Document Number: X3J16/92-0041
WG21/N0118
Date: April 8, 1992
Project: Programming Language C++
Reply to: Dan Saks
dsaks@wittenberg.edu

X3J16 Meeting No. 8
WG21 Meeting No. 3
March 16-20, 1991

BSI Conference Center
Hampden House
61 Green St.
London, W1 UK

1 Opening activities

Lenkov convened the meeting as chair at 9:05 (UTC) on Monday, March 16, 1992 by Lenkov. Clamage was the vice-chair, and Saks was the secretary.

BSI (British Standards Institute) and Symantec hosted the meeting.

1.1 Opening comments

1.2 Introductions

Clamage circulated the membership list and asked attendees to make any necessary corrections. Saks circulated an attendance list each day, which is attached as Appendix A of these minutes.

1.3 Membership, voting rights, and procedures for the meeting

Lenkov reminded the attendees that this is a joint meeting of WG21 and X3J16. In straw votes, all WG21 technical experts may vote, even if this is the first meeting they've attended; however, X3J16 attendees can vote only if they are the voting representative of a member organization that has been represented at either of the previous two meetings. In WG21 formal votes, only the head of each national delegation may vote. In X3J16 formal votes, only one representative from each X3J16 member organization may vote if the organization meets the aforementioned attendance requirement.

Carter explained that WG21 selects a drafting committee for each meeting. Each motion must be presented in writing to the drafting committee for approval and possible rewording before the committee can vote on it. This insures that members whose native language is not English (and even those whose native language is English) have an opportunity to read the motion and understand it before voting. The drafting committee for this meeting was Saks and LaJoie. Saks invited others to join, but no one did.

1.4 Approval of the minutes from the previous meeting

Saks submitted the minutes from the previous meeting (91-0136 = N0069) for approval. He noted several corrections:

1. on page 5, in the first paragraph, change

   "-- return type of virtuals (O'Riordan offered to write a paper on this, but it's not yet available)"

   to

   "-- return type of virtuals (91-0051)"

2. on page 12, in the sentence beginning "Schwarz noted ...", change "*const*" to "non-*const*".

3. on page 15, in the sentence beginning "Saks said ...": add an ' (apostrophe) immediately after "individuals'".

4. on page 17, in the last sentence of the paragraph beginning with "Koenig offered another way...", change "on" to "one".

Motion by Saks/Brück: "Move that we approve the minutes from the previous meeting with these corrections."

Motion passed X3J16: lots yes, 0 no, 1 abstain.
Motion passed WG21: 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 (92-0019 = N0097) for approval along with these changes:
-- reschedule the WG11 and WG15 liaison reports from item 1.9 to item 5
-- add liaison reports for SC21/WG3 and X3H2 under item 1.9
-- add item 1.10 to discuss the date for November meeting

Motion by Saks/Shopiro: "Move that we accept the proposed agenda with these additions."

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

1.7 Use of X3J16 mailing lists

Lenkov explained the X3 policy that technical committee member lists are public documents. X3 has no rules restricting the use of these lists, but considers it "good citizenship" to refrain from using the lists for commercial purposes. Committee members should notify Clamage if they want him to note on the list that they do not wish to receive "junk mail". Lenkov added that US Postal regulations prohibit using the addresses of names so noted for "junk" mailings.

1.8 Conversion to type I

Lenkov reported that X3 SPARC met in January, 1992 and approved X3J16's request to convert to a type I project (92-0016 = N0094). X3 must grant final approval for the conversion. Lenkov expected this approval by the next WG21|X3J16 meeting in July.

1.9 Liaison reports

=== WG14 (ISO C) ===

Simonsen reported that WG14 met in Milano, Italy in Dec. '91. WG14 is working on a normative addendum containing three principal parts:
1. UK's clarifications of ambiguities in the ISO C standard
2. Japanese extensions for multibyte character support
3. Danish alternate to trigraphs

He expected the addendum draft to be ready for their May, '92 meeting in Salt Lake City, UT, USA.

Johnson noted that the WG14 meeting minutes are available on the email reflectors as message x3j16-intl-114.

Simonsen also explained that WG14 refused AFNOR's "extern C" proposal because it was outside WG14's scope of work.

=== X3J11 (ANSI C) ===

Plum said that X3J11 also met in Dec. '91 in Milano. However, they did not have a quorum, so they simply adjourned and joined the WG14 meeting. The next meeting is May 13-15, '92 in Salt Lake City.

=== SC21/WG3 (SQL) ===

In response to Sykes' note to DeMorgan (WG21/LT-1), Carter recommended sending courtesy copies of C++ mailings to the SC21/WG3 (SQL) convener and rapporteur. Carter also said he will write a letter to SC21/WG3 members inviting them to participate in WG21|X3J16 working groups.

Several committee members agreed that sending a liaison to SC21/WG3 would be helpful, but we needed a volunteer. No one volunteered.

=== X3H7 (Object Information Management) ===

Fong explained that the Data Base Systems Study Group established an Object Oriented Data Base Task Group (OODBTG). Based on the recommendations in the Task Group's final report, X3 SPARC established X3H7 to develop a reference model for object technology. X3H7's first meeting will be April 1-3, 1992 at CBEMA headquarters in Washington, DC, USA. Their initial work plan is to look at other standardization activities related to objects, including programming languages, databases, and open distributed processing. Fong offered to send a copy of the OODBTG final report on request.

1.10 Meeting Date in November

Johnson (representing the OSF, the host for the November meeting) suggested rescheduling the meeting from the week of Nov. 8 to the week of Nov. 1 to obtain a lower hotel rate. No one objected.

2 WG reports

Lenkov opened the committee of whole.

2.1 Core Language WG

Koenig reported that the WG was still working on name lookup, but would try to resolve the issue at this meeting. He also hoped the WG would have time to discuss the lifetime of temporaries. When asked about handling other core language issues, he said he would rather not consider other issues until these two issues were resolved. Koenig also said a list of outstanding core languages issues will be distributed after the meeting.

2.2 Extensions WG

Stroustrup said there are about eighteen proposals for extensions, all of which either have technical merit, or are formal proposals that must be addressed, or both. Every proposal is of interest to some people, and all are system independent. He thought the committee had these alternatives:

1. decide that the largest language is the best,
2. look at each proposal and evaluate it on its merits, or
3. accept no extensions.

He noted that we haven't turned down any extensions yet. People who evaluate extensions tend to want them, and they try to make them work. This scares him.

Scian suggested that implementation experience should be a criteria for acceptance. Stroustrup said most extensions are easy to implement, so people will think implementation experience becomes a reason for acceptance. He added that if we are selective, we need criteria for rejecting proposals.

Shopiro suggested that an extension must be something that a significant class of users can't do without.

Koenig said that if implementation experience is a criteria for acceptance, then implementors will race to be the first to implement extensions. In effect, the first to implement a given extension gets to specify it. He suggested just tabling all potential extensions until the Core Language WG is largely done. Then we should look at the proposed extensions and try to frame evaluation criteria at that time.

Brück said we should apply the written criteria for submitting proposals very firmly and return those proposals that don't comply. This will reduce the number we must consider. Bruns agreed, noting that we had only recently set down written criteria. We should try Brück's approach and see what happens.

Stroustrup invited the committee to continue this discussion in the Extensions WG session.

2.3 Libraries WG

Vilot presented the Libraries WG report (92-0034 = N0111). The WG distributed their latest draft of a proposed section 18.1 Language Support (91-0126 = N0059) and got no feedback, so Vilot thought it might be ready for a vote at this meeting. Vilot also said Clamage's proposals for adding the Standard C library to the C++ standard were ready for a vote.

2.4 Environments WG

Lenkov explained that Chapin is now an observing member and no longer Environments group chair. Stone volunteered to act as chair. When only four of the committee members in attendance said they'd attend the Environments WG meeting, Lenkov asked the committee if the Environment WG's work is important. Many said yes.

2.5 Formal Syntax WG

Scian said that the WG had a proposal on orphaned non-terminals (92-0011 = N0089) ready for a vote by the full committee.

2.6 C Compatibility WG

Plum reported that:

-- The "impressionistic" list of C++ incompatibilities with C was distributed as email messages x3j16-compat-44 and -45, but he had not received any feedback.
-- About 45% of the terms from the C standard have made it into C++ draft.
-- Since the last meeting, Plum and Shopiro talked much about the precise definition of 'type'.
-- The WG was logjammed around the notion of incomplete type.

Plum said that Nelson took responsibility for the last (unassigned) item on the "impressionistic" list. However, Jackson is no longer on the committee, so the WG needed someone to assume responsibility for his list items.

3 WG sessions

Lenkov closed the committee of the whole.

The committee recessed to working groups at 14:00 Monday.

4 WG sessions

The committee reconvened at 11:08 Tuesday.

5 WG11 activities - Brian Meek

=== WG15 (POSIX) liaison report ===

Simonsen announced that the next WG15 meeting will be in May, '92 in New Zealand. He explained that WG15 developed a language independent specification for the POSIX kernel and a C binding to that kernel. They have begun developing bindings for other languages, such as Fortran and Ada. Simonsen asked if this committee wanted a C++ binding.

O'Riordan stated that the C binding is adequate for C++. Simonsen suggested that we might ask WG15 to develop a C++ binding. Plum thought we already had enough work, and we don't need another standard to track. Carter thought we should say we're interested in seeing what WG15 is doing, but we can't promise any help nor any feedback. Plum reiterated that a bad job is worse than no job.

Schwarz said the C binding should be adequate as long it doesn't use names that conflict with names used by C++ standard libraries. He offered to check the POSIX C bindings for such name conflicts.

Lenkov summarized the committee's position:
1. The POSIX C binding is adequate for now.
2. WG15 should not develop a C++ binding until the C++ standard exists.
3. WG15 should send a copy of the POSIX C binding to Schwarz so he can check it for conflicts with C++ standard library names.

=== WG20 (Internationalization) ===

Simonsen reported that WG20 is working on:
1. locales,
2. language support for ISO 10646, and
3. extended character sets for identifiers

Carter asked Simonsen to be WG21's liaison to WG20. Simonsen agreed and no one objected.

=== WG11 (Binding Techniques) ===

Brian Meek summarized the activities of WG11, focusing particularly on their work on Common Language Independent Datatypes (CLID).

Simonsen (the liaison to WG11) suggested that WG21|X3J16 might be interested in following WG11's work. No one reacted.

Carter referred members to documents 91-0138 = N0071 and 92-0029 = N0106 for more information on WG11.

=== NCEG (Numerical C Extensions Group) ===

Swan said that he did not attend the last NCEG meeting (in Plano, TX, USA in January '92), so he could not report on the outcome of that meeting. NCEG is preparing a technical report consisting of recommendations for extensions to C to consider when the ANSI C standard comes up for review in 1994. In addition to the possible extensions described in the NCEG liaison report of the previous minutes, the group is considering:

-- run-time subscript range checking as proposed by Stallman
-- low-level floating-point exception handling (different from exception handling in C++)

The next NCEG meeting is May 11-12 in Salt Lake City. Anyone interested in joining the NCEG email reflector should contact Tom MacDonald at tam@cray.com or uunet!cray!tam.

6 WG sessions

The committee recessed to working groups at 14:30 and reconvened at 13:35 on Wednesday.

7 WG progress reports

Moved to agenda item 12.

8 WG sessions

Moved to agenda item 11, deleting the General Session.

9 Working Paper for the Draft Proposed Standard

Shopiro presented the editor's report (92-0038 = N0115).

O'Riordan noted that *operator new* can be called with the argument zero, but the draft wasn't clear about the meaning of *new T[0]*. Saks pointed out that the Core Language WG report in the March '91 minutes (91-0045) states that the WG agreed "that *new T[0]* returns a unique pointer, which implies that the array may consume memory." Shopiro said he'd change the draft accordingly.

O'Riordan also said that the revised grammar in section 8.4 allows an initializer of the form *= { = { e1, e2 } }*, and he doubted this was intended. He also asked if the additional identifiers adopted in June '91 were also adopted by WG14. Simonsen said that they were, except for *not* and *not_eq*. *not* and *not_eq* were intended to cope with problems in EBCDIC, and WG14 was undecided about accommodating a non-standard character set.

Shopiro explained that he added the concept of "prototype scope" to the Working Paper. LaJoie wanted to be sure that tags introduced in a prototype don't disappear as they do in ISO C. Plum said that Shopiro's wording makes it clear that they do not.

Shopiro said his next version of the Working Paper will look more like an ISO document than an ANSI document.

Schwarz noted that since enumerations are not integral types, they are no longer allowed as bitfields, which must have integral types. He thought the description of bitfields should be changed to allow them to have enumeration types. Shopiro asked Schwarz to send him email.

Plum asked about sections numbers in the ISO document. He was concerned because the C standard sections were renumbered when it became an ISO document. Carter said that Keith Brannon at Geneva, Switzerland offered to review the document to see how well it conforms to ISO requirements (92-0032 = N0109), and that Shopiro should send him the document for review. Plum wanted a strong statement from this committee that we did not want the document renumbered. Stroustrup agreed. Carter said that JTC1, not SC22, controls such decisions, and that national delegations should make their desires known to them.

Shopiro will find out what ISO requires in document formatting and report on it at next meeting.

10 Rationale progress report

Waggoner had little to report.

11 WG sessions (rescheduled)

The committee recessed to working group sessions at 14:45 and reconvened at 08:35 on Thursday.

12 General Session

The WGs presented their progress reports.

=== Core Language ===

Koenig explained that the WG reached general agreement on rules for name lookup, and wanted to submit them for a vote.

=== Extensions ===

Stroustrup explained that the WG had no better criteria for accepting or rejecting extensions, but they did reach decisions on several proposed extensions. They decided to relax the rules on return type of virtuals and want to submit that decision for a vote. The group judged several other extensions acceptable in principle but found they fell apart on the details.

=== Libraries ===

Vilot said the WG planned to submit its ANSI C library proposal (92-0024 = N0101) for a formal vote.

=== Environment ===

Stone said the WG discussed Kearns' paper on static initialization (91-0137 = N0070). The four members in attendance verified that they agreed with Kearns' first five proposals, but not the sixth. They also agreed that they wanted to change the syntax for *#include* directives so that header names need not end in *.h*.

=== C Compatibility ===

Plum said the WG has six or seven proposals to bring to the committee. Also, the WG made headway on the definition of 'type', but they were still only about half way there.

=== Syntax ===

Scian said the group still wanted a formal vote on Scian's proposal for orphaned non-terminals.

=== End of WG progress reports ===

Lenkov opened the committee of the whole.

=== Core Language ===

Koenig explained the Core Language WG's resolutions on name lookup. He said they had three goals for the name lookup rules. The rules should:
1. be consistent with current usage,
2. be reasonably easy to implement, and
3. present no ugly surprises for users.

He gave this example

```
class X {
    ...
    f(T x) { T y; ... }
};
```

and asserted that we want the two *T*'s to mean the same thing. He added that a program in which they do not mean the same should be taken out and shot. Becker said that if the two *T*'s don't mean the same thing, the LANGUAGE should be taken out and shot.

Koenig gave a second example:

```
class X {
    ...
    f(T x) { T y; ... }  // 1
    void g() { xx = 0; } // 2
    typedef int T;       // 3
    int xx;              // 4
};
```

He said the *xx* in *//2* has always meant *X::xx* (the *xx* on *//4*). However, with the advent of nested types, the *typedef* on *//3* alters the meaning of *//1*. The problem is that the *T* in *f(T x)* on *//1* may refer to a declaration of a *T* at file scope, whereas the *T* in *f*'s body always refers to the *X::T* on *//3*. Thus the two *T*s on *//1* don't mean the same thing. Koenig said the WG wanted this example to be illegal.

Koenig showed another problem caused by C++'s "heritage from C". Consider the declaration:

```
typedef const P(Q);
```

If *P* is a type, then this declares *Q* of type *const P* and the parentheses are redundant. On the other hand, if P is not a type and *Q* is a type, this is a declaration for function *P* with an argument of type *Q* returning *[const] int*. The "heritage from C" is that declarations must always be interpreted in sequence. That is, what a given declaration declares is determined by what has been declared before. In contrast, languages such as Pascal have declarations in which it's always clear what they declare, regardless of prior context.

Koenig said the WG had been seeking the "holy grail" of name lookup -- that permuting the members of a class definition should not change the meaning of the class to another valid meaning. He also explained that many problems with the C heritage disappear if you disallow implicit *int* type specifiers or redundant parentheses in declarators. The WG considered banning implicit *int* inside class definitions and banning redundant parentheses everywhere. But it turns out that banning these features still does not quite attain the "grail".

Koenig presented the WG's first two proposed rules for name lookup:

1. The scope of a name declared in a class consists not only of the text following the name's declarator, but also of all function bodies, default arguments, and constructor initializers in that class (including such things in nested classes).

2. A name *N* used in a class *S* must refer to the same declaration when re-evaluated in its context and in the completed scope of *S*.

Plauger asked if these pathologies [in the examples] are caused by nested types. Plum said the problem is nested classes and constants, as well as types.

Koenig gave another example:

```
class s {
    void f(const T); // 1
    typedef int T;   // 2
};
```

He said rules (1) and (2) don't cover everything we might want them to. For example, if (a global) *T* is not defined before *s*, then *s::f* declared on *//1* takes an argument named *T* of type *const int*. If *T* is defined as a type before *s*, then *//1* is still not a use of *s::T* -- it's a use of *::T*. When the compiler encounters *//2*, the above rules don't invalidate *//1* because it's not a use of *s::T*. But moving *//2* above *//1* makes *f(const T)* in *//1* a use of *s::T*, meaning that *f* is a function taking an argument of type *const s::T*. Permuting the members gives the class a different valid meaning, and so the WG wanted to prohibit this class.

Koenig explained that, on the surface, it looks like the problem is the use of implicit *int*, but the following case makes it clear than implicit *int* is not the real problem:

```
class s {
    int f(int (T)) { T x; ... } // 1
    typedef int T;              // 2
};
```

If no type *T* is defined before *s*, then *//1* declares a function *s::f* that takes an argument of type *int* named *T*, and the parentheses around *T* are redundant. If you move *//2* above *//1*, then *s::f* becomes a function with an argument of type "function taking *T* returning *int*." Again, permuting members gives the class a different valid meaning.

Becker asked why not a have a reverse rewrite rule such as "the class is rewritten as if all types were moved to the front". Koenig replied that there are even instances where permuting just the *typedefs* within a class yields a new valid meaning.

Koenig then presented a third rule for name lookup:

3. If reordering member declarations in a class yields an alternate valid program under (1) and (2), the program's meaning is undefined.

He explained that the reason the behavior is undefined is because we don't know if we can detect the error in finite (Plum injected "linear") time. Koenig also explained that Turner defined the notion of quasi-use (92-0013 = N0091) to cover the two preceding examples. Turner had pro-

posed that a use or quasi-use of a name before its definition should be illegal. Koenig explained that the WG rejected the notion of quasi-use because it was too difficult to comprehend.

Koenig also explained that Rabinov suggested an alternative algorithm for name lookup that doesn't appear to change defined behaviors, but may be able to detect errors in a reasonable time. Rabinov's scheme requires two complete passes over the class definition, but only two passes. Koenig said that Rabinov claimed his algorithm yields the same results as the three rules except that it can detect undefined behavior as errors.

Plauger thought some of the problems explained by Koenig were similar to problems that arise in parsing Pascal. He asked if we could gain some insight by studying how Pascal solved these problems. Koenig said the problem is that the *typedef* syntax is such that the same sequence of tokens can declare a different thing in a different context. Pennello said that Rabinov's rules attempt to apply the Pascal rules to C++, but the WG had problems understanding the Pascal rules in a C++ context because of its "C heritage" (that what a C declaration declares is determined prior declarations).

Stroustrup said this was the clearest statement of the problem and the clearest statement of the solution that he's heard so far. He then asked if it would be illegal for a member function body inside a class definition to use a variable that's defined later in the class. Koenig said no -- the scope of a name defined inside a class includes the bodies of member functions.

Koenig further explained that the WG did not want to impose a constraint on programs that's so computationally complex that it can't be computed in tractable time. That's why they wanted to continue exploring algorithms like Rabinov's. Stroustrup and Scian preferred Koenig's statement of the rules to an algorithm. Stroustrup said an algorithm is good for implementors, but gives users the impression that the rules are very complex. O'Riordan said that although he favored the suggested rules for name lookup, he wanted to see an algorithm as well. Stroustrup said we need both, and he didn't want to see the algorithm relegated to the rationale document.

Pennello gave another example:

```
typedef int T;
struct S {
    typedef T *T; // 1
};
```

This is invalid by rule 2. When parsed, the first *T* in *//1* refers to *::T*. When re-evaluated in the completed scope, it refers to *S::T* (the second *T* defined in *//1*).

Pennello gave yet another example:

```
struct C { typedef T; };
struct A : C { };
struct B {
    A::T f() { A::T x; };    // 1
    struct A : C { };
};
```

Pennello gave this example to emphasize that every name, even in a compound *id-expression* like *A::T*, is subject to the first two rules. In particular, the *A* in *A::T* violates rule 2. In this example, the return type *A::T* in *//1* first refers to *::A::T*, but after completion of the class, it refers to *B::A::T*. Although it turns out that *B::A::T* refers to the same declaration as *::A::T*, the lookup follows two different paths. Koenig explained that the WG wanted to disallow this because different paths to the same declaration may have different access.

Koenig said there was some question about the word "undefined". He thought "undefined" behavior is an error that the implementation need not diagnose. Shopiro agreed. Plum said the definition of "undefined" is already in the draft. Stroustrup asked if he could produce a diagnostic if he wanted to. Koenig said yes.

O'Riordan noted that most undefined behavior is defined for the execution environment, but the third name lookup rule specified undefined behavior in the translation environment. Shopiro responded that diagnosing these errors is much liking diagnosing violations of the one-definition rule. Schwarz said he thought we should have different words in the draft for undefined behavior at translation-time vs. undefined behavior at run-time. Koenig said he thought the ARM's description of name lookup already uses the word "undefined". Stroustrup said his question had been answered. He added that there's a separate issue about what is the right way to describe run-time undefined behavior vs. compile-time undefined behavior, but that should be discussed as a separate issue.

Straw Vote: Who wants to transmit these three rules to the editor for incorporation into the draft? lots yes, 5 no, 1 abstain.

Koenig said the WG also discussed the lifetime of temporaries and will continue at the next meeting. Turner said that the direction the WG has taken is away from immediate destruction. Although some (including himself) liked early destruction, it was shot down badly at Nashua. Hazeghi said he thought early destruction wasn't shot down, but rather people had asked for elaboration. Koenig said it was obvious that if it had came up for a vote, it would have been voted down.

Lenkov asked if there were other core language issues to address before the next meeting. Pennello said we need to reconsider friends and injecting names into enclosing scopes in light of these new [name lookup] rules.

=== Extensions ===

Stroustrup said the WG began with a discussion of criteria for accepting or rejecting extensions, but they made little progress. He said he worries that:

1. we lack sharp criteria,
2. people are nice and prefer to say "yes" rather than "no",
3. there are lots of reasonable proposals,
4. people who go to the extensions group meetings are the people who like extensions.

Stroustrup also feared that a series of small steps that benefit the language may yield a negative sum [that's bad for the language]. He hoped he'd get help dealing with these things. He also noted that when we accept an extension, we impose at least a year of uncertainty about whether we can use it.

Stroustrup said that he had suggested at the WG meeting that people's favorite proposals weren't the same, but (at that meeting) Brück challenged him by asking "What if they are the same?" So Stroustrup asked each WG member to pick his/her two favorites, and two least favorites. It turned out that the group had two clear front runners and a couple of dead proposals on their hands. Their top concerns were:

1. run-time type identification (14 of 14 positive votes, and no negative votes)
2. name space control

Stroustrup then summarized the WG's responses to the analyses of several proposals for extensions.

Stroustrup said the WG discussed Kearns' analysis (92-0021 = N0099) of Adcock's proposal to allow overloading *operator* . (dot) (91-0140 = N0073). Stroustrup said the WG decided that the analysis had insufficient technical detail and ignored issues raised by Koenig in his earlier analysis. Knuttila will send a letter to Kearns asking if he'll write a better proposal.

Stroustrup summarized the WG's discussion of the proposal for free store operators specifically devoted to arrays of objects. The latest analysis (92-0012 = N0090) found the original proposal (91-0124 = N0057) insufficient and proposed an alternative. The WG decided this alternative was also insufficient, so the authors of both documents (Howard, Gibbons and Holly) will try again with a paper for the next meeting.

Stroustrup explained the WG's decision on keyword arguments (originally proposed in 91-0127 = N0060 and analyzed in 92-0010 = N0088). The WG first considered if the proposal was technically sound and complete. It was, except maybe for some nits. Next they asked if it was needed, if it will help people, and what problems will it cause? Stroustrup said the WG found the extension had unexpected impact on existing programs and probably on future programming styles.

Stroustrup explained that keyword arguments make a function's formal argument(s) part of its interface. Keyword arguments increase the binding between users and libraries. The author of function must promise not to change the formal argument names, since such a change would impact users of keyword arguments. Also, some libraries have formal argument names in headers that are long for descriptive purposes, but use short names, like *x*, in the implementation.

Stroustrup asked if we wanted to encourage users to create long argument lists. Pennello asked if anyone knows what the Ada people think of this feature. Stroustrup said that any feature tends to be touted for its benefits. If people get burned by it, they usually keep quiet. Koch noted that Ada experience may not be applicable to C++ because Ada was designed from the beginning with keyword arguments.

Stroustrup mentioned that it's generally accepted that programs in object-oriented languages tend to have shorter argument lists than do programs in procedural programming languages. The experience of WG members confirmed this. This makes keyword arguments noticeably less useful in an object-oriented language.

Stroustrup said that Hartinger (a co-author of the original proposal) withdrew the proposal and will write a final analysis to document the decision.

Stroustrup then discussed the proposal to relax the restrictions on the return type of virtual functions (originally proposed in 90-0049, and analyzed in 91-0051 and 92-0004 = N0082). He summarized the basic proposal with this example:

```
class X {
    virtual A f();
};
class Y {
    B f();
};
```

He said that the current draft required that *A* and *B* be the same type. The proposal was to relax this restriction in the following ways:

	A =	B =	where
1.	Z*	ZZ*	ZZ is derived from Z
2.	Z&	ZZ&	ZZ is derived from Z

The WG also considered and rejected the following alternatives:

	A =	B =	where
3.	Z	ZZ	ZZ is derived from Z
4.	C	CC	CC has an operator C()

Stroustrup said that Goldstein noted (in the WG meeting) that 90% of all his casts were to get around the restriction to identical return types.

Stroustrup explained that the existing rule was:

> "It is an error for a derived class function to differ from a base class virtual function in its return type only."

The WG recommended adding the following at the end of the sentence:

> "unless the return types are either both pointers to classes or both references to classes and the class in the original return type is an accessible base class of the class in the new return type."

Scian asked why the WG didn't recommend a similar relaxation for argument types. Stroustrup responded that such a relaxation would not be type safe, and that expensive run-time checking of arguments would be required. He added that Bruns and Lenkov will write a revised analysis of this proposal, including why we don't relax argument types.

O'Riordan supported this extension based on his earlier analysis of the implementation impact. He also said this extension poses no problems for functions returning pointers to members, and that no additional rules are needed. He gave the following example to show that relaxing the compatibility rules for functions returning pointers to members permits type safety violations:

```
class X { };

class Y : public X {
public:
    double d;
};

class A {
public:
    virtual X *f();  // 1
    X *g();
};

class B : public A {
public:
    Y *f();          // 2
};

X *(A::*pma)();
Y *(B::*pmb)();

X x;
X *A::g() { return &x; }
pma = &A::g;
pmb = pma;           // 3
B b;
(b.*pmb)()->d = 8.7; // 4
```

O'Riordan explained that the proposed extension to relax the return type of virtual functions allows the declaration of *Y::f* on *//2* to override the declaration of *X::f* on *//1*. This poses no safety problems. However, allowing the assignment (and implicit conversion) on *//3* allows the unsafe assignment on *//4*. The problem is that the actual value of *pmb* is *&A::g*. The expression *(B.*pmb)()* in *//4* looks like it returns a *Y ** (a pointer to a derived class object), but it really returns an *X ** (a pointer to a base class object). Thus, the expression *(b.*pmb)()->d* in *//4* refers to a non-existent data member.

Koenig also cited a paper by Luca Cardelli proving that the proposed relaxation is theoretically safe.

Stroustrup summarized that WG's decisions on several other minor issues:

-- operator overloading on enumerations (91-0139 = N0072): O'Riordan and Hoffman will write an analysis for the next meeting.

-- Snyder's encapsulation 'loophole' (91-0144 = N0077): Stroustrup said it wasn't a loophole at all. It was a request for a finer-grained protection system. Eckel will write a polite rejection letter.

-- access to protected members (91-0145 = N0078): The proposed relaxation of the protection rules would violate guarantees the WG wanted to keep. Eckel will write a polite rejection letter.

-- hidden private members (91-0146 = N0079): The WG rejected this proposal because the language lets you express something similar using abstract base classes, and because the proposal described implementation details that do not belong in a standard. O'Riordan will write a polite rejection letter.

The committee discussed whether response letters should be separately numbered documents. Stroustrup said the WG wasn't planning to submit the letters as numbered documents. Schwarz wanted each response to be individually numbered. Saks suggested that each letter need not be a document, but a summary of the reasons for rejection should be combined into a single numbered document. Plum agreed with Saks, adding that the thank you letters should be based on a form letter. Lenkov said the response letters must be from individuals, not from the committee, unless we want to take votes on each letter.

Stroustrup said the WG didn't have time to deal with the paper on *~const*. O'Riordan offered to help Knuttila write an analysis of this issue.

Stroustrup said the WG spent much time on run-time type identification. Again, they asked "Is it needed?" and "Is it technically sound?" Stroustrup said there are three parts to the proposed extension (92-0028 = N0105):

1. checked cast -- This is the ability to say "I've got a pointer to this object. Can I convert it to this type?"

2. type identity -- In the WG meeting, Lenkov and O'Riordan gave reasons why this is essential. Stroustrup and Goldstein agreed, but were nervous about the impact on programming style. Stroustrup said it encourages people to write *switch* statements on types.

3. *type_info* object -- These are objects that give added information about a type. About 60% of the WG members were already faking this feature in real programs. Stroustrup said we need a mechanism that allows users and tools to associate their own type information with the *type_info* operator. The operator should return a 'cookie' that allows users to find their own type information.

Several people asked questions attempting to get more detail, but Stroustrup said the WG was not at that level of detail yet.

Stroustrup presented several different notations for checked casts under consideration:

```
ptr_cast(D, p)  // looks macro-ish, gives tools trouble
ptr_cast<D>(p)  // uses template notation to segregate type
cast<D*>(p)
(D*)p
<D*>p
(virtual D*)p
```

Stroustrup said that currently there are two kinds of casts:
1. those that don't change the bit pattern, and
2. those that produce a value of a new type.

The WG could not decide if they should introduce a new cast notation. They looked at the impact on existing code if they reinterpret *(D*)p* as a checked cast, and found that it introduces run-time overhead.

Stroustrup said Lenkov and he will write a new paper discussing these issues.

Schwarz asked if the WG had considered using a type switch. Stroustrup said yes, but he doesn't like it because it invites programmers to write horrible code.

Stroustrup explained that virtual function return type relaxation and checked casts combine to produce a safe results. Checked casts convert arguments to the right type, and the relaxation insures that returned values have the right type.

Schwarz asked if the safe cast can be applied to things like casting *long* to *int* for use in, say, templates. Stroustrup said the idea has merit, and he will explore it.

Stroustrup briefly discussed the open issues for name space control (92-0008 = N0086). He gave this example:

```
::A::{ f(); }
::B::{ f(int); }

void g() { f(99); }
```

The question is whether to do name space resolution before overload resolution, in which case *f(99)* is ambiguous, or to do overload resolution before name space resolution, in which case *f(99)* is legal.

Stroustrup said the WG looked at several different syntactic forms:

```
::A::{ ... }        // the proposed syntax
extern A { ... }    // an alternative
A { ... }           // another alternative that's too clever by half
```

He also said that O'Riordan found an ambiguity in the syntax presented by the original proposal. The WG discussed notions like import directives, modules, packages and versioning, and decided that they were not keen on the C, Modula-2 or Ada solutions to the name space problem. Hartinger, Brück, O'Riordan and others will write a new analysis of the name space control issues.

Stroustrup announced that Simonsen and he will write a paper on extended character sets for the next meeting.

Straw vote: Who favors the proposed relaxation of the rule for return types of virtual functions? lots yes, 0 no, 0 abstain.

=== Libraries ===

Vilot began by saying that the WG recommended adding the standard C library (92-0024 = N0101) to the working paper. He then said the specification of the language support library (proposed to be section 18.1 of the working paper) needs more work. Language support includes *operator new()* and *delete()*, as well as *set_new_handler*, and the default behavior of the *new* handler. The WG identified three issues:

1. the semantics of new handlers -- what they are obliged to do or not do.

2. the use of the stack handler template -- it seemed gratuitous.

3. the description of *operator new()* and *delete()* -- they must be described differently than other library functions because programmers can replace them. The descriptions of *new* and *delete* are obligations on the programmer, not on the implementor.

Vilot said the WG wanted to require that *new* expressions throw an *xalloc* on allocation failure, so they never return a null pointer. He added that this would be a change in the working paper. Yaker asked why. Vilot said that without this change, programmers must always check for both a null return and for an exception.

Stroustrup said the change seems drastic -- lots of people simply convert *malloc* calls to *new* calls and expect to be able to check for a null return value. O'Riordan said that small machines often run out of memory, and so running out of memory should not be an exception. He noted that we [the committee] have already said "no" to resumable exceptions, and now we're considering getting rid of the resumption capability provided by *set_new_handler*.

Several members spoke in favor of throwing an exception and not returning null. Koenig said that most programs either do not check for a null return, or check and terminate the program. He suggested a way to solve O'Riordan's problem by having the *new* handler either set another handler or reset itself.

O'Riordan said you could flip Koenig's argument around to say you should set a handler that throws an exception. He also asked what we plan to do about user-defined *operator new()* with placement. Will we insert exception handling code into all of those? Stroustrup suggested that not returning null is a radical solution that's close to an extension. He wanted to see a paper that analyzes the issues.

Vilot asked for volunteers to write such a paper. LaJoie, O'Riordan and Schwarz volunteered.

Vilot then discussed open issues in the input/output library. He said the paper on multibyte character handling (92-0039 = N0116) enlightened the WG to issues they had missed, and they need to work more in this area. Vilot appealed for people with experience in *iostreams* and multibyte characters to help Schwarz with his proposal for the i/o library.

Vilot said the WG wanted to be able to interleave puts to *stdout* and *cout*. Becker added that i/o can be interleaved only for the standard streams. That is, input from *stdin* can be interleaved with input from *cin*, and output to *stdout* (*stderr*) can be interleaved with output to *cout* (*cerr*). Input from/output to other streams may not be interleaved.

Vilot explained the WG's proposal to add the standard C library to the working paper. WG decided to let Shopiro (the editor) decide whether to include the complete text for the standard C library or refer to section numbers in the ISO standard. They simply proposed to approve (92-0024 = N0101) for Shopiro to add to the working paper as appropriate. Vilot noted that the document says nothing about the interaction of exceptions and signals.

Clamage said there were nine editorial changes to 92-0024 = N0101:

1. change ANSI to ISO everywhere
2. on page 3, line 11: delete "in both ... spaces"
3. page 5, wordsmith the top paragraph
4. page 6, leave the top paragraph unchanged from the C standard
5. page 8, top paragraph - add "local objects which would have been destroyed is undefined"

Koenig asked why the proposal didn't say *longjmp* should be implemented with *throw*. Vilot said they didn't want to make a gratuitous change and they didn't want to force implementors to change their implementations. Koenig and Bruns argued that this forces an implementation to supply *longjmp* but not to make it work. Vilot replied that *longjmp* will work, just as it always did.

Koenig wondered if the WG considered making *longjmp* an anachronism. Stroustrup said C++ programmers expect it to work as it does now, and we shouldn't change it. Schwarz noted that much of the C library is an anachronism, but we aren't labelling it as such. Plum said it's not a good political move to label programming in style that supports C as anachronistic. He added that SC22 charged us with minimizing differences between C and C++ unless we have a good reason to introduce incompatibilities.

Clamage continued with his list of editorial changes to 92-0024 = N0101:

6. page 9 - delete the last three paragraphs and the footnotes, except for the first sentence of the first of those paragraphs
7. page 10 - restore the description of *abort* to as it was in the C standard
8. pages 11-14 - tighten up the wording of the string functions
9. page 14 - make the future library directions the same as the C standard

Straw vote: Who wants to include 92-0024 = N0101 with the listed changes as part of the draft? lots yes, 0 no, 0 abstain.

Vilot discussed the WG's thinking about string classes. He said they are leaning toward specifying value semantics. He also said they are not sure what to do about temporaries, so they are hedging their bets by considering two conversions to *char **:

1. that stores the text in a user-defined buffer, and
2. that places the text in a buffer that lasts to the next conversion to *char **.

The WG is also looking at the effects of locales on string comparisons.

Koenig argued that implicit conversion to *char ** is always dangerous. Schwarz said the WG decided not to do implicit conversions.

Vilot explained that the WG is trying to strike a balance between specifying too many overloaded versions of catenation operators and specifying too few. For example, should the library include both a *cat* operator that accepts a string and one that accepts a *char ** so that users can be sure they will get an efficient implementation? Koenig said it's not just a matter of efficiency. If the library provides several *cats* that accept different types, and another class *X* has conversions to more than one of those types, then calling *cat* for an *X* object is an ambiguity. Koenig advised members to look at Rob Murray's article on building C++ libraries with well-defined type relationships. Stroustrup encouraged building libraries with a variety of overloadings for efficiency. He also said that anything that discourages too many user-defined conversions is good.

Vilot continued his description of the string library issues. He said the WG wanted to allow 0x00 in the middle of a string. The 0x00 only affects the value of a string when the string is converted to *char **.

Vilot briefly summarized the WG's work on container classes. He said that Steinmuller's proposal for array classes includes two template classes: *DynArray<T>* and *PointerDynArray<T>*. They each provide an *operator[]* that returns a reference. In both classes, *operator[]* can't tell if it's being used as lvalue or rvalue. Vilot invited comment.

Vilot said that Allison has been working on a bitset class. The WG decided not to produce an abstract set type with operations such as intersection, union, etc. Allison is now working on classes that support fixed-sized and dynamic-sized bit strings.

Vilot listed the WG's future work:

-- revise the proposals for language support, i/o, strings and containers
-- signals and exceptions
-- *wchar_t* and *size_t* -- should they be "distinct types" or wrapper classes?
-- Does redefining global *new* and *delete* replace all uses of the implementation's *new* and *delete*, i.e., must the library use the user's global *new* and *delete*? When does the binding occur?
-- *stdarg* processing of C++ types
-- name space issues

=== Environments ===

Stone reported that the Environments WG discussed the order of static initialization. He said the WG members agreed with five of the proposals suggested by Kearns (91-0137 = N0070). Stone reviewed the proposals.

Several people noted problems with the first proposal. O'Riordan said this proposal leaves the value of non-class objects undefined. Schwarz said that the proposal contradicts itself by saying that objects without a constructor get initialized by a generated default constructor.

O'Riordan said we should strike the phrase "or an object that can be brace initialized" from the third proposal.

Turner asked if the fifth proposal changes the rules for temporaries generated to initialize references. Yaker said the intent of the text is that there is a logical "block" around all static initialization so that temporaries created during static initialization are destroyed by the same (as yet to be determined) rules for destroying temporaries in any other block.

Stone reviewed Kearns' alternative proposals 6A through 6E. He said the WG had not reached agreement on these proposals. Turner asked why the WG didn't like 6C. Yaker replied that they wanted to avoid a language

extension, especially one that raised the likelihood of programmer error. Becker said that if the problem is serious enough, then the WG shouldn't back away from proposing extension.

Schwarz asked if the WG was in agreement that all initialization occurs before entering *main*. Stone said "not really". Lenkov said that another paper by Wilkinson addresses this issue.

O'Riordan saw a problem with the suggestion in Kearns' paper that the translator could determine the initialization order by global static analysis at link time. He said that device drivers sometimes have an initialization object so that the constructor call for that object initializes the device, and the destructor call shuts down the device. This can't be caught by static analysis.

Someone observed that you can handle some cases of mutual dependencies between translation units by analyzing the dependencies between individual objects rather than between entire units. However, the consequence is that some objects within a translation unit may not be initialized in declaration order (as currently specified). Yaker responded that the WG wanted to avoid confusion of having initialization order different from declaration order within a unit. Becker said the Borland compiler has facilities for specifying the order of initialization, but he suspected people aren't confused by them because they probably don't use them.

Schwarz thought Kearns' third proposal was ready for a vote. O'Riordan said it would be if we deleted the phrase about "brace initialization". Koenig said that we should not vote on just one aspect of the solution, because it might unduly constrain future attempts to solve related problems.

Straw vote: Who agrees with Koenig? lots yes, 5 no.

Yaker, Shopiro, O'Riordan, Stone, Rafter, Becker, Swan and Koch volunteered to work further on the order of static initialization.

Stone presented other environments issues. The WG suggested that a strictly-conforming program be allowed to omit the *.h* in header file names. They also suggested increasing the header name length beyond six.

Vilot wanted to be sure that the name of the header in an include directive need not be the name of a file. Koenig replied that nowhere in the C standard does it say that a header name must name a file. It simply says that including a header defines a set of names.

Lenkov called for straw vote on this proposal. Stone said the question for vote is "will *‹file.h›* and *‹file›* be equivalent?" Many said that's not what they understood. Plauger said there are two issues:

1. The C standard says you can write any of 15 different names between ‹ › in an include directive and something magical happens.

2. If you want to write a portable program, then use six letters, a dot, and an 'h' for the header name, and no operating system will reject it.

Bruns noted that a header can be six letters, a dot, and then any letter (not just 'h'). Edelson said the proposal was simply to drop the requirement for the dot and the following letter in a header name.

After more discussion, Lenkov suggested that the issue was not ready for a vote. He summarized who was working on what:

-- Turner, Johnson and Rabinov offered to work on a paper on the one definition rule.
-- Swan and Stone will work on translation limits.
-- Hartinger, Yaker and Schwarz will work on mixed C/C++ environments.

=== Syntax ===

Scian presented the Syntax WG's proposal on orphaned non-terminals (92-0011 = N0089). The proposal suggested replacing certain prose in the working paper with grammar rules. Shopiro and Stroustrup suggested adding the grammar rules but not deleting the prose.

O'Riordan noted that *template-name* needs to have a rule that connects it to *identifier*.

Who favors adding the grammar rules: lots yes, 0 no, 0 abstain.
Who favors deleting the prose: 2 yes, lots no, 10 abstain.

Lenkov suggested that this need not be handled as a formal motion -- Shopiro could simply incorporate the changes. No one objected.

Lenkov closed the committee of the whole.

The committee recessed at 17:35 Thursday and reconvened at 09:00 on Friday.

13 General Session

Motion by O'Riordan/Scian: "Move that we direct the editor to:

1. add to the end of section 10.2 paragraph 1 sentence 4 in the Working Paper:

   unless the return types are either both pointers to classes or both references to classes, and the class in the base class function's return type is an accessible base class of the class in the derived class function's return type.

2. and adjust the surrounding text and examples accordingly."

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no.

Motion by Plum/O'Riordan: "Move we accept the current Working Paper (N0101 = 92-0023) with the following corrections:

1. change references to 'integral types' into 'integral and enumeration types' where appropriate
2. fix the grammar for initializers."

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 6 yes, 0 no.

Motion by Pennello/Charney: "Move that we adopt the following name lookup rules for incorporation into the Working Paper, with appropriate editorial modifications:

1. The scope of a name declared in a class consists not only of the text following the name's declarator, but also of all function bodies, default arguments, and constructor initializers in that class (including such things in nested classes).

2. A name N used in a class S must refer to the same declaration when re-evaluated in its context and in the completed scope of S.

3. If reordering member declarations in a class yields an alternate valid program under (1) and (2), the program's meaning is undefined."

Hartinger suggested that the editor add examples to illustrate the third rule because it's so complex.

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 6 yes, 0 no.

Motion by Vilot/Landauer: "Move that we add the contents of N0101 = 92-0024 to the Working Paper, with the following changes and appropriate editorial modifications:

1. change 'ANSI' to 'ISO' where referenced in the paper.
2. delete the end of line 11, page 3 'in both ... name spaces.'
3. make the top paragraph on page 6 unchanged from the ISO C standard.
4. add to the *longjmp* description '... local objects which would have been destroyed is undefined'
5. delete the last three paragraphs, except the first sentence, from the bottom of page 9.
6. make the description of *abort* the same as in the ISO C standard.
7. make the description of 'Future Library Directions' the same as in the ISO C standard."

Edelson: shouldn't "undefined" be "unspecified" in 4? Schwarz and others said no.

Motion passed X3J16: 30 yes, 1 no.
Motion passed WG21: 6 yes, 0 no.

Lenkov opened the committee of the whole.

=== C Compatibility ===

Plum presented the work of the C Compatibility WG. He said Johnson was working on conformance issues, such as required diagnostics. He also thanked Shopiro for his careful work in specifying the type system.

Plum described the type description scheme developed by Shopiro and ratified by the WG:

```
DECL          -> TYPE
------------------
class X x;    -> type of x is ``X''

X *p;         -> type of p is ``pointer to X''

int f(X[]);   -> type of f is ``function of pointer to X returning
                 int''
```

Plum said that the last example requires applying a type transform shared with C -- that an argument of type ``array of *T*'' becomes ``pointer to *T*''.

Plum then turned to consider the type of

```
struct { int i; } y; -> type of y is ``...''
```

He said that he once proposed that the type of *y* is ``fabricated tag name'', but that has changed. He asked the committee to bear with him while he filled in the ``...''. He noted that implementations differ over the type of a tagless struct, such as

```
typedef struct { int i; } *PY, Y;
```

Some find *PY* and somehow associate it with a made-up tag, while others look for the first (leftmost) plain identifier, *Y*, and say it's the tag.

Plum presented several transformations involving arrays of unknown dimension:

```
int g(Y[]);       -> type of g is ``function of pointer to ...
                     returning int''

typedef int A[]; -> type A is defined as ``array of unknown
                     dimension of int''

int h(A);         -> type of h is ...
                  -> ``function of A returning int''
                  -> ``function of array of unknown dimension of int
                     returning int''
                  -> ``function of pointer to int returning int''
```

Plum said the compiler must transform *typedef* names while parsing the declarator. Pennello thought that it should be said in one place that the type of *typedef-name A* is not *A* but the type that it stands for. Then transforming *typedef*s while parsing declarators is not a special case. Plum agreed in general.

O'Riordan said that a *typedef-name* that names a class is a class name, and that the Core Language WG had agreed that the first *typedef* name that names a class is a class name. Thus, for example, in

```
typedef struct { int i; } *PY, Y, Z, W;
```

Y is the class name.

Scian asked what is the type of *A* in

```
int h(A const);
```

Plum said there is not yet a rule stating that *A const* gets converted to its canonical form ``*const A*''. Another unstated rule might be that ``*const* array of *T*'' is ``array of *const T*''.

Vilot asked Plum how the C Compatibility WG planned to fill in the dots in

```
struct { int i; } y; -> type of y is ``...''
```

and what they planned to do about tagless classes. Plum restated his opening remark that there were two alternatives for tagless classes:

1. fabricate a tag, or
2. prohibit tagless classes.

Plum said the WG recommended the second alternative. Jaws dropped. Stroustrup asked if the WG plans to change its name to the "C Incompatibility" group.

Plum said the committee can create an incompatibility with C when there is a clear and convincing reason to justify it. Putting tags everywhere makes it much easier to write a program that's both C and C++. If you don't [put the tags everywhere], you get all sorts of surprises.

Scian asked about the type of anonymous unions. Plum said they are not types.

Schwarz noted that C and C++ compatibility doesn't mean writing code that's both C and C++. He wants to mix C and C++. He doesn't want to rewrite C headers to use them with C++ code. Plum replied that all your compiler need do is produce a warning when it accepts a tagless *struct*. Plum thought trying to achieve compatibility without any changes is too strict. The discussion continued for a while with no consensus.

Lenkov relinquished the chair to Clamage.

Plum reported that the WG wanted to incorporate missing parts of the C lexical grammar C++. He suggested that the C++ lexical grammar should be the same as C. No one objected.

Plum proposed to define some more terms in the Working Paper:

	C	C++
1)	function	function [type]
2)	object	object [type] of known size
3)	incomplete	object [type] of unknown size

(Plum inserted [type] in red during the discussion.)

Plum said these terms identify three categories of type. He gave this example:

```
class X;  // now X is known to be a type
X *p;     // p is of type ``pointer to X''
```

Plum said "unknown size" is just an abbreviation for a whole bunch of unknown stuff. Several members questioned the use of "known" vs. "unknown". Some were confused about whether Plum was talking about objects or types.

Plauger noted that K & R [the ANSI C base document] was cavalier about saying "object" when it meant "object type". He said he tried writing the C standard in terms of objects of known and unknown size instead of object and incomplete type. He said you can do it either way, but you must debug the document to make sure it uses the words consistently. The C standard has already been debugged, and he'd hate to have to do it over again for C++ just because we chose different words.

Stroustrup didn't like the choice of type category names and he thought they needed work. He noted that we often say ``pointer to *int*'' when we mean ``pointer to object of type *int*''. We could say somewhere near the front of the draft that the short form is what we use uniformly when we mean the longer.

Clamage said we should continue this discussion on the *compat* email reflector. Plum said he needed some terms to convey these categories to continue his work. Stroustrup echoed Schwarz's suggestion that we adopt terms like FT (function type), COT (complete object type) and IOT (incomplete object type) for the time being.

Straw vote: Who wants to use an acronym for the each type category? lots yes, 1 no, 1 abstain.

Plum summarized other pending proposals:

1. make ``function taking pointer to array of known dimension'' incompatible with ``function taking pointer to array of unknown dimension'', e.g.,

```
extern int f((*p)[N]);
extern int f((*p)[ ]);
```

because this is hard to understand in light of overloading.

2. make ``pointer to array of known dimension'' incompatible with ``pointer to array of unknown dimension'', e.g.

```
extern int (*p)[N];
extern int (*p)[ ];
```

3. make ``pointer to array of unknown dimension'' invalid (except maybe as an expression?).

Plum noted that these proposals are increasingly stricter. That is, (2) subsumes (1), and (3) subsumes (2).

O'Riordan thought that *(*p)[]* was a hole in the type system. It allows you to pass a two-dimensional array and simultaneously lose both dimensions. Plauger said it's a wart, but a necessary side-effect of some necessary things. O'Riordan claimed that *a[][]* is equivalent to *a(*p)[]*.

Schwarz asked what Plum meant by using a ``pointer to array of unknown dimension'' as an expression. Plum said you can use the address of an *extern* array of unknown dimension in an expression. Schwarz disagreed.

14 New business (if any)

None.

15 Review of the meeting

15.1 Review of decisions made and documents approved

See Appendix B.

15.2 Review of action items

Clamage asked for questions on action items. There were none.

16 Schedule of mailings and volunteers to help

Clamage drew attention to (N0108 = 92-0031) for important dates. He noted that the Boston dates are all moved a week earlier.

17 Plans for the future

17.1 Agenda items for the next meeting

Plum asked for scheduled time for national delegations to caucus sometime during the week.

17.2 Technical sessions for the next meeting

Vilot said that if the strings proposal is ready, he'll want a technical session on it. Steinmuller offered to lead the session.

Becker noted that the strings proposal has two parts. The second part is about international character handling. He suggested holding a technical session on this.

17.3 Day Schedule for the Toronto Meeting

Clamage said it would be the same as for the London meeting.

17.4 Confirming hosts for the next three meetings

Clamage closed the committee of the whole.

Motion by Holly/Brück: "Move we accept the amended schedule for the next three meetings (N0108 = 92-0031)."

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

17.5 Call for hosts for meetings in 1993

No action taken.

18 Adjournment

Motion by Knuttila/Goldstein: "Move that we thank Neil Martin of BSI and Mike Cote of Symantec for hosting the meeting, as well as the BSI staff for delicious coffee, tea, and biscuits."

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

Motion by Brück/Allison: "Move we adjourn."

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

The committee adjourned at 12:00.

Appendix A - Attendance

Name	Affiliation	Status	M	Tu	W	Th	F
Rabinov, Arkady	Apple Computer	P	V	V	V	V	V
Koenig, Andrew	AT&T Bell Labs	S	A	A	A	A	A
Stroustrup, Bjarne	AT&T Bell Labs	S	A	A	A	A	A
Shopiro, Jonathan	AT&T/USL/NCR	P	V	V	V	V	V
Carter, Steve	Bellcore	A	A	A	A	A	A
Krohn, Eric	Bellcore	P	V	V	V	V	V
Becker, Pete	Borland International	P	V	V	V	V	V
DeMorgan, Richard	BSI IST 5/21	A	A				A
Hill, Alec	BSI IST 5/21	O	A	A			
Rafter, Mark	BSI IST 5/21	P	V	V	V	V	V
Swan, Randall	C-Team	P	V	V	V	V	V
Bruns, John	Chicago Research & Trading	P	V	V	V	V	V
Druker, Samuel	Cognex	P	A	A	A	A	A
Comeau, Greg	Comeau Computing	P	V	V	V	V	V
Holly, Mike	Cray Research	P	V	V	V	V	V
Glassborow, Francis	CUG(UK)	O	A	A	A	A	A
Allison, Chuck	DECUS	P	V	V	V	V	V
Winder, Wayne	Digital Equipment	P	V	V	V	V	V
Simonsen, Keld	DKUUG/DS	P	V	V	V		
Adamczyk, Steve	Edison Design Group	O	A	A	A	A	A
Anderson, Mike	Edison Design Group	O	A	A	A	A	A
Lenkov, Dmitry	Hewlett-Packard	P	A	A	A	A	A
Mehta, Michey	Hewlett-Packard	A	V	V	V	V	V
Knuttila, Kim	IBM	P	V	V	V	V	V
Lajoie, Josee	IBM	A	A	A	A	A	A
Banahan, Mike	Instruction Set	O			A	A	A
Jones, Derek	Knowledge Software	A	A				
Munch, Max	Lex Hack & Associates	O	A			A	
Turner, Prescott	Liant Software (LPI)	A	V	V	V	V	V
Hazeghi, Sassan	Lucid	P	V	V	V	V	V
Schwarz, Jerry	Lucid	A	A	A	A	A	A
Brück, Dag	Lund Inst. of Tech.	P	V	V	V	V	V
Yaker, Laura	Mentor Graphics	P	V	V	V	V	V
Pennello, Tom	MetaWare	P	V	V	V	V	V
Gray, Jan	Microsoft	A	V	A	A		
O'Riordan, Martin	Microsoft	P	A	V	V	V	V
Fong, Elizabeth	NIST	A	A				
Hashimoto, Tatsunori	NTT	O	A	A	A	A	A
Vilot, Mike	ObjectWare	P	V	V	V	V	V
Johnson, Andy	Open Software Foundation	P	V	V	V	V	V
Edelson, Daniel	Perennial	O	A	A	A	A	
Stone, Paul	Perennial	P	V	V	V	V	V
Plum, Thomas	Plum Hall	P	V	V	V	V	V
Charney, Reg	Program Conversions	P	V	V	V	V	V
Eckel, Bruce	Revolution 2	P	V	V	V		
Saks, Dan	Saks & Associates	P	V	V	V	V	V
Budge, Kent	Sandia Nat'l Lab	O	A	A	A	A	
Koch, Gavin	SAS Institute	P	V	V	V	V	V
Buschmann, Frank	Siemens AG	A	A	A	A	A	A
Kiefer, Konrad	Siemens AG	P	V	V	V	V	V

Hartinger, Roland	Siemens Nixdorf	P	V	V	V	V	V
Steinmueller, Uwe	Siemens Nixdorf	A	A	A	A	A	A
Goldstein, Ted	Sun Microsystems	A	A	A			A
Landauer, Doug	Sun Microsystems	P	V	V	V	V	V
Cote, Michael	Symantec	P	V	V	V	V	
Hoffman, Jane	Symphony Software	A	A	A	A	A	
Hoffman, Jeff	Symphony Software	P	A	A	A	A	
Gibbons, Bill	Taligent	P	V	V	V	V	
Clamage, Steve	TauMetric	A	V	V	V	V	V
Dodgson, David	Unisys	P	V	V	V	V	V
Gautron, Philippe	Univ. de Paris VI	P	V	V	V	V	V
Waggoner, Susan	US West	P	V	V	V	V	V
Rumsby, Steve	Warwick University	O	A	A	A	A	A
Scian, Anthony	Watcom	P	V	V	V	V	V
Welch, James	Watcom	A	A	A	A	A	A
Plauger, P. J.		P	A	A	A	A	A
Total Attendance			65	61	60	58	53
Total Voting Members			38	38	38	36	34

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

Appendix B - Motions Passed

1. Motion by Saks/Bruck: "Move we approve the minutes from the previous meeting with these corrections:

   1. on page 5, in the first paragraph, change

      '-- return type of virtuals (O'Riordan offered to write a paper on this, but it's not yet available)'

      to

      '-- return type of virtuals (91-0051)'

   2. on page 12, in the sentence beginning 'Schwarz noted ...', change '*const*' to 'non-*const*'.

   3. on page 15, in the sentence beginning 'Saks said ...', add an ' (apostrophe) after 'individuals'.

   4. on page 17, in the last sentence of the paragraph beginning with 'Koenig offered another way...', change 'on' to 'one'."

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

2. Motion by Saks/Shopiro: "Move that we accept the proposed agenda with these additions:

   -- reschedule the WG11 and WG15 liaison reports from item 1.9 to item 5
   -- add liaison reports for SC21/WG3 and X3H2 under item 1.9
   -- add item 1.10 to discuss the date for November meeting"

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

3. Motion by O'Riordan/Scian: "Move that we direct the editor to:

   1. add to the end of section 10.2 paragraph 1 sentence 4:

      unless the return types are either both pointers to classes or both references to classes, and the class in the base class function's return type is an accessible base class of the class in the derived class function's return type.

   2. and adjust the surrounding text and examples accordingly."

   Motion passed X3J16: 30 yes, 0 no.
   Motion passed WG21: 6 yes, 0 no.

4. Motion by Pennello/Charney: "Move that we adopt the following name lookup rules for incorporation in the Working Paper, with appropriate editorial modifications:

1. The scope of a name declared in a class consists not only of the text following the name's declarator, but also of all function bodies, default arguments, and constructor initializers in that class (including such things in nested classes).

2. A name N used in a class S must refer to the same declaration when re-evaluated in its context and in the completed scope of S.

3. If reordering member declarations in a class yields an alternate valid program under (1) and (2), the program's meaning is undefined."

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 6 yes, 0 no.

5. Motion by Vilot/Landauer: "Move that we add the contents of N0101 = 92-0024 to the Working Paper, with the following changes and appropriate editorial modifications:

1. change 'ANSI' to 'ISO' where referenced in the paper.
2. delete the end of line 11, page 3 'in both ... name spaces.'
3. make the top paragraph on page 6 unchanged from the ISO C standard.
4. add to the *longjmp* description '... local objects which would have been destroyed is undefined'
5. delete the last three paragraphs, except the first sentence, from the bottom of page 9.
6. make the description of *abort* the same as in the ISO C standard.
7. make the description of 'Future Library Directions' the same as in the ISO C standard."

Motion passed X3J16: 30 yes, 1 no.
Motion passed WG21: 6 yes, 0 no.

6. Motion by Holly/Bruck: "Move we accept the amended schedule for the next three meetings (N0108 = 92-0031)."

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