Document Number: X3J16/91-0136
WG21/N0069
Date: December 3, 1991
Project: 738-D
Reply to: Dan Saks
dsaks@wittenberg.edu

X3J16 Meeting No. 7
WG21 Meeting No. 2
November 11-15, 1991

Harvey Hotel - Addison
14315 Midway Road
Dallas, TX 74244 USA

1 Opening activities

The meeting was convened at 9:05 am (CST) on Monday, November 11, 1991 by Lenkov, chair of X3J16. Clamage served as acting vice-chair, and Saks served as secretary.

Texas Instruments hosted the meeting.

1.1 Opening comments

1.2 Introductions

Lenkov circulated an attendance list. Clamage circulated the membership list and asked attendees to make any necessary corrections. The attendance list was circulated each day, and is Appendix A of these minutes.

1.3 Membership, voting rights, and procedures for the meeting

Lenkov explained that this is a joint meeting of X3J16 and WG21. He said that each X3J16 member organization and each WG21 technical expert gets one vote. Both X3J16 members and WG21 members may vote in informal votes. In formal votes, Lenkov counts only the X3J16 votes, and Carter (convener of WG21) counts only WG21 votes.

Lenkov later explained that only one person from each X3J16 member organization can vote. A voting organization cannot vote if this is first meeting attended by representatives of that organization.

1.4 Approval of the minutes for previous meeting

Saks submitted the minutes from the previous meeting (N0013 = 91-0092 and N0014 = 91-0093) for approval. He noted one correction: item 20.3 on page 32 of N0013 = 91-0092 should be numbered 18.2.

Motion by Plum/Saks:

"Move that we approve the minutes from the previous meeting."

Motion passed: lots yes, 0 no, 1 abstain.

1.5 Distribution of position papers, WG deliverables, and other documents not distributed before the meeting

1.6 Agenda review and approval

Lenkov submitted the proposed agenda (N0053 = 91-0120) for approval along with these changes:

-- reschedule the WG14 liaison report under 1.10 as item 12.1
-- add item 1.11 on email reflectors

Motion by Saks/Bruck:

"Move that we accept the proposed agenda with these additions."

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

1.7 Agenda review and approval

Lajoie asked permission to distribute committee documents to Canadian ISO members. Lenkov said that members may distribute documents for standardization purposes, but not for any other purposes. He cautioned that some committee documents are copyrighted, and we should be careful to avoid violating copyrights.

Lenkov noted that some members have distributed committee documents for purposes other than standardization, and he reminded members not to do it in the future.

1.8 Vice-chair position

Lenkov explained that Miller resigned as vice-chair because he changed employers. Speaking for the committee, Lenkov said Miller had done a good job and we appreciated his efforts. X3 has issued a call for volunteers to be vice-chair. Volunteers should submit their names to the X3 SMC (Secretariat Management Committee) by the end of Dec., 1991. SMC will select the vice-chair. Clamage volunteered to act as vice-chair for this meeting.

Plum had heard that SMC would prefer that only one name be submitted. Shopiro asked if anyone else will put their name forward. Lajoie announced that she would. Lenkov recommended that the committee consider the candidates during the week and vote on Friday under agenda item 12.2.

1.9 Conversion to type I

Lenkov reported that the motion (by Shopiro/Druker under item 16.1 of 90-0115) to recommend that SPARC convert X3J16 to a type I project passed in a letter ballot (38 yes, 0 no, 12 to 15 abstained.). Lenkov and Clamage will prepare a letter advising SPARC of the voting results. Plum noted that X3J16 must also write a new project proposal. Lenkov said he had already done it.

1.10 Liaison reports

Swan reported on the work of NCEG (the Numerical C Extensions Group, officially designated X3J11.1). NCEG intends to prepare a technical report recommending extensions to C for X3J11 to consider when the ANSI C standard comes up for review in 1994. Among the topics discussed at the last NCEG meeting were:

1. restricted pointers (akin to the *noalias* qualifier) that grant a pointer exclusive access to a region of memory,

2. floating point extensions,

3. complex arithmetic type using cartesian coordinates,

4. array syntax for operations on entire arrays,

5. variable-dimensioned arrays,

6. extended integer ranges (e.g., *long long int* with a minimum of 64 bits),

7. syntax for initializing specific array elements.

There are two proposals for variable-dimensioned arrays: one from Tom MacDonald of Cray and the other from Dennis Ritchie of AT&T Bell Labs. Ritchie's proposal requires that users explicitly *malloc* the array storage, while MacDonald's proposal is for the translator to do the *malloc* automatically.

Plum noted that SC22 had previously insisted that C++ not evolve in directions incompatible with C without good reason. At the plenary in Vienna, SC22 voted in other direction -- they don't want C to evolve incompatibly with C++ without good reason.

Lenkov explained that Plum has been confirmed by SMC as the official US IR (international representative) to WG21.

Carter presented the SC22 Liaison Report. SC22 took the following actions:

1. formally established WG21
2. confirmed Carter as convener of WG21
3. confirmed Shopiro as WG21 project editor
4. left the working arrangement between WG21 and X3J16 unchanged
5. asked all WGs to consider:

-- support for international character handling
-- support for language independent data types and arithmetic operations
-- portability annex

Carter listed the remaining work for SC22:
1. get ISO/IEC JTC1 to provide for interpretations, and
2. get ISO/IEC JTC1 to allow for meeting fees.

1.11 Email reflectors

Koenig reported that since Miller resigned as vice-chair, he (Koenig) has been besieged by requests to change membership on the reflectors. Koenig arranged to incorporate the working group member information into the master membership list. The vice-chair will be responsible for maintaining the lone membership database.

Carter requested that WG21 members use the *x3j16-intl* reflector for WG21 messages.

Koenig also reported that between 15 and 20 messages bounced each day over the past month. Koenig appealed to members to test their email addresses by sending a message to *x3j16-ping@redbone.att.com*.

Lenkov asked Koenig, Carter, Clamage and Plum to handle the problems of reflectors for WG21 members.

2 WG reports

2.1 Core Language WG

Koenig reported that the WG had made progress on resolving the name lookup issue at the Lund meeting. Since then, the discussion on the group's email reflector had had little to do with name lookup. He had new insights into the name lookup issue to present to the WG at this meeting that he hoped would resolve the issue. If the group finished with name lookup, they would work on the lifetime of temporaries.

Lenkov noted that at the last meeting, Ball said he'd write a paper on inline friend functions (page 20 of N0013 = 91-0092) and O'Riordan said he'd write paper(s) on operator name hiding and overloading resolution (items 6, 7, 8 on page 20 of N0013 = 91-0092). Lenkov wondered if these papers were ready. Neither Ball nor O'Riordan were present to respond.

Stroustrup said Koenig's paper on name lookup (previously broadcast as email message *x3j16-core-820*) offers real insights into resolving the problems and is worth reading.

2.2 Extensions WG

Stroustrup said he has previously wished that more work would go into the core language than into extensions, so he should be pleased that there hadn't been much activity on extensions.

At the Lund meeting, Stroustrup and Goldstein promised to write a set of guidelines for submitting proposals for extensions. Stroustrup said a first draft of that letter was available for review. Stroustrup also listed some of the open issues left over from the previous meeting:
-- overloading operator . (dot)
-- return type of virtuals (O'Riordan offered to write a paper on this, but it's not yet available)
-- *const* member initialization (Gibbons and Goldstein were to write a paper for this)
-- runtime type identification (91-0063)
-- name space pollution (91-0041)

Lenkov also noted that Plum and Gautron were supposed to have worked on AFNOR's proposal for 8-bit characters in identifiers (N0003 = 91-0070).

2.3 Library WG

Vilot said the group would work on the following items:
-- approach toward errors and predefined exceptions
-- proposal for section 18.1 (language support)
-- proposal for section 18.2 (input/output)
-- interaction between C and C++ libraries
-- proposal for section 18.3 (ANSI C library)
-- proposal for section 18.4 (strings)
-- rationale for the library
-- container classes

2.4 Environments WG

Chapin listed the open issues before the WG:
-- translation limits
-- order of static object initialization
-- one definition rule
-- mixed C and C++ implementations
-- preprocessing

2.5 Formal Syntax WG

Roskind reported that only one WG member was at the Lund meeting, so the WG had made little progress since March. He said the group would review several proposals, including Pennello's proposals on template delimiters (91-0033) and throw syntax (91-0034). Roskind also noted that Scian's proposal on *new* initializer syntax (N0046 = 91-0013) was accepted as editorial changes and incorporated into the latest version of the C++ draft.

2.6 C Compatibility WG

Plum reported that:
-- About 15% of the definitions from the C standard approved for incorporation in the C++ standard (90-0088) had actually been incorporated.
-- The chapter on the preprocessor had not yet been brought over from the C standard.

-- The specification of declarators is much better.
-- The "impressionistic" list (email message *x3j16-compat-38*) is the first draft of the documented incompatibilities requested by SC22. The list is called "impressionistic" because the WG can't determine whether they are indeed incompatibilities until the committee reconciles C++ terminology with C terminology. 16 incompatibilities are already listed in chapter 18 of the ARM. The impressionistic list contains 17 new items to consider.
-- The C Compatibility WG needs a new member to compile the "impressionistic" items for chapter 5.

Plum listed the 17 new "impressionistic" items (Secretary's Note: The following list includes revisions made during the later WG meetings):

Editorial?

3.1 What does it mean for an object to be "used"? An unused object need not be defined.
3.4 There's no guarantee on the type of *main* (the wording is weak).
3.3 A name at file scope can't be *static* and then *extern*.

Open?

3.6 "compatible type" and "incomplete type".
4.6 *(void *)0* is not a null pointer constant.

Justified?

2.2 // comments.
3.1 A "tentative definition" is a definition.
3.4 *main* can't be recursive, and can't be addressed.
3.6 *char ** and *void ** representations might differ.
4.6 Can't assign a *const ** or a *volatile ** to a *void **.
5.4 Converting *int* to *enum* is undefined unless the converted value is an enumerator.
6.6 Can't use *return* without a value in a non-*void* function.
7.1 Can't declare *static struct x {...}*.
7.1.6 *const* objects must be initialized in C++, except...
8.2 Can't define a type in a return type or an argument type.
9.9 A *typedef* in a class declaration can't be redefined.
16.10 definition of *__STDC__*.

3 WG sessions

Lenkov announced that Insinga was not at this meeting, so there would be no technical session on strings. Instead, Schwarz would present a technical session on predefined exceptions, and Lenkov would present a technical session on runtime type identification.

The committee recessed to working groups at 11:40 am Monday.

4 WG sessions

5 WG progress reports

The committee reconvened at 3:25 pm Tuesday.

Clamage announced that members wishing to join an email reflector should, for the time being, send email to Koenig stating their request.

6 WG sessions

The working groups summarized their progress and the work they hoped to accomplish Wednesday morning. In explaining the work of the Extensions group, Stroustrup asked for someone to champion the proposal to allow overloading of the dot operator (a heretofore unnumbered document by Adcock). In the absence of such support, the group will reject the proposal. Stroustrup also asked for someone not already aligned with one side or the other to provide additional analysis of 91-0067, the proposal to allow initializing *const* objects inside class scope.

The committee recessed at 3:42 pm Tuesday and reconvened at 1:35 pm Wednesday.

7 Working paper for the draft proposed standard

Lenkov opened the committee of the whole.

Shopiro presented the editor's report. He said he spent most of his time improving the description of declarators. That description is now better than it was, but still not quite satisfactory.

Shopiro explained that the C standard uses the notion of "compatible type" in several places. Almost everywhere that the C standard mentions "compatible type", the C++ draft refers to "the same type". However, he has begun to introduce the notion of compatible type into the C++ draft. For example, the result types of the second and third expressions in a conditional (?:) expression must be "compatible". Shopiro said this needs more work.

Shopiro also explained that the index originally distributed with the Sept. '91 draft (N0048 = 91-0015) had dashes missing from the section numbers, (e.g., 5-12 appeared as 512). He distributed a new index.

Shopiro reported that the draft no longer says that enumerations are integral types, although enumeration values still promote to *int*. He believed the current draft captures the original intent. Plum noted that this is an extra incompatibility that didn't exist before. Pennello expressed surprise that *a++* (where *a* has an enumeration type) is no longer allowed. He asked how do you now write a *for* loop that steps through the values of an enumeration. Koenig said use a cast.

Krohn noted that section 7.2 still says that an enumeration is a distinct integral type. Schwarz noted an oversight in section 12.8: the first sentence of paragraph 5 should be amended to include base classes as well as members.

Shopiro summarized his plans for the next meeting as:

-- fix the two errors noted above
-- add the new keywords and operators approved in Lund
-- continue to refine the description of declarators
-- add discussion on incomplete types
-- continue incorporating terms from C standard (from 90-0088)

Lenkov closed the committee of the whole.

Motion by Wilkinson/Johnson:

"Move we accept the current Working Paper (N0048 = 91-0115) with the two corrections noted previously by Krohn and Schwarz."

Pennello questioned whether eliminating ++ and -- for objects of enumeration type is a substantive change. Stroustrup said that *++a* (where *a* has an enumeration type) was not allowed previously. He explained that *++a* is same as *a += 1*, and += is not applicable to enumerations.

Motion passed X3J16: lots yes, 0 no.

Motion passed WG21: 4 yes, 0 no.

8 Proposal for the rationale

Lenkov opened the committee of the whole.

Waggoner proposed writing a two-part rationale, outlined in N0064 = 91-0131. The section numbers in the first part match the section numbers in the C++ draft standard, and each section provides rationale for the corresponding part of the draft. The second part of the rationale is a collection of essays on issues handled by the different working groups.

In response to a question from Koenig, Lenkov said that the rationale will be a numbered document distributed for public review along with the draft. Plum had heard that the ISO editors may be inclined to accept rationale documents in standards. Carter suggested that the rationale could be an appendix to the draft.

The committee discussed copyright issues regarding essays in the rationale. Stroustrup feared that the contributions of some people might be hundreds of pages long and lead to copyright problems. Koenig suggested that the authors could copyright their essays, but grant ISO and/or ANSI the right to distribute their essay(s) for standardization purposes. ISO and/or ANSI could copyright the collection, although they wouldn't have rights to any of the essays.

Straw vote: Who favors Waggoner's approach to the rationale document? lots yes, 1 no, 3 abstain.

9 General session

--- Extensions ---

Stroustrup summarized the process for evaluating proposals for extensions. He distributed the first draft of a letter on how to write proposals for extensions.

Stroustrup reported that the proposal from the Univ. of Waterloo on concurrency (N0066 = 91-0133) was reviewed and judged not to be a formal proposal. It will not be brought before the whole committee for a vote. Stroustrup also explained that the Extensions group could not recommend Adcock's proposal to allow overloading for the dot operator without better justification. Kearns volunteered to study the proposal further.

In a discussion of how to reject a document, Saks noted that all documents considered by a WG should have a number so their status can be logged in the document list (at the time, Adcock's paper on overloading dot still did not have a number).

Stroustrup listed the open issues and the committee members responsible for providing papers on these issues:
-- runtime type identification: Stroustrup, Lenkov, Kiefer
-- keyword parameters: Eckel, Gibbons
-- return type of virtuals: Lenkov, Bruns
-- array *new* and *delete*: Holly, Gibbons
-- name space pollution: Dovich, Hartinger
-- character set issues: Stroustrup, Hartinger
-- ~*const*: Knuttila, Codella
-- templates: Sloane, Knuttila, Gautron
-- static member initialization: ?
-- *operator.()*: Kearns

--- Core Language ---

Koenig summarized the WG's progress on name lookup issues. The WG had discussed Koenig's proposals (email message *x3j16-core-820*) and found sufficient problems with the proposal that Koenig withdrew it. The WG reached general agreement on an alternate solution outlined by Pennello. Koenig summarized the first half of the agreement with the following example:

```
struct X {
        ZDDDDDDDDD?
    int a3;        3
   ZDDDDDY  ZDDD?  3
   3void f()3 { 3} 3
   3     ZDDY   3  3
   3int b3;     3  3
   3     @DDDDDDY  3
   @DDDDDDDDDDDDDDDY
};
```

Under Pennello's proposal, the scope of *a* is the outer box, and the scope of *b* is the inner box. That is, the scope of *b* is everything in *struct X* that occurs after the declaration of *b*, plus all function bodies and constructor initializers for functions defined in *X*.

Koenig gave a rough description of the proposed rule: the scope of a name declared in a class consists not only of the text following the name's declarator (to the end of the class), but also of all function bodies and constructor initializers in that class.

Koenig and Pennello presented several illustrative examples:

1.
```
struct A {
    typedef int T;
    struct B {
        void f() {
            typedef double T;  // 1
            T x;  // T is double
        }
        typedef char T;  // 2
        T z;  // T is char
    };
    T y;  // T is int
};
```

This is legal. If you delete // 1, it's still legal, but *f*'s *x* is of type *char*, because the scope of // 2 extends into the function body. If you delete // 1 and give *f* an argument of type *T*, then it would be an error, because of the other half of the proposal (described later) which subsumes the current redefinition rule and rejects a wider class of undesirable programs.

2.
```
struct A {
    T f() { T x; }
    typedef int T;
};
```

This an error because the scope of *T* extends only into the body of *f* and not to the return type. Therefore, the return type *T* is undeclared at this point.

3.
```
typedef int T;
struct A {
    T x;
    typedef int T;
};
```

This is an error, determined as follows: when the compiler sees the end of *struct A*, it goes back and looks again at every name that was used inside *struct A*. It interprets those names in the "completed" scope of *A*. When it does this, it finds that the use of *T* refers to the second *typedef*. Koenig said he prefers to think of this as an ambiguity error.

Koch asked "what about default argument values?" Koenig replied that default arguments are not in the function body nor in the constructor initializer, so they are treated like everything else in the class body.

Pennello presented a tentative statement of the rule:

> Except for function bodies and constructor initializers, a name used in a class scope *S* must denote the same declaration when *S* is "complete".

Koenig explained that the WG plans to have a formal proposal on these name lookup rules ready for the next meeting. Pennello and Turner will write that proposal with a rationale and examples.

Stroustrup agreed to write a paper on friend function issues for the mailing before the next meeting.

Koenig explained that at the Nashua (March '91) meeting, the Core Language group agreed that early destruction of temporaries was preferred, but the whole committee shot down this agreement. He presented a new "killer" example:

```
#include <iostream.h>

ostream f() { return cout; }

main()
{
    f() << "Hello" << "world" << "\n";
}
```

With early destruction, *f() << "Hello"* returns a reference to a temporary *ostream* object that's been destroyed (*<<* returns *ostream &*). Koenig now believes that temporaries should live as long as possible but not so long that you need to set flags to note if the temporary was created (as under a condition). For example, Koenig suggested that

```
String x, y;
char *z;
...
z = f(x + y);
```

might be compiled as

```
String x, y, t;
char *z;
...
t = x + y;
z = f(t);
```

and so *t* should persist until after *f* returns, but not beyond the end of any conditional (*if*, *while*, etc.) enclosing the call. He explained that he was not ready to make a formal proposal, but he will work on this as soon as the name lookup issue is resolved.

Stroustrup noted that Koenig was careful not to use the term "basic block" because the precise definition is slightly different from the region described by Koenig, but you can think of Koenig's suggestion as "temporaries persist to end of the enclosing basic block". Plum asked if you could have a conditional between the statements

```
t = x + y;
z = f(t);
```

as long as it converged before the second statement. Stroustrup said yes, and that's why Koenig was careful not to call it a "basic block".

Schwarz noted that Koenig's first example (using *iostreams*) won't compile under cfront 3.0 because it initializes a *const* reference with a temporary.

Lenkov asked for volunteers to handle some of the other core language issues, but no one volunteered. Koenig will produce list of issues and divvy them up later.

Lenkov closed the committee of the whole.

The committee recessed at 4:00 pm on Wednesday and reconvened at 8:40 am on Thursday.

10 General session (continues)

Lenkov opened the committee of the whole.

Vilot summarized the work of the Libraries WG. He presented a revised version of the list of open issues (that first appeared as page 3 of X3J16/91-0030). Among the additions were:

1. under Mixed C/C++ Interactions:
   -- a template function called *renew* that provides functionality similar to *realloc*
   -- an alternative *assert* that throws an exception
2. under String:
   -- exception specifications
   -- substrings and temporaries
3. under Other:
   -- container classes

Vilot said that Koenig described the *renew* template function in his C++ column in the *Journal of Object-Oriented Programming*. Vilot briefly summarized Schwarz's proposal for standard exceptions (N0049 = 91-0116), including the proposal to allow *assert* to throw an exception.

Vilot provided some detail on the issues concerning the input/output library:

1. National character set data streams
2. File open modes, newline translation, etc.

   -- Implementors may have to add parameters on constructors to provide access to system-specific features.
   -- The rules for library conformance must find a way to say it's OK to add such parameters.
3. Wide character support
4. Interaction with other standards, e.g. ASN.1
5. Names of (existing C++) headers
   -- The WG tentatively decided to provide access to the standard library by using the C library convention of header "files".
   -- The headers for the new part of the C++ library might not use *.h* suffix.
   -- Existing C header files will, of course, use the *.h* suffix.
6. *streampos*, *streamoff* "types"
   -- They were specified as classes, but this may be over-specification.
   -- Now they are specified by a *typedef* with ellipsis, giving implementors latitude to specify them as types or classes.
7. Exceptions thrown by *streambuf*
   -- Under the current proposal, *streambuf*s always throw an exception upon error.
   -- Maybe users should be able to select whether streams return an error code or throw an exception.
8. Mode flag "types"
   -- Current practice represents each mode flags as an enumeration value. Users combine modes by *or*-ing flags together producing an *int* value that's passed to a mode-setting member function.
   -- This makes type checking impossible.

Koenig suggested making the modes a class. Schwarz said he would have made the modes classes if he could do it over, but he feared breaking existing code. Bruck suggested overloading | (or) to operate on enumerations (of the same type) and yield an enumeration (of that type). Koenig explained that this only works if the result of | is a value in the enumeration type of the operands.

Vilot continued with the details of the issues concerning the input/output library:

9. Name space
10. I/O support for strings and wstrings
11. *stdio*/*streams* interactions (mixing *printf* and <<)

Vilot explained that the WG agreed to include the ISO C library in the C++ standard by referencing (as opposed to copying) specific paragraphs in a specific version of the ISO C standard. The C++ standard should state explicitly where it differs from the C standard. The C++ library rationale should explain each change or omission from the C library.

Chapin asked if a mixed C and C++ implementation would have only one copy of the C standard library that is shared by both C and C++. Vilot said that will be unspecified. The C++ standard will specify that the standard i/o facilities will be available by including *stdio.h*. Whether it binds to the C library is implementation-dependent.

Vilot explained that the WG was considering an *mbstring* (multibyte string) class in addition to *string* and *wstring* classes. The WG decided to back up from previous work and establish more complete requirements for string classes.

Vilot said the WG considered providing access to the *char* * representation of a string, but only by explicit conversion. Koenig asked if it would be possible to put a '\0' into a string. If the standard specifies an underlying implementation using null-terminated strings, then putting a '\0' into a string might change its value. Becker said the representation won't necessarily be null-terminated.

Vilot summarized some of the issues concerning the definition of library "conformance". He suggested that implementors should be able to:

-- add members (especially private ones) to a library class. The standard won't specify private members; an implementation must add its own.
-- add additional, defaulted arguments to functions (e.g. file open modes).
-- derive library classes from base classes.
-- make some member functions virtual or non-virtual, if the standard does not specifically require one or the other.

Vilot reported that a review of available libraries revealed a small set of common classes: string, vector or array, bitset, list, queue, stack, map or directory, and hash table. The WG considered specifying "simple" classes using parameterization but not inheritance. They have volunteers to investigate vectors and bitsets. The WG also wants to consider language independent data types, and numeric types (those under investigation by NCEG).

Stroustrup said he doesn't think the committee has any choice but to look at the issue [of containers]. Users often ask why aren't we standardizing these sorts of things. We can't do all they ask because the job is too big and controversial. We shouldn't make promises we can't keep, but because of great expectations and interest, we must look into the issues to see what we can and can't do.

Stroustrup also said he saw this set of classes as "concrete types". It doesn't contain the grand generalities of a SmallTalk hierarchy in which you can't get one of them without significant extra baggage. This is both good and bad. The only possibly acceptable compromise involves classes at this level that you can use independently. He said the possibility of mapping them into some rooted hierarchy is intriguing, because that would allow a compromise he hadn't thought possible.

The committee discussed whether this set of classes was adequate to consider. Schwarz said that the WG should consider at most these classes and possibly less, not at least this set and possibly more. Steinmuller said he had the feeling from previous discussions that there was no agreement to do all these classes. The WG decided that the array class was doable and useful, and so was the bitset class. There was no

agreement to do any of the other classes. Some thought the array class might be easier than the string class. He didn't think so, but thought we should try to do some of it.

Stroustrup said that whether it is our responsibility to do this, there are many reasonable people who think we should do much more than this. He suspected that only by working on the issue will we get an idea of how to limit our work.

Vilot listed the group's work plan (and responsible individuals):

1. Decide error/exception approach (technical session)
2. Proposal for 18.1 Language Support (Vilot)
3. Proposal for 18.2 Input/Output (Schwarz)
4. Define C/C++ library interaction
5. Proposal for 18.3 ISO C Library (Clamage)
6. Proposal for 18.4 String Class (Becker)
7. Start on rationale sections
8. Header "file" names (Ward)
9. Decide container classes approach
   -- Vector (Steinmuller)
   -- BitSet (Allison)
10. Description of library "conformance" (Vilot)

Saks questioned committing resources to investigating container classes. He said it might be a disservice if we didn't standardize some of these things, but it will also be a disservice if we don't get a standard out in a reasonable time frame. Several members argued that if volunteers wanted to tackle these assignments, no one could stop them.

Bruck said that standardization work is an excellent example of parallel processing. These issues will be raised sooner or later, so we might as well start working on them right away. People will wonder why we didn't consider bitsets, why we didn't consider arrays. It's best to start considering issues.

Saks said the problem is not just individuals time. It's also the working group time and the open committee time that is spent in discussing the issues.

Stroustrup restated his argument that the user community is very interested in these classes, and we must invest time to see how much we can really do.

Straw vote: Who favors an investigation into bitset and vector classes? lots yes, 3 no, 0 abstain.

Straw vote: Who approves the library group's plans? lots yes, 1 no, 1 abstain.

--- Environments ---

Chapin introduced a report from the Environments WG (N0056 = 91-0123) that summarized the issues for and against adding translation limits to the C++ draft. He posed this question to the committee:

> "Should the standard contain a section on translation limits similar to that in C standard (with some numbers increased and new C++ specific numbers added)?"

He listed example of C++ specific limits:
-- number of direct base classes
-- number of levels of nested classes
-- number of template arguments
-- number of virtual functions
-- number of handlers associated with a try block
-- number of virtual base subobjects

Welch said he had no objection to limits in principle, but cautioned that we must choose the limits carefully. Stroustrup expressed two worries about translation limits: (1) he tends to bump into them, and (2) managers don't understand that these limits are supposed to be for minimal systems.

Clamage opposed limits. He observed that the WG report suggests that even if the limits don't do any good, they don't do harm. He disagreed, citing commentary on *comp.lang.c* and *comp.std.c* complaining about the ineffectiveness of the limits in the C standard. He further explained that people have said that demonstrating only one program which hits all the limits proves nothing, and that the [C] committee must not have been very bright to think that this was worth the bother. Clamage concluded that limits are harmful, because they makes people think less of the committee.

Plum favored limits. He said that if there are no limits, programmers must be aware of the limits on every possible target system on which they develop software. He explained the "rubber teeth" limits specified by the C standard: a conforming translator must compile at least one program that reaches each translation limit. Plum said that you could complain that testing only one program is no guarantee of capacity in general, but in practice it appears to give reasonable assurances. He added that omitting limits from the C++ standard raises a C compatibility issue. If the C++ standard doesn't support limits at least as high those for C, there's no assurance that C programs that observe C's limits will work as C++ programs.

Becker argued that C's implementation limits don't measure the right thing; they should measure the total resources used by a compilation. As Plum pointed out, real-world translators handle programs that exceed the limits. Thus "rubber teeth" limits aren't really meaningful.

Eckel was against limits, but he understood Plum's concern for C compatibility. He said just adopt C's limits and be done with it.

Stroustrup again cautioned that minima tend to become maxima. By picking limits, we're setting maxima for some implementations. He recommended that the C++ standard pick unreasonably large limits, but not require that any program hit all of them at once.

In refuting Plum's assertion that real-world programs don't run up against the limits, Koenig cited an example of a C compiler that enforced portability by treating code that exceeds a limit as an error (not a warning). cfront output easily exceeds the translation limit for nested blocks (15), and that C compiler refused to compile such output.

Schwarz observed that committee members were using "conformance" and "portability" interchangeably, when in fact, the terms are not interchangeable. Portability is the practical problem of writing code that runs under a large class of compilers, and Standard C is just another C dialect that he must deal with. Schwarz also noted that he runs up against runtime limits more often than compile-time limits.

Koenig offered another way to specify limits: publish both minimum and maximum limits, where a maximum may be considerably higher than its corresponding minimum. Specify that each compiler must compile at least one program that reaches all minimum limits, and for each maximum limit, at least on program that reaches that maximum.

Plum agreed with Becker that the C standard doesn't address the problem of the total bulk of a program. He also noted that Stroustrup's solution doesn't address this problem, either. Putting criteria upon the total bulk of a program appears to be an unsolvable problem. He said he still preferred Stroustrup's solution because it's a more convenient packaging of what X3J11 tried to do. He also suggested that each vendor supply a "multiplier" that indicates the compilation capacity of a translator as a multiple of the capacity required by the standard.

Clamage took issue with Plum's assertion that if there are no limits, any translation limits are a violation. If you set no limits, the user can't claim the translator doesn't conform for failing to meet a limit. He also said he doesn't care if a compiler has unreasonably low limits and can successfully claim conformance, because no one will buy it. If we have very high individual limits, then it may not be possible to have a conforming compiler on a small system.

Holly said that, as an implementor who must deal with limited address space, he appreciates knowing the limits his implementation must meet. For example, knowing the maximum number of arguments to a function lets him know how many bits to set aside for a field that indicates the argument number. If he knows the limit is 31, he can safely set aside 8 bits.

Bjarne noted that Holly showed how limits creep into compilers and become maxima. He expressed concern about (1) the ability of implementors to make good judgments based on limited information, and (2) the ability of bureaucrats to read the standard.

Lenkov summarized the three suggestions for setting translation limits:

1. "rubber-teeth" limits: require that each implementation demonstrate at least one program that tests all limits.
2. require that each implementation demonstrate one program for each limit that reaches that limit.
3. Koenig's proposal that combining (1) and (2): the standard should specify separate minimum and maximum limits, and that each implementation must demonstrate at least one program that reaches all the minima, and for each maximum, demonstrate at least one program that reaches that maximum.

Straw vote: Who favors some limits? lots yes, 4 no, 5 abstain.

Straw Vote: Who prefers (1)? 5 (2)? 25 (3)? 17

Chapin presented the current thinking on static object initialization. The WG had concluded that there should be two "phases" of initialization:

1. initialize objects with "constant" initializing expressions (where "constant" remains to be determined), and
2. initialize objects with non-"constant" initializers (everything else).

Chapin explained the WG's view that each translation unit has (is "as if" it has) an initialization function generated by the compiler. The function evaluates initializers and invokes constructors for static objects in that translation unit, using the same semantics as normal function evaluation (e.g., lifetime of temporaries inside function, and sequence points). The order of invocation of initialization functions is unspecified, but all are done before *main* is entered.

The WG proposed changing the model of a C++ program to distinguish two types of translation units (TUs):

1st Kind: in which all static objects are initialized before entering *main*.

2nd Kind: in which all static objects are initialized before executing any function in that unit via a thread from *main* (as opposed to via a thread from a constructor of another static object).

Someone asked why the WG is abandoning the language in draft. Shopiro said it was to insure that an object with a constructor gets initialized even if it's never referenced, namely, to put the object into a TU of the first kind. He added that TUs of the second kind are there to model dynamic linking or shared libraries. Chapin added that if the current language is taken literally, it's impossible to do in general. For example,

```
file 1:          file 2:

int a = f(b);    int b = f(a);
```

Vilot had three questions:

1. Has the WG abandoned the present wording in the draft?

Chapin replied that the WG had roughly decided to weaken the present wording in the draft to align it with current practice (that the order is unspecified).

2. Was the WG planning to provide programmers with a way to distinguish the two types of translation units?

Shopiro replied that it's outside the scope of language.

3. Is the WG going to define "thread"?

Chapin said that's a good question. (Secretary's note: Chapin later said this meant "yes".)

Chapin summarized the alternatives:
1. Dynamic initialization (every static object has an initialization flag): Run time expense != 0.
2. Static analysis: Complicated (especially in the general case).
3. User specifies the order: Error prone.

Koenig, Shopiro, and Schwarz discussed whether it's possible to program around the lack of specification. Schwarz said it was, but it's difficult. Koenig said it would be best to leave the order undefined if the problem can be solved by programming techniques.

Turner asked for a straw vote on "abandoning the holy grail" (that everything is initialized before it's used to the extent that it's even meaningful). Shopiro said that the proper way to do this (static initialization) is to use global flow analysis. We will be able to do this in future, so we should leave the order unspecified for now, so we can employ global flow analysis in the future.

Chapin said that the WG was leaning toward leaving the order unspecified, and letting programmers use programming techniques to control the order if necessary.

Chapin explained the WG's latest thinking on the one-definition rule (ODR). 91-0024 suggested giving *typedef* names linkage as a means of enforcement, but the group has since rejected that notion. However, enumeration values must still have linkage.

Chapin plans to write a revision to 91-0024, with the following changes:
1. "expand" *typedef* names after preprocessing, but before ODR analysis.
2. 91-0024 suggests that

```
static int Y;
struct X {
    int foo() { return Y; }
};
```

is an error if *X* has external linkage. This should only be an error if it's done in more than one translation unit.

Chapin explained that the WG proposes to:

1. treat class template definitions just like externally linked class definitions.
2. treat function template definitions just like externally linked function definitions.

Chapin presented the following example:

```
template <class T> void f(T thing);
...
int x;
double y;
...
f(x);
f(y);

----------

static int A;
template <class T> void f(T thing) {
    ...
    A = 10; // only one "A" object shared by all instances
    ...
};
```

Chapin said that if we were to treat function template definitions like externally-linked inline function definitions (so they could be placed into headers), then according to the rules in 91-0024 the previous example would be an error, because the template instantiations refer to different internally-linked objects. The WG feels that all the automatically generated instances of *f* should behave as if they were in the same translation unit.

Plum said we need more precision about "names". For externally-linked names, the spelling of a name is probably OK, but for internally-linked or block scope names, we may need to qualify the name somehow.

Chapin listed the work assignments for the Environments WG:

1. Revised ODR proposal (Chapin)
2. Static object initialization (Kearns)
3. Shared library issues for initialization (Wilkinson)
4. Translation limits (Swan)

Schwarz asked about mixed C/C++ environments. Chapin added it to the list as (5), but no one volunteered to do it.

Schwarz noted that his name had been attached to that task in the past, but he's taken on too much work. Clamage recommended that Plum and Chapin meet by Friday morning to find someone to work on item 5.

--- Formal Syntax ---

Pennello reviewed previous votes on template delimiters from the March '91 meeting. He noted that the votes were inconclusive, and he hoped to reach a decision at this meeting. He reviewed 91-0033.

Pennello noted that < and > have problems as template delimiters:
1. two consecutive closing delimiters >> can be mistaken for a single shift operator
2. actual template arguments can't have the form *a > b*

He offered a solution to (2) that rewrites the grammar with a "short-circuit" into the middle of the grammar for expressions:

```
template-arg -> shift-expression
```

The drawback to this solution is that expressions with operators whose precedence is lower than a shift operator must be parenthesized:

```
T<a?b:c>
   ^ syntax error
```

Pennello suggested another grammatical solution that creates a "parallel" grammar for expressions without the > operator:

```
template-arg -> expression-2
expression-2 -> assignment-expression-2
             -> expression-2 , assignment-expression-2
...
relational-expression-2
             -> shift-expression
             -> relational-expression-2 < shift-expression
             -> relational-expression-2 >= shift-expression
             -> relational-expression-2 <= shift-expression
             (omit the rule for >)
```

He noted that default arguments in a template declaration, e.g.

```
template<int x = a > b>
                  ^ syntax error
```

have the same problem as actual template arguments and require a similar fix.

Pennello said that those who think we can solve the problem by "fixing" shift-reduce conflicts in the parser are mistaken. You might always choose a shift over reduce (the common solution), but then you can never

close a template argument list. Choosing reduce doesn't solve the problem either, not even if the user uses additional parentheses, because then expressions everywhere can't have a > operator.

Koenig suggested a lexical hack that turns > into a closing bracket when scanning a template argument list.

Pennello listed additional problems with < and > as template argument delimiters:

3. the parser must know if an identifier is a template name, to recognize constructs like

```
new T < ...
```

4. there's a redeclaration ambiguity in

```
const T;
```

Pennello recommended using ( ) instead of < > as template argument list delimiters. He said this choice eliminates problems 1 and 2, but doesn't change 3 and 4. He noted that we could use [ ] and { } as well. He summarized the pros and cons for using ( ) over < >:

```
+   existing practice uses ( )s in "template" macros
    < and > don't become matching delimiters
    >> problem is gone
    closing > delimiter problem is gone
-   textbooks use < >
    const T; problem remains
```

Pennello suggested an alternative to adopt ( ) and keep < > as an anachronism.

Stroustrup argued that < > are the correct solution. The reasons for using ( ) are all low-level, considering implementation technology only, such as parsers. The main argument for using < > is that they're more readable. Stroustrup said he has personal experience with both styles, and has no doubt about which is easier for human use.

Vilot argued that ( ) are already heavily overloaded. Some Ada users complain that Ada is less readable because it overuses ( ). Vilot claimed that resolving the conflicts properly is a small cost to pay to allow < > as matching delimiters.

Pennello first responded to Vilot by saying that in fact even with the conflicts resolved, < > are not "full" parentheses like ( )s. He reviewed a macro from 91-0008:

```
#define f(x) C<x>
```

Here the < >s don't act like ( )s, and so to guard against passing an argument of the form *a > b*, one must instead write:

```
#define f(x) C<(x)>
```

Charney stated that the choice of < >s clutters C++ text with even more symbols, such as the ( )s in the above macro, or the ( )s required in

```
T< (a > b > c) >
```

(where *T* is a template name), whereas parentheses solve the problem already:

```
T( a > b > c )
```

Pennello disagreed with Str[illegible]rup's argument of "low-level reasoning" and said that the problem is not just a "parsing" problem -- it is a real language definition problem that just happens to manifest itself as a parsing problem. All four problems with the < > delimiters must be explained in the draft, not as an issue of compiler technology, but as language definition. Choosing ( )s reduces the amount of explanation in the draft.

Saks suggested considering Stroustrup's idea (posed at the March '91 meeting) to severely restrict the syntax for actual template arguments to, say, identifiers and literals, and keeping the < > delimiters.

Koenig noted that using ( ) reduces the number of problems from four to two. Also, the second problem rarely occurs in practice.

Straw vote: Who wants any change in using angle brackets? 7 yes, 26 no, 5 abstain.

Pennello asked which of the two syntactic solutions the committee wanted to use to resolve the ambiguity. Plum suggested that a lexical solution exists that's consistent with existing C lexer technology. Pennello responded that Plum's solution was isomorphic to the second grammatical solution (using a parallel expression grammar).

Gibbons wanted to restrict template arguments to primary expressions. Scian supported Saks' suggestion of restricting actual arguments because there's another problem: whether two templates instantiated with different arithmetic expressions that might evaluate to the same value are indeed the same.

Koenig suggested that the problem of mistaking two adjacent closing delimiters as a shift could be solved by a lexical hack.

The committee debated whether this is a syntax issue or a core language issue. Pennello again asked the committee to choose the way to restrict the language.

Straw vote: Who wants to send this to the Core Language WG? 18. To the Syntax WG? 5.

Pennello discussed the unresolved ambiguity in the syntax for throw expressions (91-0034, 91-0013, 91-0048). The ambiguity is in the rule:

unary-expression -› throw *expression-opt*

so

1, 2, throw 3, 4

could be either

1, 2, throw (3, 4)
1, 2, (throw 3), 4

Pennello proposed two possible solutions:

1. *expression*
   -› *throw-expression*
   -› *comma-expression*
   -› *comma-expression* , *throw-expression*
   throw-expression
   -› throw *comma-expression-opt*
   comma-expression
   -› *assignment-expression*
   -› *comma-expression* , *assignment-expression*

   All these rules are changes to the existing grammar. The Syntax group doesn't favor this because it has two "colors" of comma (at different precedence levels).

2. *expression*
   -› *assignment-expression*
   -› *expression* , *assignment-expression*
   assignment-expression
   -› *conditional-expression*
   -› *unary-expression assignment-operator assignment-expression*
   -› throw *assignment-expression-opt*
   conditional-expression
   -› *logical-or-expression*
   -› *logical-or-expression* ? *expression* : *assignment-expression*

   Only the 7th and 10th lines (marked by change bars) are changes to the existing grammar. The Syntax group favors this solution because it poses no conflicts in the grammar, and gives all commas the same "color".

Bruns wondered what happens if you put *return* in place of *throw* in

1, 2, throw 3, 4

Pennello said it's a syntax error because *return* is a statement.

Koenig said he favored the first alternative at Nashua meeting, but now favors the second. He said the second alternative lets you transform commas to semicolons without changing the semantics of the expression shown immediately above. It preserves the intuitive notion that commas have lower precedence than everything except semicolons.

Pennello summarized his preference for the second alternative:

+ only two changes in the grammar
+ *a ? x = y : q = r* is now OK (it wasn't before). This accommodates existing practice (from some pcc's)
+ no conflicts
+ all the commas in *1, 2, throw 3, 4* have the same "color"
+ *throw* can follow a colon in a *cond-expr*
+ *throw-expression* can appear in an argument list
- changes the base languages definition of ? : (by allowing assignment to the right)
- *throw-expression* can appear in an argument list

Pennello explained that the 6th and 8th items above are the same. With the current grammar, you can't write *f(throw a)*, but now you can. It's not clear if this is a plus or a minus. He also noted that under the second proposal,

```
a ? x = y : q = r
```

groups as

```
a ? (x = y) : (q = r)
```

Straw Vote: Who wants to make the second set of changes to the grammar? lots yes, 1 no.

Krohn described a syntax problem in template definitions (91-0061). He noted that formal argument names can be omitted, as in

```
int f(int size, char *) { return size; }
```

and that template declarations allow the same omission:

```
template<class T, int> class U;
```

Now, suppose you declare:

```
class stat;
int stat(const char *, class stat &);
```

then

```
template<class T, stat statbuf> class T1;
                  ^ error
```

is an error because the class name *stat* is hidden. Thus you must write the declaration as

```
template<class T, class stat statbuf> class T2;
```

That is, you elaborate the class name *stat* as *class stat*. Krohn pointed out that if you leave out the formal argument name *statbuf*, as in

```
template<class T, class stat> class T3;
```

then you get an ambiguity: *class stat* could be a type argument or it could be an expression argument of type *stat*. Krohn proposed to resolve the ambiguity with the following rule:

> Whenever a class identifier occurs by itself in a *template-argument*, then it is a *type-argument* and not an *arg-declaration*, even though it looks like both.

He offered a workaround: use the keyword *struct* instead of *class* to indicate an *arg-declaration*, as in

```
template<class T, struct stat> class T3;
```

Krohn offered a second workaround: always provide a *typedef* name as an alias for an *elaborated-type-specifier*, and use the *typedef* name as the formal argument of the template, as in

```
typedef class stat Stat;
template<class T, Stat> class T3;
```

Plum and Clamage asked why you would want a template with an unused formal argument. Schwarz explained that you might have a template class *stack* that takes a size as a template argument, and then you later rewrite the template so it sizes itself. Although you no longer need the size argument, you might not want to change all the template instantiations.

Koenig said he doesn't want *class* to differ from *struct* in any context. Pennello insisted that it is different in this context. Roskind suggested that the Syntax WG should find a way to remove the distinction between *class* and *struct* in this context. Krohn pointed out that any solution must also consider the keyword *union*.

Koenig asked what

```
template<class T, T> class T4;
```

means? Pennello says it means what it always meant. Krohn said this is an issue for the Core Language group.

Saks suggested that Krohn's proposal use *class-key* instead of the keyword *class*. Schwarz said this would be a significant change. Koenig said we should not change it now, and deliberately consider this later.

Straw Vote: Who wants to adopt the text of Krohn's proposal? lots yes, 1 no, 4 abstain.

Roskind reported that Saks found discrepancies between the grammar in the text of the Working Paper and its appendix. Saks will email the list of discrepancies to Shopiro. Roskind also noted that the grammar currently does not have a start symbol, such as *translation-unit*. He asked committee members for input on the choice of a start symbol.

Wilkinson asked what happened to Charney's proposed non-terminal name changes (91-0082 and 91-0083). Roskind said the WG couldn't agree on supporting the changes, and decided not to pursue them.

2

--- C Compatibility ---

Plum explained the draft's specification of declarators and "incomplete types". He said the current draft (91-0059) is much closer to the C standard and considerably more precise than the previous draft. He gave a "guided tour" of declarators:

-- 8.2.5 Functions: The draft is good.

-- 8.2.4 Arrays: The draft needs work on the concept of "dimension". It also needs a rule stating, for example, that the type of *s* in

```
extern char s[];
```

is "array of unknown dimension of char".

-- 8.2.1 Pointers: The description scheme in the current draft employing the term "type modifier" should be applied to the description of pointers, references, and pointers to members.

-- 8.2.5 Functions (again): Wording in section 8.2.5 paragraph 3 states that arguments of type *array of T* are adjusted to type *pointer to T*. This wording obsoletes wording on overloading elsewhere in the draft.

Plum also noted that a *register* storage class applied to a formal parameter is not part of the function type, and it's overlooked in any discussion of overloading. Similarly, there's no clear statement about whether *cv-qualifiers* applied to formal arguments (like *const int*, but not *const ** or *const &*) participate in overloading.

Plum described another problem illustrated by

```
void h(int (*p)[]);
```

The type of *h* is "function of pointer to array of unknown dimension of int returning void". Are declarations like

```
void h(int (*p)[2]);
void h(int (*p)[10]);
```

overloadings or redeclarations of *h*?

Koenig asked if the C standard ascribes meaning to "pointer to array of unknown dimension". Plum said it's like *void*; you can't dereference it nor increment it. Plum suggested that C++ might ban them. Then it's not an issue for overloading.

Straw Vote: Who believes there should be no pointers to array of unknown size? 4 yes.

Plum summarized the work assignments for the C Compatibility WG:

-- distinguish "parameter" from "argument" (Koch)
-- add lexical grammar (Jackson)
-- rewrite Chap. 3 vs. Chap. 7 redundancy (Lajoie)
-- incorporate "incomplete types" (Nelson)
-- allow constant, not constant expression, for null pointer constant (Kohlmiller)

The committee briefly discussed null pointer expressions. Clamage asked if you define

```
const int NIL = 0;
```

could you use *NIL* as a null pointer constant? Plum said the WG hadn't thought about it.

Lenkov closed the committee of the whole.

The committee recessed at 5:30 pm on Thursday and reconvened at 8:35 am on Friday.

11 General session (continues)

Lenkov opened the committee of the whole.

--- C Compatibility ---

Plum continued with the C Compatibility WG's report. He listed another work assignment:

-- define conformance (Johnson)

Plum said that two email messages on the *x3j16-compat* reflector may be of general interest:
-compat-12: the list of definitions from the C standard that should be reconciled with the C++ draft
-compat-38: the first draft of differences between C++ and C requested by SC22.

Plum said he'd announce on the *-all* reflector when a new copy of the differences document becomes available. He recommended that all WG chairs announce the availability of their reports on the *-all* reflector.

Plum explained that the editorial changes in 90-0088 are on the C Compatibility WG's critical path. He proposed that each WG identify editorial changes that are in its critical path, and that the editor attend to these changes first.

Kohlmiller said it appears that the preprocessing chapter is on the critical paths of several WGs. Shopiro explained that adding the preprocessing chapter is trivial. However, adding many of the C compatibility definitions involves subtle core language issues, and they have been difficult to make. Saks said he appreciated the difficulty, and no one expects Shopiro's first attempt to be perfect, but if the wording isn't even in the draft, no one can even help Shopiro correct it.

A few members spoke favorably about the quality of Shopiro's work. Bruck said this is not a discussion for the whole committee. Koenig noted there's a general issue of how to treat logjams that should be discussed by the whole committee.

Shopiro offered to continue consulting with Plum on adding the definitions. Plum asked for a straw poll on whether editorial changes on critical paths must be done first. Schwarz said he didn't see a problem. If a change is purely editorial, then who is being held up? If a change is substantive, then it must be handled like all other changes. Plum replied that the committee decided over a year ago to merge the terminology from the two base documents, yet only 15% the terms have been been incorporated in the C++ draft.

Straw Vote: Who believes that editorial changes on the critical path of a WG should be implemented first, and that the C Compatibility WG's list of definitions is on its critical path? 11 yes, 13 no, 16 abstain.

Lenkov closed the committee of the whole.

Motion by Roskind/Bruns:

"Move that the following changes (noted by change bars) be made to the syntax in the draft (for *throw* expressions):

assignment-expression
-› *conditional-expression*
-› *unary-expression assignment-operator assignment-expression*
-› throw *assignment-expression-opt*
conditional-expression
-› *logical-or-expression*
-› *logical-or-expression* ? *expression* : *assignment-expression*

Motion passed X3J16: 35 yes, 0 no.

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

Motion by Charney/Gibbons:

"Move that we add the following wording to the draft (disambiguating template arguments): Whenever a class identifier occurs by itself in a *template-argument*, then it is a *type-argument* and not an *arg-declaration*, even though it looks like both."

Motion passed X3J16: lots yes, 1 no.

Motion passed WG21: 7 yes, 0 no, 0 abstain.

12 New business (if any)

12.1 WG14 liaison report

Simonsen explained that WG14 hadn't met since WG21|X3J16 met in June, so he had nothing to report. The next WG14 meeting is in Milano, Italy in December, 1991.

Simonsen reported that he attended the WG20 (internationalization) meeting in San Jose, CA earlier that week. WG20 has been working on the following items:

1. Proposing an NP (new proposal) addendum to TR 10176 - guidelines for the design of programming languages.
2. Proposing a number of NPs on internationalization by adapting work from the ISO POSIX working group on cultural elements (*locale* and *charmap*). WG20 intends to produce a registry of *locale*s and *charmap*s. They will also write an NP on short mnemonics for characters.
3. WG15 (POSIX) asked WG20 how to extend the allowable characters in identifiers, as proposed by WG14. WG20 is addressing a proposal from WG14 to add *is_wident()*, a function to determine if a *wchar_t* is allowed in an identifier.

Plum summarized AFNOR's proposal made in June '91 for 8-bit characters in identifiers. He asked Simonsen if anything has been done about it. Simonsen replied that AFNOR's request was communicated to WG20, but WG20 delayed their response.

Plum asked if anything has been done about null characters in ISO 10646 character strings. (ISO 10646 is the new international standard large character set.) Simonsen said that SC2/WG2 (the ISO WG developing 10646) responded that they have been aware of this issue and they stand by their current formulation. There may be characters that have a null octet (that's "byte" to you Americans) as part of some character. Plum cautioned the committee that when we define strings, we must be aware that they might have null bytes in them.

Schwarz pleaded for someone who understands international issues to attend the Libraries WG meetings. He welcomed participation from non-Americans and non-members who know something about international issues.

Koenig understands that if ISO 10646 passes, then characters will be 16-bits, and 8-bits characters will be an anachronism. Simonsen said 10646 is really a 32-bit code. SC22 and SC2 has said that 8-bit and wider character sets will coexist for some time. There will be mechanisms to shift in : out of these different coded character sets.

Simonsen will include a copy of the resolutions from the WG20 meeting in the next WG21|X3J16 mailing. Lenkov said that WG21|X3J16 will designate Simonsen as the liaison with WG20, but no formal resolution was made.

12.2 Vice-chair position

Lajoie withdrew her application for vice-chair, so Clamage was the only candidate.

12.3 Email reflectors (revisited)

Carter asked if meeting minutes should be posted on the *-all* reflector or some new reflector.

Koenig said that most of the bounced email messages occur on the *-all* reflector. In the next few months you should get a message from Koenig saying in effect "respond to this or I'll drop you".

Shopiro asked if anyone does not want to the receive minutes. No one said they did not. Saks agreed to post the minutes on the *-all* reflector.

Bruck suggested setting up an FTP archive for large messages. Koenig said he'd do it.

Stone asked about non-members sending mail to reflectors. Koenig said we can't stop people from sending email to any reflector, but they only get mail back if they're on the list of recipients. Clamage recommended that members not distribute the reflector addresses, to cut down on unwanted mail.

Lenkov said that every ISO WG member can be an ex-officio member of X3J16 by contacting Clamage.

Roskind asked about non-members participating in WG email reflectors. Koenig suggested that a WG chair can invite anyone to join the reflector. Lenkov noted that a WG chair must ask Clamage to add a new name to the WG's reflector.

13 Review of the meeting

13.1 Review of decisions made and documents approved

See Appendix B.

13.2 Review of action items

Lenkov summarized the action items:
-- Pennello will write a new proposal on name lookup.
-- Stroustrup will a paper on friend functions.
-- Koenig will write a paper on the lifetime of temporaries.
-- Carter will send a portability specification to Schwarz and documents on language independent data types to Vilot.
-- Carter will explore the ramifications of translating the draft document into French.

14 Schedule of mailings and volunteers to help

Carter confirmed that the following volunteers will handle future mailings (with the thanks of the committee):
-- Bellcore: Jul '92, Jan '93
-- Tektronix: Mar '94

Carter also noted the volunteers for the next two mailings (the dates listed are the dates by which documents must arrive at Clamage, care of Taumetric):
-- Apple Computer, post-Dallas mailing (Nov 29, '91)
-- Sun Microsystems, pre-London mailing, (Feb 4, '92)

Shopiro asked if seven weeks was really needed as lead time for the mailings. Carter explained that even with seven weeks lead, not all the mailings arrived two weeks before this meeting.

Bruck registered his displeasure with the way the mailing to Europe was handled this time (from Bellcore to BSI to European members).

Clamage will make copies of each document and send one set of copies to the American distributor, to the European distributor (BSI), and to the Far Eastern distributor (ITJSC).

Koenig asked that mailed documents have holes punched in them so they can be placed in ring binders.

Carter solicited volunteers for future mailings. In particular, he said we need someone to handle the pre- and post-meeting mailings for the meeting in Nov '92.

Charney asked if Zortech will do the mailing after the March '92 meeting. Carter said it was his understanding that Symantec (who bought Zortech) stands by Zortech's commitments. Carter said he'd confirm this after this meeting.

15 Plans for the future

15.1 Agenda items for the next meeting

Vilot and Koenig asked not to have a technical session on Tuesday, especially during the daytime. Lenkov agreed that technical sessions will only be on Monday and Wednesday of the next meeting.

15.2 Technical sessions for the next meeting

Lenkov asked for suggestions for technical sessions.

Carter said Brian Meek will hold a session on language independent data types. Carter plans another session on international character handling. Schwarz invited the person giving that technical session to attend the Library WG meetings.

15.3 Day Schedule for the London Meeting

Carter explained that the meeting will be held in the BSI Conference Center. The center is open from 8 am to 8 pm. He said we might meet from 8 am to 5 pm, and then hold technical sessions from 6 pm to 8 pm. He also said that the conference center is not open on Sunday. WG21 will meet at Cumberland Hotel on Sunday.

Koenig noted that we can't start meeting at exactly 8 am if the building doesn't open until 8 am. Bruck suggested dropping technical sessions from the London meeting.

Straw vote: Who favors dropping technical sessions from the London meeting? 13 yes, 14 no.

Lenkov said he'd use his judgement about holding technical sessions.

15.4 Confirming hosts for the next three meetings

Referring to N0023 = 91-0101 and N0040 = 91-0107, Carter said the date for the next meeting (in London, UK) has been changed from March 8-13, 1992 to March 15-20, 1992. He discussed several other changes in the meeting schedule beyond 1992. The committee discussed scheduling problems for the meeting in Japan in 1993.

Lenkov noted there was no objection to next year's planned meeting schedule (including the date change for the March '92 meeting).

18.5 Call for hosts for meetings in 1993

Lenkov deferred further planning.

19 Adjournment

The committee thanked the Mary Fontana and Texas Instruments for hosting the meeting.

Motion by Roskind/Charney:

"Move we adjourn."

Motion passed without objection.

The committee adjourned at 12:10 pm.

Appendix A - Attendance

Name	Affiliation	Status	M	Tu	W	Th	F
Saks, Dan	Saks & Associates	P	V	V	V	V	V
Plum, Thomas	Plum Hall	P	V	V	V	V	V
Clamage, Steve	TauMetric	A	V	V	V	V	V
Bruns, John	Chicago Research & Training	P	V	V	V	V	V
Holly, Mike	Cray Research	P	A	A	A	A	A
Dovich, Steven	Cadence Design Systems	A	A	A	A	A	A
Pieb, Wolfgang	Amdahl	P	V	V	V		
Stone, Paul	Perennial	P	V	V	V	V	V
Adamczyk, Steve	Edison Design Group	O	A	A			
Nelson, Clark	Intel	P	A	A	A	A	A
Allison, Chuck	DECUS	P	V	V	V	V	V
Islam, Shaun	Texas Instruments	A	A	A	A	A	A
Fontana, Mary	Texas Instruments	P	V	V	V	V	V
Kennedy, Brian	Texas Instruments	A	A				
Traughber, Tom	Texas Instruments	A	A	A	A	A	A
Kohlmiller, Paul	Control Data	P	V	V	V	V	V
Kelley, David	Data General	P	V	V	V	V	V
Kearns, Steven	Software Truth	P	A	A	A	A	A
Chapin, Peter	Vermont Technical College	P	V	V	V	V	V
Dodgson, David	Unisys	P	A	A	A	A	A
Rabinov, Arkady	Apple Computer	A	A	A	A	A	
Eckel, Bruce	Revolution 2	P	V	V	V	V	V
Charney, Reg	Program Conversions	P	V	V	V	V	V
Scian, Anthony	Watcom	P	V	V	V	V	V
Welch, James	Watcom	O	A	A	A	A	A
Wilkinson, John	Silicon Graphics	P	V	V	V	V	V
Mehta, Michey	Hewlett-Packard	A	V	V	V	V	V
Johnson, Andy	Open Software Foundation	P	V	V	V	V	V
Turner, Prescott	Liant Software (LPI)	A	V	V	V	V	V
Codella, Chris	IBM Research	S	A	A	A		
Koch, Gavin	SAS Institute	P	V	V	V	V	V
Gautron, Philippe	Rank Xerox	P	V	V	V	V	V
Knuttila, Kim	IBM	P	V	V	V	V	V
Lajoie, Josee	IBM	A	A	A	A	A	A
Bruck, Dag	Lund Inst. of Tech.	P	V	V	V	V	V
Zeiger, Paul	US West	A	A	A	A	A	
Swan, Randall	C-Team	P	V	V	V	V	V
Becker, Pete	Borland International	P	V	V	V	V	V
Krohn, Eric	Bellcore	A	A	A	V	V	V
Carter, Steve	Bellcore	P	V	V	A	A	A
Yaker, Laura	Mentor Graphics	P	V	V	V	V	V
Ward, Cynthia	Tektronix	P	V	V	V	V	V
Waggoner, Susan	US West	P	V	V	V	V	V
Shopiro, Jonathan	AT&T (USL)	P	V	V	V	V	V
Schwarz, Jerry	Lucid	A	V	V	V	V	V
Hartinger, Roland	Siemens Nixdorf	P	A	A	A	A	A
Steinmueller, Uwe	Siemens Nixdorf	A	A	A	A	A	A
Buschmann, Frank	Siemens AG	A	A	A	A	A	A
Kiefer, Konrad	Siemens AG	P	A	A	A	A	A
Koenig, Andrew	AT&T	A	A	A	A	A	A

Stroustrup, Bjarne	AT&T Bell Labs	A	A	A	A	A	
Roskind, Jim	Roskind Software	P	V	V	V	V	V
Vilot, Mike	ObjectWare	P	V	V	V	V	V
Winder, Wayne	Digital Equipment	P	V	V	V	V	V
Downing, Glenn	MCC	P	A	A	A		
Saito, Nobuo	Keio University	P	V	V			
Yamaryo, Masakazu	Fujitsu	P	V	V	V	V	V
Pennello, Tom	MetaWare	P	V	V	V	V	V
Gibbons, Bill	Apple Computer	P	V	V	V	V	V
Sloane, Alan	Sun Microsystems	A	V	V	V		
Jackson, Paul	SCO Canada	A	V	V	V	V	V
Lenkov, Dmitry	Hewlett-Packard	P	A	A	A	A	A
McLay, Michael	NIST	P		V	V	V	V
Moudgal, Praveen	Microtec Research	A			V	V	V
Sehorne, Mark A.	IBM	O				A	
Simonsen, Keld	DS/DKUUG	P					V
Scherrey, Ben	Business Data Management	O					A
Munch, Max	Lex Hack & Associates	O	A			A	
Total Attendance			62	60	59	56	54
Total Voting Members			39	40	40	38	39

Status: P = Principal, A = Alternate, S = Second Alternate, O = Observer
Mark: V = voting; A = attending (not voting)

Appendix B - Motions Passed

1. Motion by Plum/Saks: "Move that we approve the minutes from the previous meeting." Motion passed: lots yes, 0 no, 1 abstain.

2. Motion by Saks/Bruck: "Move that we accept the proposed agenda with these additions: (1) reschedule the WG14 liaison report under 1.10 as item 12.1, and (2) add item 1.11 on Email reflectors." Motion passed: lots yes, 0 no, 0 abstain.

3. Motion by Wilkinson/Johnson: "Move we accept the current Working Paper (N0048 = 91-0115) with the two corrections noted previously by Krohn and Schwarz." Motion passed X3J16: lots yes, 0 no. Motion passed WG21: 4 yes, 0 no.

4. Motion by Roskind/Bruns: "Move that the following changes (noted by change bars) be made to the syntax in the draft (for *throw* expressions):

    assignment-expression
        -> *conditional-expression*
        -> *unary-expression assignment-operator assignment-expression*
    ¦   -> throw *assignment-expression-opt*
    conditional-expression
        -> *logical-or-expression*
    ¦   -> *logical-or-expression* ? *expression* : *assignment-expression*

    Motion passed X3J16: 35 yes, 0 no. Motion passed WG21: 6 yes, 0 no, 1 abstain.

5. Motion by Charney/Gibbons: "Move that we add the following wording to the draft (disambiguating template arguments): Whenever a class identifier occurs by itself in a *template-argument*, then it is a *type-argument* and not an *arg-declaration*, even though it looks like both." Motion passed X3J16: lots yes, 1 no. Motion passed WG21: 7 yes, 0 no, 0 abstain.