straw vote: Who agrees with these recommendations? lots yes, 0 no.

==== Plauger ====

Plauger presented a proposal to remove the caching semantics from use_facet. He explained that the changes allow streams to be imbued with transparent locales in a way that the global locale still "shows through" so that parts of the locale can be modified independently. (See Motion 30.)

Dawes asked how strongly the Library subgroup supported this. Plum said it was 9 for and 2 against. Myers said he recalled that the support was for Plauger to write the proposal, not the proposal itself. Clamage said the vote was to assess support for the principle in order to see whether it was worth writing a proposal.

Plum said this proposal matches the recommendation made in Plauger's issue list which has been available for some time. Myers said he objected to the proposal because it removes protection against some problems. He also complained that it is a sweeping change to the architecture of locale that breaks an invariant. He said he has an alternative proposal to remove the caching behavior without breaking this invariant.

Clamage asked whether the WG should take the issue back. Plauger felt that the straw vote indicated that it's ready for a committee vote. Plum agreed. Dawes said that the proposal includes a lot of new wording that we haven't seen. Corfield said we rarely see detailed wording until meetings; if the issue has been on an issues list for a long time, and if the WG's straw vote favored the proposal, then we should move

forward. Welch and Stroustrup also spoke in favor of moving to a vote.

Myers tried to invoke the two-week rule (X3's rule that a committee member may object to voting on an issue not presented to members at least two weeks before the meeting). Plum said he would rephrase the proposal in terms of a paper that satisfies the two-week rule. A somewhat tart exchange ensued. Plauger said his recommendation has been on record since March '95.

Straw Vote: Should we proceed on this issue: lots yes, 1 no.

Straw Vote: Who favors this proposal: 10 yes, 1 no, 11 abstain.

Welch requested separate WG21 and X3J16 votes.

Straw Vote: Who favors this proposal:
    WG21: 2 yes, 0 no, 3 abstain.
    X3J16: 9 yes, 1 mo, 12 abstain.

Stroustrup asked the abstaining votes to think carefully about the issue before the formal vote. Plum asked if we are going to open a debate in full committee. Harbison said the large number of abstentions indicates that people just want to read the proposal. Clamage suggested moving forward with a vote on this issue on Friday.

Myers wanted to continue debating the issue, preferably on the reflector. Koenig said we have a paper on the table, we should read it and vote. Bruck said he abstained because he has not had a careful look at the issue until just now.

==== Koenig ====

Koenig presented a proposal for improving the description of input iterators. He noted that this will probably cause editorial ripples through parts of the descriptions of other iterators. (See Motion 31.)

Stroustrup asked if anyone who had studied the issue objected. Koenig said all four participants in the discussion now agree. Clamage pointed out that these four did not agree initially, so this represents a step forward.

Myers recommended a "bold change" for the other iterators. Dawes suggested that these other changes be made sooner rather than later. Unruh asked which model the proposal represents. Koenig explained that, under this proposal, ++ invalidates all copies of the iterator.

==== Wilhelm ====

Wilhelm presented a proposal to resolve various basic_string issues (as per the recommendations in N0800R1 = 95-0200R1). He provided details on some of the issues. (See Motion 32 for the full list of issues.)

Wilhelm explained that the recommendations for the following issues clean up string_char_traits: 002, 018, 024, 030, 060, and 067. Stroustrup asked if this unifies the string and stream char_traits. Wilhelm said the unification is a separate issue that needs further discussion.

Unruh asked whether eos can return different values on different calls. Wilhelm replied that this behavior is not specifically prohibited. Unruh asked that this be added to an issue list. Wilhelm agreed.

Plauger asked whether the additional functions in string_char_traits provide the memchr and fill functionality. Wilhelm said yes.

Wilhelm explained that the issue 077 addresses the problem that the

second argument to the append member is sometimes a position and other times a length. He gave this example:

```
string s = "123";
s.append(string("abc"), 2);     // 2 is a position, s = "123c"
s.append("abc", 2);             // 2 is a length, s = "123ab"
```

The WG recommended making the first call an error by requiring both the position and length arguments in the call, as in:

```
s.append(string("abc"), 2, 1);  // s = "123c"
```

Wilhelm explained that the WG considered it better to make the previous use an error instead of silently changing the meaning of the call. Gibbons asked if the WG considered switching the order of the length and position arguments. Wilhelm said the WG decided that would confuse people even more.

Wilhelm listed the remaining open issues that were not addressed at this meeting:
-- 014: fix argument order for copy out to charT
-- 059: unify of string_ and stream_ char_traits
-- 062: add requirements on charT
-- 080: allow template specializations for basic_string and string_char_traits
-- 081: remove redundant descriptions

Straw vote: Who agrees with these recommendations? lots yes, 0 no.

==== Podmolik ====

Podmolik presented a proposal to resolve various container issues (as per the recommendations in N0781R2 = 95-0181R2). He listed the specific issues:

-- 23-010: requirements for type T closed with no action
-- 23-024: fix copy constructors with respect to allocators by removing the signature that includes the allocator
-- 23-030: update descriptions of deque operations
-- 23-031: specialize 'swap' for containers

Plauger asked whether the specializations are packed with the containers or with 'swap'. Podmolik said they're with the containers.

-- 23-032: affirm that priority_queue does not require a non-const top()
-- 23-033: clean up resize() effects for deque, list and vector
-- 23-034: reverse iterator types for list by changing the name of the adaptor to reflect that list has a bidirectional iterator
-- 23-035: correct argument list to vector<bool>::insert
-- 23-036: add semantics for at() for deque/vector
-- 23-037: add semantics for a.back() in sequence requirements
-- 23-038: specify iterator properties for clauses 21 and 23 -- they provide bidirectional iterators
-- 23-039: specify that erase(iterator) returns an iterator pointing just past the erased element(s)
-- 23-040: add typedefs for map and multimap T type

Koenig observed that map and multimap store a {key, value} pair, and there's no convenient typedef name for the value part. Podmolik said the WG did not feel strongly about any particular name.

Myers asked something about the get_allocator() member. Podmolik said this is an editorial correction to add a line missing from the table.

Myers pointed out the priority_queue template has an incorrect default

container type. (Apparently, this made it into Motion 33.)

Podmolik said two issues remain open:
-- remove empty sections in the WP
-- fix problems with insert() argument lists

Straw vote: Who agrees with these recommendations? lots yes, 0 no.

6.3 ANSI C compatibility WG

Plum presented a proposal to allow extended identifiers and literals in C++ source programs (N0808 = 95-0208). The proposal introduces a notation somewhat similar to trigraphs for representing characters from ISO 10646 sets with encodings larger than eight bits. For example, ??u05D0 represents the 16-bit character whose code is x05D0.

Plum explained numerous details of the proposal. Clamage asked whether this proposal was in the mailing. Plum said no. The committee discussed more technical issues. A few members expressed uneasiness about voting on the proposal. Bruck requested the following straw vote.

Straw Vote: Who supports extended characters in identifiers and the intent of the proposal? lots yes, 1 no.

Koshida asked that we not hold a formal vote on this.

6.1 Core (revisited)

==== Gibbons ====

Gibbons explained that a proposal clarifying "point of instantiation" was accidentally omitted from the formal motions at the previous meeting. Inasmuch as the committee had accepted the proposal by straw vote, Gibbons and others managed to get the proposal into the WP along with an editorial box (number 56) that explained what happened. Gibbons asked that we simply ratify this change (and then we can remove the editorial box).

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

(See Motion 14.)

7 WG sessions (if any time is left)

8 Distribution of formal motions

The committee discussed Motion 15 (concerning library header inclusion). Koenig said he would not move the motion because of problems found during the drafting session. He said he had not yet found a working alternative.

Plauger explained that the WP spells out which headers a header may include. Unfortunately, it’s impossible to implement the headers as specified. Until recently it didn’t matter, because those parts of the WP weren’t normative. However, a recent editorial change made them normative. Therefore Plauger was concerned that, without this motion, it would be impossible to comply with the draft in its present form.

9 General session II

9.1 Core Language WG

Gibbons summarized the recent work of his Core subgroup. He said the group agreed with N0779 = 95-0179 in recommending editorial changes regarding exception handling, and they will forward them to the editor. He said they were working on the template compilation model but had no

firm recommendation(s).

Gibbons said the subgroup has been looking at friend name injection. The group rejected Gibbons' original suggestion, and were working on an alternative similar to the operator name lookup rules.

Stroustrup said we must try to preserve certain well-published programming techniques, but at the same time address all the horrible problems that general injection causes. Gibbons said that, in order to keep injection but tame it, we might have to introduce a reconsideration rule that covers the entire translation unit. So the group is trying to find an alternative. Laughter. Plauger asked what our objection was to compiling everything twice. More laughter.

Gibbons said he and Kiefer discussed a canonical name format for the string returned by type_info::name. This needs further discussion within the subgroup. Plauger suggested that this might be a good subject for a Technical Report. He explained that TRs have to go through an approval process, but they are a good approach for difficult issues where we need guidance.

Lajoie discussed ongoing work in the other Core subgroups. She said the "Schwarz" optimization as currently worded is too widely applicable -- allowing copied objects to be omitted in too many situations -- and it breaks certain resource acquisition techniques. Koenig said a discussion on the reflector suggested that the optimization should be allowed for local automatics and named variables. Murphy was concerned that this would cause portability problems across platforms.

9.2 Library WG

Dawes outlined the Library WG plans. They intend to work on the detailed issues arising from NB comments and issues lists.

9.3 ANSI C compatibility WG

Plum said the C Compatibility WG will continue to work on the issue of extended identifiers since this received a favorable response from the committee. Plum asked interested parties to contact Koenig if they want to be added to the -compat reflector. Corfield explained that, although the committee's straw vote showed support for the concept, we agreed not to take a formal vote on extended identifiers at this meeting.

9.4 Edit WG and other general session business

Clamage asked Koenig if there would be an editing session following the meeting. Koenig said that there would not be an organized session, but anyone who wants to work on the WP is welcome to do so. Koenig asked that each motion have an "owner" who can be contacted about the exact intent of the wording. Plum said that the proposer should be considered as the owner. Clamage agreed, and announced that the proposer will be treated as the owner for issues moved at this meeting.

Clamage announced that he has produced a PDF version of the WP with bookmarks.

Clamage said a few people had expressed concern over some of the motions so he would discuss them issues offline today. Corfield asked whether we should reconvene at the end of the day to hear the results of any such discussions. Clamage agreed to reconvene the committee at 16:00.

The committee recessed into working groups at 9:45 and reconvened at 16:05.

Clamage said we are very close to quorum on X3J16 and asked everyone to make sure they are present for formal votes. Plum explained that the X3J16 votes are to establish a position for the US WG21 representative,

so a quorum is not strictly necessary unless a specific X3J16 business issue arises. Plauger said the danger is that, if X3J16 did not have a quorum, it would open the committee to criticism in terms of how representative the decisions were.

Clamage took a count of X3J16 voting members: 22. This is a quorum... just.

Clamage asked if any subgroups wanted to change any motions coming up for a vote.

Podmolik explained a change to motion 33 which adds the resolution of a new issue, 23-042, added at this meeting. The issue fixes an editorial error in the default argument of the priority_queue adaptor template.

Myers said the Library subgroup had decided a whole bunch of iostream issues which will come up for a vote at the next meeting.

The committee discussed Motion 30. During the day, the Library WG voted to defer this motion. Clamage called for a another straw vote of the whole committee on Motion 30. Unruh asked for a three-way vote, with the option to defer the issue to a later meeting. Harbison said that if the motion should fail tomorrow, more work will be done and some variant of the proposal will return at a future meeting. He urged a simple yes/no vote.

Straw vote:

> Who would vote YES on the motion tomorrow? 5
> Who would vote NO on the motion tomorrow? 0
> Who wants to defer the vote to a future meeting? lots

Lajoie said her Core subgroup discussed the ’long long’ proposal from the C committee. The subgroup agreed to ask WG14+X3J11 to resolve the issue, and present the solution in the form of a technical report. She wanted to know if WG21+X3J16 agreed with this position.

Gibbons said we could take either of two approaches. We could say to the C committee:
1) please address the issue
2) we want longer integers; please add them
He said it’s important to phrase this properly. Lajoie said the subgroup decided on (2). Gibbons wanted to ensure that the committee had the option to decide which approach they wanted.

Stroustrup felt that asking the C committee to add an extension was more serious than simply adding it ourselves.

Plum related the C committee’s approach to this issue. They have explored two directions:
1. a practical, market-oriented approach to add the type ’long long’
2. a new way of describing integer types wherein the program specifies the number of bits required, whether a larger integer is acceptable, and so on

Plum said he has argued that replacing the current simple numeric type system with a very fine granularity would be very bad for a language like C++ with overloading. He argued that for compatibility reasons, C should pursue the ’long long’ solution.

Stroustrup supported Plum’s position on this. However, he still felt that this looks like "extension by the back door" and he was reluctant to support another extension. Koenig thought that whatever the C committee decided would come too late for this C++ standard. Stroustrup said we that if we send a strong message to the C committee, we will be obliged to adopt C’s solution -- whatever it is. Welch said users are concerned that ’long long’ is not a long-term solution.

Straw Vote: Who wants to...
discourage 'long long'? 7
monitor the 'long long' proposal? 6
encourage 'long long'? 12

Plum asked if we should discuss this on the reflector. He also asked the 'discourage' votes to explain their reasons.

Schreiber said 'long long' is too restrictive; it provides only one more type to solve this one problem. He preferred a general solution more like FORTRAN's width specifiers. Gibbons said it won't be long before we need more than 64 bits, but the question should be "do we need longer integers for pointer arithmetic". Either way, 'long long' is a stop-gap solution.

Stroustrup said he has had requests for longer doubles, longer pointers and longer characters. We have not looked at how it would impact the conversion and overloading rules, so pushing another committee to add this is dangerous.

Plum asked if anyone strongly opposed 'long long'. Koenig said that if C adopts 'long long', C++ compilers will add it anyway. The straw vote showed four people were strongly opposed. Stroustrup said he was very unhappy that, having decided not to add extensions ourselves, we are considering encouraging someone else to do it.

Plum said SC22 has expressed the position several times that C++ should stay reasonably compatible with C. Koenig said that there will be other features that C9x will adopt and those will also have an impact on us. Plum said we adopted by reference parts of the C library. Koenig said we agreed not to change the C subset of our library without a vote even if C changed.

Harbison announced he will discuss the mid-Ballot procedures (to cover the Stockholm meeting) on Friday. He also asked NB representatives to consider the progress made on their issues in time for the post-meeting mailing.

10 WG sessions (if any time is left)

The committee recessed at 17:00 and reconvened Friday at 9:00.

11 Review of the meeting

Clamage called roll because there appeared to be many members missing. He confirmed that all 22 voting members were present.

Clamage said the working groups have made good progress and the meeting has gone smoothly.

Plum announced that he has assumed ownership of the locale issues list. Myers thanked him.

Clamage noted that the proposer of each motion will be recorded as the owner of that motion. X3J16 votes will be taken first followed by WG21 votes. The abbreviated form of X3J16 voting will be used -- only votes against will be counted unless otherwise requested.

11.1 Formal motions

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

Move we accept N0785 = 95-0185 as the current WP.

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

==== presented by Adamczyk ====

2) Motion (to reinstate operator void()) by Lajoie/Adamczyk:

Move we amend the WP as follows:

Delete the last sentence of 12.3.2 [class.conv.fct], paragraph 1, and replace the period at the end of the preceding sentence with:

, or to (possibly cv-qualified) void.

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

3) Motion (to define semantics of call-by-value argument initialization) by Adamczyk/Lajoie:

Move we amend the WP as described in N0780 = 95-0180.

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

4) Motion (to adjust the definition of scalar and fundamental type) by Gibbons/Unruh:

Move we amend the WP as described in N0774 = 95-0174.

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

5) Motion (to clarify uses of function typedefs for member functions) by Schreiber/Adamczyk:

Move we amend the WP as described in N0813 = 95-0213.

Corfield asked if operators can be declared using function typedefs. No one seemed sure. Lajoie agreed to take this as a Core issue.

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

6) Motion (to allow some dynamic initializations to be done statically) by Lajoie/Rumsby:

Move we amend the WP as follows:

-- add to 3.6.2 [basic.start.init], paragraph 1, at the end of the sentence "Objects with static storage duration initialized with constant expressions...":

; see also _dcl.init.aggr_ for additional initialization cases for which initialization must be done statically.

-- add to 3.6.2 [basic.start.init], after paragraph 1, as a new paragraph:

An implementation is permitted to perform an initialization of a nonlocal object with static storage duration as a static initialization even when such initialization is not required to be done statically, provided that (a) the dynamic version of the initialization would not cause the value of any other nonlocal object with static storage duration to change prior to their initialization, and (b) the static version of the initialization produces the same value in the initialized object as would be

produced if all initializations not required to be done statically were done dynamically. [Note: as a consequence, it is unspecified whether a reference, in an initialization, to a variable potentially requiring dynamic initialization, and defined later in that compilation unit, will obtain the value fully initialized or merely zero-initialized. Example:

```
struct A {
    float x;
    A(float p) : x(p) {}
};
extern A b;
A a(b.x);   // unspecified forward reference to b.x
A b(1.0);
```

-- end note]

-- add to 8.5.1 [dcl.init.aggr], at the end of paragraph 13:

If all of the member initializer expressions are constant expressions, and the aggregate is a POD type, the initialization is done during the static phase of initialization.

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

==== presented by Lajoie ====

7) Motion (to resolve start and termination issues) by Lajoie/Adamczyk:

Move we amend the WP as follows:

-- adopt the proposed resolutions for issue 429 from N0802 = 95-0202.

-- add to 3.6.1 [basic.start.main], paragraph 4:

If 'exit' is called to end a program during the destruction of an object with static storage duration, the program has undefined behavior.

-- replace 3.6.1 [basic.start.main], paragraph 1, first sentence with:

A C++ program shall contain a global function called main, which is the designated start of the program. It is implementation-defined whether a program in a freestanding environment is required to define a main function. [Note: in a freestanding environment start-up and termination is implementation-defined; start-up contains the execution of constructors for nonlocal objects with static storage duration; termination contains the execution of destructors for objects with static storage duration.]

-- replace 3.6.3 [basic.start.term] paragraph 1, sentences 2-4 with:

Objects with static storage duration (declared at block scope or at namespace scope) are destroyed in the reverse order of the completion of the execution of their constructors. For an object of array or class type, all subobjects of that object are destroyed before any local object with static storage duration initialized as a side-effect of constructing the subobjects is destroyed.

If a function contains a local object of static storage duration

that has been destroyed and the function is called during the destruction of an object with static storage duration, the program has undefined behavior if the flow of control would pass through the definition of the previously destroyed object.

If a function is registered with atexit then following the call to exit, any objects with static storage duration initialized prior to the registration of that function will not be destroyed as part of the termination process until execution of the registered function has completed.

If an object with static storage duration is constructed then following the call to exit, any functions registered with atexit prior to completion of the construction of that object will not be called as part of the termination process until execution of that object's destructor has completed.

-- adopt the proposed resolution from N0772 = 95-0172.

Gibbons asked if the subgroup had considered the impact of the fourth bullet of the motion on a multi-threaded environment. Lajoie said no. Gibbons said the resolution would require process locks on initialization. Welch said an implementation must do this already for local statics and he believed there were optimizations for the global case. Plum said multi-threading was beyond the scope of the standard and we should discuss it separately.

Gibbons said historically the static initialization process was a large fraction of the startup cost. This resolution would increase it, possibly to where it may be unacceptable for some applications. Plum said he agreed with Gibbons' concerns, but maybe the way to resolve this is to say that these words do not apply to multi-threading. Bruck did not want to block the proposal, yet he didn't want to make difficulties for implementations in multi-threading environment.

Gibbons asked that an editorial box be added to indicate there was contention regarding multi-threading. Coha asked to vote on the fourth bulleted item separately.

Motion to amend by Coha/Gibbons:

Move we remove the fourth bulleted item from the motion.

Spicer said the WP already contains words on initialization which imply whatever restrictions on multi-threading Gibbons is concerned about. Plum spoke against the amendment on the grounds that we might be undermining the consistency of the subgroup's work. Lajoie said it would be fine to split the motion -- the other bullets remain consistent. Bruck said he was in favor of the bullets 1, 2, 3 and 5 of the motion but had a problem with bullet 4.

Motion to amend passed X3J16: lots yes, 4 no, 0 abstain.
Motion to amend passed WG21: 5 yes, 0 no, 1 abstain.

Motion (as amended) passed X3J16: lots yes, 0 no.
Motion (as amended) passed WG21: 6 yes, 0 no, 0 abstain.

7b) Motion (to specify order of destruction for objects with static storage duration) by Lajoie/Welch:

Move we amend the WP as specified in the original fourth bulleted item of Motion 7.

Welch noted that his implementation handles multi-threading and shared libraries that are dynamically loaded and unloaded without noticeable performance overheads. Myers asked if Watcom implemented locking during

initialization. Welch said yes.

Welch agreed to write a paper for the reflector on how Watcom implements locking during initialization.

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

8) Motion (to resolve memory model issues) by Lajoie/Adamczyk:

    Move we amend the WP as follows:

    -- adopt the proposed resolution for issue 554 from N0803 = 95-0203.

    -- change 3.9 [basic.types], paragraph 2 both occurrences from:

        memcpy library function

        to:

        memcpy or memmove library functions

    xx add to 5.2.1 [expr.sub], 5.2.5 [expr.post.incr], 5.3.2 [expr.pre.incr]:

        If the operand is a pointer to T and the object pointed to by the pointer is not an object of type T, the program has undefined behavior.

    xx add to 5.3.1 [expr.unary.op] paragraph 1:

        If the operand is a pointer to T and the object pointed to by the pointer is not an object of type T or an object of a type derived from T, the program has undefined behavior.

    ++ add to 5.9 [expr.rel] and 5.10 [expr.eq]:

        If the operands are of type pointer to T and the pointers do not refer to objects of type T or to objects of a type derived from T, the program has undefined behavior.

    xx add to 5.17 [expr.ass]:

        In a compound assignment (+=, -=), if the left operand is a pointer to T and the object pointed to is not an object of type T, the program has undefined behavior.

    -- add to 5.7, after paragraph 6, the following new paragraph:

        If the value 0 is added to or substracted from a pointer value, the result compares equal to the original pointer value. If two pointers point to the same object or function or both point one past the end of the same array or both are null, and the two pointers are subtracted, the result compares equal to the value 0 converted to the type ptrdiff_t.

    xx change 5.9 [expr.rel], paragraph 2, last bullet to:

        -- Other pointer comparisons are unspecified.

Lajoie wanted to amend the motion to remove the changes to 5.2.1, 5.3.1, and 5.9 since it would be more appropriate to make these changes to the lvalue/rvalue section. She said she'd add a Core issue regarding the changes to lvalues and rvalues. Koenig asked Lajoie if she should remove the proposed changes to 5.2.5 and 5.3.2 as well as 5.2.1. Lajoie

said yes.

Gibbons said the lvalue/rvalue conversion issue was important. Unruh suggested also removing the changes to 5.17, because ++p is is defined to be the same as p += 1. Lajoie agreed to remove them as well.

[Note:
Apparently, there was a procedural lapse here; I have no record of a motion to amend. However, it seems that no one complained that they did not know what they were voting on. I have used different "bullets" in the motion above to indicate what I understand to be the accumulated effect of this free-for-all:

    -- means keep the bullet item this as is
    ++ means delete the reference to clause 5.9 from this bullet item
    xx means delete this entire bullet item
:DS]

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

9) Motion (to resolve object model issues) by Lajoie/Rumsby:

    Move we amend the WP as follows:

    -- change 9.2 [class.mem], paragraph 11 to:

        The order of allocation of nonstatic data members separated by an access-specifier is unspecified.

    -- change the first two sentences of 9 [class], paragraph 3 to:

        A class with an empty sequence of members and base class objects is an empty class. Complete objects and member subobjects of an empty class type shall have a nonzero size.

    -- add to 10 [derived], after paragraph 3:

        A base class subobject can be of zero size; however, two base class subobjects of the same class type that belong to the same complete object shall not be allocated at the same address.

    -- add to 10.3 [class.virtual], after paragraph 5:

        The return type of the overriding virtual function shall not be an incomplete class type if it differs from the return type of the overridden function.

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

10) Motion (to resolve special member function issues) by Lajoie/ Corfield:

    Move we amend the WP as follows:

    -- delete 12 [special], paragraph 1, sentence 2.

    -- add to 12 [special], at the beginning of paragraph 1:

        The default constructor, copy constructor, copy assignment operator and destructor are special member functions. The implementation will implicitly declare these member functions for a class type when the program does not explicitly declare them except as noted in _class.ctor_. Programs may explicitly refer to implicitly declared special member functions.

[Example:

```
struct A { };              // implicitly-declared A::operator=
struct B : A {
    B& operator=(const B &);
};
B& B::operator=(const B& s) {
    this->A::operator=(s);  // well-formed
}
```

-- end example]

-- change 12.4 [class.dtor], paragraph 1, first sentence to:

A destructor of a class is declared with a ~ followed by the class name followed by an empty parameter list.

-- add to 12.4 [class.dtor], before paragraph 12:

In an explicit destructor call, the destructor name appears as a ~ followed by a type-name that names the class type. The object expression in an explicit destructor call shall be of the same class type as the destructor's class type or shall be of a class type derived from the destructor's class type. [Example:

```
struct B {
    virtual ~B() { }
};
struct D : B {
    ~D() { }
};
D D_object;
typedef B B_alias;
B* B_ptr = &D_object;
D_object.B::~B();   // calls B's destructor
B_ptr->~B();        // calls D's destructor
B_ptr->~B_alias();  // calls D's destructor
```

-- end example]

-- change 12.8 [class.copy], paragraph 2, first sentence to:

A constructor for class X is a copy constructor if its first parameter is of type X&, const X&, volatile X& or const volatile X&, and either there are no other parameters or else all other parameters have default arguments (_dcl.fct.default_).

-- change 12.8 [class.copy], paragraph 5 to:

The implicitly-declared copy constructor for a class X will have the form

X::X(const X&)

if

-- each direct or virtual base class B of X has a copy constructor whose first parameter is of type const B& or const volatile B&, and
-- for all the nonstatic data members of X that are of a class type M (or array thereof), each such class type has a copy constructor whose first parameter is of type const M& or const volatile M&.

Otherwise, the implicitly declared copy constructor will have the form

X::X(X&)

-- change 12.8 [class.copy], paragraph 9, first sentence to:

A user-declared copy assignment operator X::operator= is a non-static member function of class X with exactly one parameter of type X, X&, const X&, volatile X& or const volatile X&.

-- change 12.8 [class.copy], paragraph 10 to:

If a class definition does not explicitly declare a copy assignment operator, one is declared implicitly. The implicitly-declared copy assignment operator for a class X will have the form

X& X::operator=(const X&)

if

-- each direct base class B of X has a copy assignment operator whose parameter is of type const B& or const volatile B& and
-- for all the nonstatic data members of X that are of class type M (or array thereof), each such class type has a copy assignment operator of type const M& or const volatile M&.

Otherwise, the implicitly declared copy constructor will have the form

X& X::operator=(X&)

-- replace 12.6.2 [class.base.init], paragraph 2, last sentence with:

If a ctor-initializer specifies more than one mem-initializer for the same member, same base or for multiple members of the same union, the ctor-initializer is ill-formed. A ctor-initializer can initialize a member of an anonymous union defined in the constructor's class member list.

-- change 12.6.2 [class.base.init], paragraph 4, last sentence to:

If a class X has a nonstatic data member that is of reference type or of a const type that is not eligible for default initialization (_dcl.init_) and there is a constructor for class X which does not specify that member in its mem-initializer, the program is ill-formed.

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

==== presented by Gibbons ====

11) Motion (to accept proposed resolutions to various namespace issues) by Spicer/Bruck:

Move we amend the WP as described in 95-0212 = N0812.

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

12) Motion (to accept proposed resolutions to various typeid issues) by Adamczyk/Gibbons:

Move we amend the WP as follows:

-- replace 5.2.7 [expr.typeid] with the following:

The result of a typeid expression is an lvalue of type const std::type_info (_lib.type.info_), whose lifetime extends to the end of the program. Whether or not the destructor for the type_info object is called at the end of the program is unspecified.

When typeid is applied to a type-id, the result is a type_info object representing the type of the type-id. If that type is a reference type, the result is a type_info object for the referenced type. In both cases, if the type for which the type_info is returned is a class, that type shall be completely defined.

When typeid is applied to an lvalue expression whose type is a polymorphic class type, the result is a type_info object for the type of the complete object (_intro.object_) that contains the lvalue. If that expression is the result of applying unary * to a pointer [Footnote: If p is an expression with pointer type, then *p, (*p), (*(p)), and so on all meet this requirement.] and the pointer is a null pointer value (_conv.ptr_), the typeid expression throws the bad_typeid exception (_lib.bad.typeid_).

When typeid is applied to an expression other than an lvalue of a polymorphic class type, the result is a type_info object for the (static) type of the expression. The expression is not evaluated.

Lvalue-to-rvalue (_conv.lval_), array-to-pointer (_conv.array_), and function-to-pointer (_conv.func_) conversions are not applied to the expression. If the type for which the type_info is returned is a class type, that type shall be completely defined.

In all cases, typeid ignores the top-level cv-qualifiers of the type for which the type_info object is returned. [Example:

```
class D { ... };
D d1;
const D d2;

typeid(d1) == typeid(d2);       // yields true
typeid(D)  == typeid(const D);  // yields true
typeid(D)  == typeid(d2);       // yields true
typeid(D)  == typeid(const D&); // yields true
```

-- end example] [Note: Clause _class.cdtor_ describes the behavior of typeid applied to an object under construction or destruction.]

If <typeinfo> (_lib.type.info_) has not been included prior to a use of typeid, typeid returns an lvalue of type const std::type_info, but the class std::type_info is considered incompletely defined and is not visible by name at that point.

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

13) Motion (to accept proposed modifications to the rules for casting pointers to members) by Gibbons/Lajoie:

Move we amend the WP as follows:

-- change 5.2.8 [expr.static.cast], paragraph 9 from:

...can be converted to...where B is a base class of D...

to:

...can be converted to...where B is a nonvirtual base class of D...

-- change 5.2.8 [expr.static.cast], paragraph 9 from:

If class B contains or inherits the original member, ... Otherwise ...

to:

If class B is neither a base nor derived class of the class containing the original member, the behavior of the cast is undefined.

-- add to 5.5 [expr.mptr.oper], after paragraph 3:

If the dynamic type of the object does not contain the member to which the pointer refers, the behavior is undefined.

-- add to 4.11 [conv.mem], after "or ambiguous (10.2)":

or virtual (_class.mi_)

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

14) Motion (to ratify the point of instantiation) by Unruh/Corfield:

Move we ratify the working paper changes referred to by editorial box 56 in 14.3.2 [temp.point] paragraph 1, and remove the editorial box.

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

==== presented by Dawes ====

15) Motion (to clarify library header inclusion):

Move we amend the WP as follows:

-- change 17.3.4.1 [lib.res.on.headers], paragraph 2 and footnote 138 to:

Except for names reserved to the implementation [lib.global.names], the C++ headers in Table 21 shall declare or define exactly those names required by their descriptions. [Footnote: It is an implementation convenience for headers to include each other. Despite the restriction, that convenience is available by having headers define everything needed for implementation in a namespace called, say, _Implementation, and then selectively making those names available through using-declarations.]

-- throughout the WP, in any Synopsis clause that claims a header includes other headers, delete the text making such a claim.

This motion was not moved. An editorial box will be added about this issue.

16) Motion (to resolve several library issues from clause 17 issues list) by Dawes/Rumsby:

Move we amend the WP as described in the proposed resolutions for issues 001 through 003 from N0801 = 95-0201 Version 2.

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

17) Motion (to resolve several library issues from clause 18 issues list) by Dawes/Rumsby:

Move we amend the WP as described in the proposed resolutions for issues 013 and 014 from N0784 = 95-0184 Version 3.

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

18) Motion (to resolve several library issues from clause 20 issues list) by Myers/Dawes:

Move we amend the WP as described in the proposed resolution for issue 018 (alternative 2) from N0789 = 95-0189 Revision 2, and close without taking any action issues 014, 017, 019, 020, and 022 from N0789 = 95-0189 Revision 2, and resolve issue 021 from N0789 = 95-0189 Revision 2 as follows:

-- add to 20.2.2 [lib.pairs], the constructor declaration:

pair();

-- add to 20.2.2 [lib.pairs] the following Effects:

Effects: it initializes its members as if impemented:

pair::pair() : first(T1()), second(T2()) { }

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

19) Motion (to resolve several library issues from clause 25 issues list) by Dawes/Myers:

Move we amend the WP as described in the proposed resolutions for issues 001, 003, 005, 007, 009, 010 and 011 from N0793 = 95-0193, and close without taking any action issues 006 and 008 from N0793 = 95-0193, and amend the WP as described in the proposed resolution issue 002 from N0793 = 95-0193 with the following change:

-- the complexity specification is:

At most (last1 - first1) * (last2 - first2) applications of the corresponding predicate.

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

==== presented by Myers and Clamage ====

20) Motion (to change the ios and locale parameters in facets) by Myers/Rumsby:

Move we amend the WP as follows:

-- adopt the proposed resolution for 22-044 from N0788R1 = 95-0188R1.

-- throughout 22 [lib.localization], remove the "const locale&" parameter from every facet member function that takes an

ios_type& [now ios_base&] argument.

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

21) Motion (to cleanup streambuf iterator semantics) by Myers/Rumsby:

Move we amend the WP as described in N0791 = 95-0191.

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

22) Motion (to simplify the error handling of the facet get members) by Myers/Rumsby:

Move we amend the WP as described in N0788R1 = 95-0188R1, issue 22-065.

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

23) Motion (to make error handling consistent for reading all numeric types in facets) by Myers/Dawes:

Move we amend the WP as described in N0788R1 = 95-0188R1, issue 22-063.

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

24) Motion (to make moneypunct<> and numpunct<> consistent) by Myers/Dawes:

Move we amend the WP as described in N0788R1 = 95-0188R1, issue 22-048, resolution 1.

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

25) Motion (to clarify the semantics of internal padding for all numeric formats) by Myers/Dawes:

Move we amend the WP as described in N0788R1 = 95-0188R1, issue 22-053, resolution 2.

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

26) Motion (to resolve various locale issues) by Myers/Dawes:

Move we amend the WP as follows:

-- adopt the proposed resolutions from N0788R1 = 95-0188R1 for the following locale issues:

22-009, 22-019, 22-022, 22-035, 22-042, 22-043, 22-045, 22-049, 22-059, 22-066.

-- add to 22.2.6.3.2 [lib.locale.moneypunct.virtuals], in the description of moneypunct<>::do_pos_format() and moneypunct<>::do_neg_format():

The base class implementation returns an object of type pattern initialized to { symbol, sign, value, none }.

Motion passed X3J16: lots yes, 0 no.

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

27) Motion (to clarify how monetary values are parsed by moneypunct<>) by Myers/Dawes:

Move we amend the WP as described in N0788R1 = 95-0188R1, issue 22-055, resolution 2.

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

28) Motion (to resolve various iostream issues) by Myers/Dawes:

Move we amend the WP as follows:

-- adopt the proposed resolutions from N0794 = 95-0194 for the following issues:

27-003, 27-202, 27-302.

-- change 27.4.1 [lib.stream.types], before paragraph 2:

typedef INT_T streamsize;

to:

typedef SZ_T streamsize;

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

29) Motion (to remove the exceptions from setstate() and clear()) by Myers/Rumsby:

Move we amend the WP to resolve issue 27-201 from N0794 = 95-0194) as follows:

-- throughout clause 27 [lib.input.output], delete the exception specifications from the member functions setstate and clear.

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

==== presented by Plauger ====

30) Motion (to remove the caching semantics for locales from use_facet):

Move we amend the WP as follows:

-- delete 22.1.1 [lib.locale] paragraphs 6-8.

-- change 22.1.1.5 [lib.locale.statics], the description of locale::transparent, change paragraph 3, Returns, and Notes from:

static locale transparent();

3 The continuously updated global locale.

Returns: A locale which implements semantics that vary dynamically as the global locale is changed.

Notes: The effect of imbuing this locale into an iostreams component is unspecified (_lib.ios.members_).

to:

static locale transparent();

3 An empty locale.

Returns: A locale that has no facets.

-- change 22.1.2 [lib.locale.global.templates], the description of use_facet, Effects, Notes, Returns, and Notes from:

template <class Facet>
const Facet& use_facet(const locale& loc);

1 Get a reference to a facet of a locale.

Effects: If the requested Facet is not present in loc, but is present in the current global locale, returns the global locale's instance of Facet. Because locale objects are immutable, subsequent calls to use_facet<Facet>(loc) return the same object, regardless of subsequent calls to setlocale or locale::global.

Notes: The only exception to this rule is for the locale returned by locale::transparent(); it always returns the facet found in the global locale at the time of each call.

Throws: bad_cast if (has_facet<Facet>(loc) || has_facet<Facet>(locale())) is false.

Returns: A reference to the requested facet.

Notes: The result is guaranteed by locale's value semantics to last as long as the value of loc.

to:

template <class Facet>
const Facet& use_facet(const locale& loc);

1 Get a reference to a facet of a locale.

Throws: bad_cast if (has_facet<Facet>(loc) || has_facet<Facet>(locale())) is false.

Returns: If the requested Facet is present in loc, returns a reference to that instance of Facet. Otherwise, if the requested Facet is present in the current global locale, returns a reference to the global locale's instance of Facet.

-- delete 22.1.2 [lib.locale.global.templates], in the description of has_facet, Returns, the sentence:

If use_facet<Facet>(loc) has already been called successfully, returns true.

-- delete 22.1.1.5 [lib.locale.statics], in the description of locale::global:

If loc is (a copy of) the value returned by locale::transparent(), throws runtime_error.

[This last change will not be moved if motion 26 does not pass.]

This motion was not moved.

==== presented by Koenig ====

31) Motion (to clarify the requirements on input iterators) by Koenig/Myers:

Move we amend the WP by replacing 24.1.1 [lib.input.iterators], table 3 and the following paragraph with:

Operation	Type	Semantics, pre/post conditions
X u(a);	X	Post: u is a copy of a Note: a destructor is assumed to be present and accessible.
u = a;	X&	Result: u Post: u is a copy of a
a==b	convertable to bool	== is an equivalence relation over its domain
a!=b	convertable to bool	bool(a==b) != bool(a!=b) over the domain of ==
*a	T	Pre: a is dereferenceable If a == b, and both a and b are in the domain of ==, then *a is equivalent to *b
a->m		Pre: (*a).m is well-defined. Equivalent to (*a).m
++r	X&	Pre: r is dereferenceable Result: r Post: r is dereferenceable or past the end Post: any copies of the previous value of r are no longer required either to be dereferenceable or to be in the domain of ==.
(void)r++		Equivalent to (void)++r
*r++	T	{ T tmp = *r; ++r; return tmp; }

The term "the domain of ==" is used in the ordinary mathematical sense to denote the set of values over which == is (required to be) defined. This set can change over time. Each algorithm places additional requirements on the domain of == for the iterator values it uses. These requirements can be inferred from the uses that algorithm makes of == and !=. [Example: the call find(a,b,x) is defined only if applying ++ zero or more times to a eventually yields a value i such that i==b or *i==x.]

Saying that an expression e1 that mentions a is equivalent to an expression e2 that mentions b means that every expression that does not mention b and mentions a only in the context of e1 has the same meaning if every occurrence of e1 is replaced by e2. Every expression that does not mention a and mentions b only in the context of e2 has the same meaning if every occurrence of e2 is replaced by e1.

If a is a copy of b and a or b is in the domain of == then a==b. However, a==b does not require that a is a copy of b.

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

==== presented by Wilhelm ====

32) Motion (to resolve issues with basic_string template) by Dawes/Kiefer:

Move we amend the WP as described in the recommended resolutions from N0800R1 = 95-0200R1 for the following issue numbers: 2, 13, 17,

18, 24, 25, 26, 27, 28, 29, 30, 31, 34, 34a, 37, 60, 61, 63, 67, 68, 74, 76, 77, 78, 79.

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

==== presented by Podmolik ====

33) Motion (to resolve issues with the containers library) by Dawes/Myers:

Move we amend the WP as follows

-- adopt the the recommended resolutions from N0781R2 = 95-00181R2 for the following issue numbers: 10, 24, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40.

-- change the declaration of priority_queue in 23.2.4.2 from:

```
template<class T, class Container = deque<T>,
    class Compare = less<Container::value_type>,
    class Allocator = allocator>
class priority_queue { ... };
```

to:

```
template<class T, class Container = vector<T>,
    class Compare = less<Container::value_type>,
    class Allocator = allocator>
class priority_queue { ... };
```

Myers explained that this simply changes 'deque' to 'vector' (on the first line of the declaration). This corrects an error that occurred while transcribing the STL specification to the WP.

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

Koenig asked a procedural question about voting rights for members representing companies that split into separate companies. Clamage asked Koenig to ask him offline.

34) Motion (to thank our host for their hospitality) by Harbison/ Clamage.

Motion passed by acclamation.

11.2 Review of action items, decisions made, and documents approved by the committee

Clamage reminded members that those who proposed motions are responsible for communicating with Koenig regarding wording changes to the WP.

Dawes said that each library clause owner is responsible for producing an updated version of that clause's issue list. Lajoie said the same applied to the Core subgoups.

11.3 Issues delayed until Friday

Harbison explained the procedures for mid-ballot meeting (Stockholm, July '96):
-- Work at the meeting will address issues that we expect to be raised in National Body comments on the ballot.
-- We will not take any official votes changing the Working Draft; however, we may take straw votes on changes. The secretary will record the results of any straw votes in the meeting minutes.

-- At the first meeting after the Summary of Voting becomes available, we will present any mid-ballot straw votes for ratification in light of the ballot comments.
-- The project editor may issue a mid-ballot WD that reflects the impact of straw votes, but he will be able to remove any changes that we don't ratify later.

Some members questioned whether the last point was appropriate. Plum and others said they felt it would be acceptable to SC22.

12 Plans for the future

Clamage said that future work will focus on resolving issues from the various WGs' issue lists.

Harbison said that, for the next meeting, we are scheduled to finish resolving CD Ballot comments and vote out the second and final CD.

Clamage said we should try to clear the issues lists between now and the next meeting. Harbison said the issues lists do not have to be completely clear in order to vote out the CD, but we certainly don't want to go to a third CD. Several people expressed agreement.

Lajoie said the Core issue lists do not have very many open issues remaining -- perhaps 50. The only major issue remaining is the template compilation model, and she thought we'd have a resolution for that in time for the next meeting.

Harbison asked NB representatives to communicate with their SC22-level representatives to establish what they require to vote YES on the second CD Ballot. He asked that NB representatives let him know within the next few weeks their NB's opinion of the progress made at this meeting on their issue lists.

12.1 Next meeting

No one from Borland was present to confirm the arrangements for the next meeting.

12.2 Mailings

Lajoie said the deadline for the post-Tokyo mailing is December 1.

WG21+X3J16 generally agreed that electronic distribution of the mailings is working well and should continue.

12.3 Following meetings

Harbison listed the dates and locations for the upcoming meetings:
-- 10-15 March 1996 in Santa Cruz, USA hosted by Borland
-- 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 (location and host to be determined)
-- 13-18 July 1997, somewhere in the UK, hosted by Programming Research
-- 9-14 November 1997 (location and host to be determined)

13 Adjournment

Clamage asked if there was any other business.

Spicer thanked Corfield for standing in as secretary and Crowfoot for taking backup notes. Applause.

Motion by Welch/Myers:

Move to adjourn.

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

The meeting adjourned at 10:50 on Friday.

Appendix A - Attendance

Name	Affiliation	Stat	M	Tu	W	Th	F
Koenig, Andrew	AT&T Bell Labs	A	V	V	V	V	V
Stroustrup, Bjarne	AT&T Bell Labs	A	A	A	A	A	A
Bruck, Dag	Dynasim AB	P	V	V	V	V	V
Adamczyk, Steve	Edison Design Group	P	V	V	V	V	V
Spicer, John	Edison Design Group	A	A	A	A	A	A
Jonsson, Fredrik	Ericsson	P	V	V	V	V	V
Umekawa, Ryuichi	Fujitsu	O	A	A	A	A	
Coha, Joseph	Hewlett-Packard	A	V	V	V	V	V
Murphy, Michael	IBM	A	A	A	A	A	A
Lajoie, Josee	IBM (Canada)	P	V	V	V	V	V
Kamimura, Tom	IBM (Japan)	O	A	A		A	A
Sawatani, Yuriko	IBM (Japan)	O				A	A
Andersson, Per	Ipso Object Software	P	V	V	V	V	V
Stuessel, Marc	IST GmBH	P	V	V	V	V	V
Schreiber, Ben	Microsoft	P	V	V	V	V	V
Adachi, Taka	Miwa Systems	O	A	A		A	
Nagao, Masahiko	NTT Data Comm Systems	O	A				
Corfield, Sean	Object Consult Services	A	A	A	A	A	A
Nakano, Kazutoshi	Oki Electric	O	A	A	A	A	A
Benito, John	Perennial	P	V	V	V	V	V
Plum, Tom	Plum Hall	P	V	V	V	V	V
Southworth, Mark	Programming Research	P	V	V	V	V	V
Myers, Nathan	Rogue Wave Software	P	V	V	V	V	V
Smithey, Randy	Rogue Wave Software	A	A	A	A	A	A
Wengler, Christian	SET Software Consulting	P	V	V	V	V	V
Kumagai, Norihiro	Sharp	O	A				
Kiefer, Konrad	Siemens AG	P	V	V	V	V	V
Unruh, Erwin	Siemens Nixdorf	A	V	V	V	V	V
Kung, Michael	Silicon Graphics	P	A		A	A	
Podmolik, Larry	STR	P	A	A	A	A	
Wilhelm, Richard	STR	A	A	A	A	A	
Clamage, Steve	Sun Microsystems	A	V	V	V	V	V
Feng, Yinsun	Taligent	A	A	A	A	A	
Gibbons, Bill	Taligent	P	V	V	V	V	V
Harbison, Sam	Tartan	P	V	V	V	V	V
Koshida, Ichiro	Tokyo Engr Univ	O	A	A		A	
Nakamura, Akira	Toshiba	O	A				
Yamada, Asahiro	Toshiba	O	A				
Rumsby, Steve	UK	A	A	A	A	A	A
Welch, Jim	Watcom	A	V	V	V	V	V
Crowfoot, Norm	Xerox	P	V	V	V	V	V
Dawes, Beman		P	V	V	V	V	V
Plauger, P. J.		O	A	A	A	A	A
Total Attendance			42	37	35	39	32
Total Votes			22	22	22	22	22

Stat (Membership Status): P = principal; A = alternate; O = observer
Under M - F: V = Voting; A = attending