Minutes of Plano WG14/X3J11 Meeting
5-9 December 1994

Harvey Hotel
1600 North Central Expressway
Plano, Texas

Attendees

John Benito, Fred Crigger, Frank Farance, Ron Guilmette, Doug Gwyn, Rex Jaeschke (X3J11 Chair), Larry Jones, David Keaton, John Kwan, Tom MacDonald (X3J11 Vice Chair), Neil Martin, Randy Meyer, Randy Meyers, Dave Mooney, P.J. Plauger (WG14 Convenor), Tom Plum (Retiring X3J11 Vice Chair), Edward Ream, Linda Stanberry (Retiring Secretary), Jim Thomas, Fred Tydeman, Douglas Walls, Ted Van Sickle, Jonathan Ziebell.

Legend

The following symbols in the left margin of these minutes have the indicated meaning:

A	General approval
SV	Straw vote
MSP	Moved, seconded, passed (formal vote)
MSN	Moved, seconded, not passed (formal vote)
***	Action item

The activities reported here are grouped by subject and do not necessarily follow the exact chronological order of presentation during the meeting.

Formal votes are reported as:

In-favor / Opposed / Abstaining / Not-voting / Total-eligible

Secretary's note: "Meyer" refers to Randy Meyer (the singular) from Convex, and "Meyers" refers to Randy Meyers (the plural) from DEC in these minutes.

1. Opening Activities

1.1 Opening comments

Plauger convened the co-located WG14/X3J11 meeting at 10 AM, EST, 5 December 1994, and immediately turned the meeting over to Jaeschke, chair of X3J11.

Jaeschke stated that the main goals of the meeting were to complete the work on Defect Reports from the Tokyo meeting, and to continue work on revision

of the standard, including finalizing work on the numerical C extensions Technical Report.

1.2 Introduction of Participants/Roll Call

Attendees introduced themselves and indicated roles as focal points for NCEG subgroups or other work items. A copy of the attendance sheet of X3J11 members is attached to these minutes. Martin (BSI) and Plauger comprised the ISO membership in attendance at the meeting.

1.3 Selection of Meeting Chair

A Jaeschke was selected to chair the meeting.

1.4 Procedures for this Meeting

Plum clarified voting membership. Tydeman and Guilmette were identified as voting members as of this meeting.

Meyer (Convex) was the host, and described arrangements for duplicating and file printing. Information on local eateries was also provided.

After the attendance list was circulated, Plum announced that there were 20 eligible voting members of X3J11, of whom 18 were attending. Two countries were attending for WG14, the US and the UK.

1.5 Approval of Previous Minutes [N352/94-037, N375/94-060]

A The minutes from the San Jose and Tokyo meetings were accepted as distributed, after numerous typographical errors were reported in both.

1.6 Review of Action Items and Resolutions

[From the San Jose meeting, N352/94-037]

MacDonald will post an announcement to the NCEG reflector that it will be discontinued and future postings should be sent to the J11 reflector since the two committees have now merged. Done.

Farance will review API - C Binding for File Transfer and respond on our behalf. Done.

Jaeschke will take on task of being our liaison to X3T2 [a.k.a. WG11]. Done.

Farance will circulate his response to CBEMA RFI's on electronic documents to J11. Done, N371/94-057.

Keaton will split the compound literals and designated initializers proposals into two proposals. Done, N356/94-041 and N357/94-042.

MacDonald, Mooney, and Gwyn will review the separated compound literals and designated initializers proposals. Done.

MacDonald, Tydeman, Plum, and Keaton will serve as review committee for the [complex with imaginary type] proposal. Done.

Meyer, Tydeman, Kwan, and Keaton will review the edited [FPCE] document. Done.

Zeeb, Plum, Gwyn, Keaton, Thomas, and Walls will review Jaeschke's revision of this draft [C Standard Revision] charter. Done.

*** Benito will post the effective date of the DAM to J11 email reflector when known. Still pending.

[From the Tokyo meeting, N375/94-060]

Secretary's note: the action items reported in the text of the minutes were reviewed instead of the ones summarized under "Review of Actions Items" in the minutes, according to my notes, although I have a few notes marked for just the 'review' action items as well. There are contradictions in some of these as indicated. Also, in the following items, "Jones" refers to Derek Jones (a.k.a. "Jones the less than or equal to") and not Larry Jones (a.k.a. "Jones the greater").

Plauger will prepare liaison report [on C++ draft for CD registration for pre-Dallas meeting]. Still pending.

Plum will let people know how to send their comments on C++. Still pending. Secretary's note: 'review' action item stating Plum will post comment template and receive all C++ comments is marked as 'done' in my notes, so I must have heard one of them incorrectly???

Jaeschke will draft revision proposal and guidelines document. Done.

Jones will provide wording for DR#109. Done at meeting.

Jones will provide wording on "heap" [for DR#138 (or DR#68?)]. Done at meeting.

Farance will locate all reference to "lifetime" [DR#138 (or DR#168?)] Done at meeting.

Jones will send [coversheet] template to Jaeschke by 1994-08-05 [or 1994-08-08?]. Done.

Jaeschke will produce standard cover letter (guidelines for submission). Done.

Jaeschke will ask Ed Keiser about hosting 1996-06 meeting. Done.

Plum will send E-mail to the reflector on new mailing dates. Done?

Farance will provide schedule on combined document. Done.

Farance will investigate straw poll (ballot) over E-mail reflector prior to the 1994-12 meeting. Done.

Secretary's note: the following 'review' action items were not covered in the text of the minutes, and I have no notes on their resolution, but I assume they were completed as reported in N377/94-062.

Plum will produce revised wording for DR#69.

Plauger will draft wording for DR#88.

Plum will provide wording for DR#140.

1.7 Approval of Agenda [WG14/N378]

New items needing agenda time were identified, and documents relating to each item were also identified. There was discussion for how to distribute time to items so that required items would all be addressed at this meeting.

Jaeschke stated that a separate US meeting was needed to select a project editor (5.2).

Martin suggested adding an agenda item to discuss future logistics for defect reports (handled under 6.1).

Jaeschke noted that we should determine the X3J11 position on numerical issues so that participants know about the future. Would like to wrap up any changes to the Technical Report by the end of the meeting.

Plum identified a proposal from Bruck to ban implicit int to be addressed as a Monday paper (5.3).

Evening meetings were announced to cover a review of the new SGML version of the base document (Farance), to review and answer any questions on details of the DPCE proposal (Stanberry and Keaton), and to discuss the extended integer proposals (Kwan and Farance).

1.8 Distribution of New Documents

New documents were assigned WG14/X3J11 numbers, and will appear in the next X3J11 mailing. Available documents were distributed.

N388/94-073 "UK Defect Reports #2-25," Martin.
N389/94-074 "Comments on LIA," Schaffert.
N390/94-075 "Extending character constants for named characters," Farance, distributed.
N391/94-076 "Arrays as first class objects," Farance, distributed.
N392/94-077 "Extending VLA's to include variable rank arrays," Farance, distributed.
N392/94-078 "Adding support for distributed objects," Farance, distributed.
N394/94-079 "Project Proposal–Revision of C Language Standard ANSI/ISO 9899-1990," Jaeschke, distributed.
N395/94-080 "Data Parallel C Extensions, Version 1.6," Stanberry.
N396/94-081 "Minutes of the Plano WG14/X3J11 Meeting," Stanberry.
N397/94-082 "Complex C Extensions," Thomas and Tydeman.

Note: NCEG documents are available through anonymous ftp to Keaton's ftp site. They are in directory DMK/nceg at

```
ftp.dmk.com
```

1.9 Information on Next Meeting

Plauger confirmed that the next meeting will be June 12 to 16 in Copenhagen, Denmark.

*** Jaeschke will contact Keld Simonsen about getting information on the June meeting into the post-Plano mailing.

Gwyn asked confirmation that meetings will be scheduled such that ANSI members who cannot attend non-US meetings can maintain voting rights. Yes, 2/3 of the meetings will be scheduled in North America.

Plauger suggested that those attending the Copenhagen meeting consider a downtown hotel since the meeting site is isolated from those areas that are most likely to be of interest for nonmeeting times.

2. Liaison reports

2.1 X3J11

Jaeschke reported there had been no meetings but there was some administrivia to report. CBEMA is changing its name to ITI (Information Technology Industries Council) as of December 1st. (Note that the acronym is ITI, not ITIC!) They have also announced a new policy of assessing a 1% late fee per month for late payment of fees for 1995. ITI also has new email addresses, which were sent to the J11 reflector:

```
<name>@itic.nw.dc.us
```

2.2 X3J16/WG21

Plum gave an update on the progress of the C++ committee. They are still on schedule. The March meeting in Boston will vote on the CD ballot which will introduce the public review period. They are using a 60 day review as part of a very tight schedule. They have no new liaison issues with us.

Plauger stated that we do have a liaison issue of commenting on their draft. We should go on record for noting incompatibilities with C, and should comment before the CD registration ballot ends in January. He suggested we should form a small committee here to review a draft response to SC22.

Guilmette asked what had become of Jaeschke's review: is it being edited into the working paper? Plum believes that it is, but doesn't know the status of those edits.

Farance stated that it is hard to review such a huge document that keeps changing so frequently. It would also be desirable to have an electronic version so that searches were possible. Other problems noted with trying to review the draft are getting access to compilers that implement new concepts to evaluate them and see how they interact with C. In summary, he's reluctant to devote resources to do this review.

Plauger suggested that, given the changing state of the document, we must chose an appropriate level at which to comment. It is imperative that we comment, even if the draft has vacuous semantics; indeed, that's a legitimate concern where the semantics may impinge on C.

Farance agreed, and recalled that Plum had suggested such a high-level review at the Tokyo meeting.

Plum noted two categories of change that have been occurring in the C++ committee. First, clarifications that affect global changes to the entire document (modeling changes). Second, the addition of new features or extensions. The Golden Rule of Liaison: we should give the kind of cooperative commentary that we would like to receive. Be clear where we would like to give or be given independence of development, and where we would go to the wall to argue for/against features that affect compatibility. And in the middle ground, he recommends that we handle issues only "informatively."

Farance suggested that even informative comments should be made early.

Gwyn stated that we should concentrate on what are philosophical issues: environmental issues that would cause C/C++ to require separate compilers. The PC market is moving toward a single C/C++ compiler packaging so identify any other issues (namespace, linkage) that would cause vendors to drop one or the other language.

Guilmette said that how conformance is specified is a major "go to the wall" issue with him. He would like to see C++ use the same approach as C.

*** Plauger will assemble list of issues with the WG21/X3J16 CD registration working paper and submit this to SC22 as our response.

Submit comments to Plauger by New Year. We should give general concerns now so that we can be more specific about our concerns in the future.

Guilmette believes we should qualify even our early comments to indicate which issues we feel strongly about.

*** Farance, Martin, Benito, Guilmette, and Plum will serve as review committee for our response to WG21.

Jaeschke suggested that our response be given to Keaton to be made available through his ftp site.

2.3 WG15 (POSIX)

Farance reported that he had reviewed POSIX.4 Real Time Extensions Document and sent comments to them and to the reflector.

Martin asked if there were some issues with locales–e.g., locale registry?

Plauger said that POSIX was extending C's support for international locales. WG20 is seeking approval to include such extensions but still doesn't have a standard out.

2.4 WG20 (Internationalization)

Jaeschke asked if there was any impact on our revision.

Plauger reported that they are working on a "universal sort string" which differs from `strxfrm` model. We need to keep watching this. Also, they are working on their charter which is difficult because of the task they have. We need to keep watch on this too.

2.5 Other Liaison Activities

LIA

Plauger reported that he had just received WG11's response to comments (including Tydeman's) in the US. These will be circulated in next mailing.

Plenary

The Plenary session of SC22 was held in September at The Hague.

Plum said, "The reason we go to these plenary meetings is to keep bad stuff from happening." Plum and Plauger attended, and bad stuff didn't happen! They promised to develop a timetable and nominate a project editor for the C revision at this meeting. They complained to SC22 that they still didn't have any information on whether Amendment 1, TC1 and RR1 have been approved. Henceforth, the convenors are supposed to be notified of the status...but still haven't been.

Martin then announced that TC1 has been published, and he had a copy to prove it!

Plauger noted that he had received a request for some edits to Amendment 1.

IRDS–Information Resource Dictionary System (SC21/WG3)

Plauger received notification that this committee is drafting a C binding for which they suggest C be extended by adding a Boolean type and adding a standard NULL constant (i.e., they failed to notice that it already exists). They have requested that we provide comments and guidance to them. The notification said that a copy of their draft was included, but it was not; we need to get a copy from them. Plauger will respond on our behalf.

*** Martin will investigate whether Derek Jones has already commented on the IRDS C binding.

Gwyn suggested that their binding specify a header which defines the needed Boolean type.

Plauger agreed they can get what they need with a header file without waiting. We should volunteer to review (Derek Jones has already done so, apparently). And we should determine what we want to do for the future.

Farance stated that we should submit any formal response as a circulated document to cite for any future response.

*** Plauger will respond to IRDS after consulting with Derek Jones.

IEEE

Farance reviewed IEEE 1238.1. Problems were noted in interrupt servicing (see N387/94-072). Farance also voiced concern with ISO 10646 extended character recognition which is to provide C bindings of their characters.

Plauger shared amusing, semi-serious proposal raised at the Plenary about requiring all languages to produce a C binding, recognizing C as the universal assembly language.

MacDonald reported that X3J3 has created a new subcommittee to investigate interfacing with C and C++, as well as other languages.

3. Status of Amendment 1

To the best of anyone's knowledge (i.e., according to rumor and faith), this has been approved.

4. Documentation Project

Farance reported on the status of the X3 ad hoc group on mechanization, which he had taken an action item to investigate and participate in. They are following our suggestions pretty much–e.g., looking at postscript, Adobe Acrobat, searchable text form. Note: there are four forms to consider for each document: text, display, structural, and conceptual.

Farance then described N385/94-070, which is an SGML version of the Standard plus TC1. He proposes we use SGML for the structural form of the next revision; he has no recommendation yet on display form. SGML can be exported to many popular word processors and converter tools. Farance has created some converters. HTML is a superset of SGML so could be derived from SGML–i.e., HTML is a specific SGML format with extensions for links.

Plauger mentioned that we should ignore Ventura Publishing due to problems with newer version of this tool, and other tools now available.

Farance proposes putting SGML validator on Keaton's ftp site for use by editors/contributors to the document.

Plum noted that everyone has to use the same DTD (Document Type Definition), so this needs to be on the ftp site also.

Guilmette asked for further clarification of HTML vs. SGML. Are tools to/from TROFF available? Yes, Farance will make these available. What are the arguments against using SGML? Disadvantage is the absence of tools to do conversions.

Gwyn asked what does ANSI want? SGML.

Plauger stated that we should aim to have HTML document by the time of CD registration. Farance indicated that the work involved plus investigating the legal issues for this would take about a year.

Guilmette wanted to know if there was a stable standard for SGML. Yes, X3J6 (and ISO committee?) has been working on this for over 6 years.

Gwyn asked if it was worth the investment if converters aren't available on WWW to read the document.

Plum and Plauger noted that J16 has some experience with making HTML document available, and that restricting access to committee members is possible.

Martin believes ISO is using HTML.

SV In favor of keeping the Standard document in SGML?
21 Yes. 0 No.

Gwyn expressed concern about whether the MM and SGML versions are the same. Farance will come up with mechanical way to verify translation.

Jaeschke asked when will the SGML document be ready to formally adopt as the base document. Farance indicated it would be done 28 April 1995, so in time for the Copenhagen meeting.

5. Revision of ISO/IEC 9899:1990

5.1 Review of Charter for Revision [N374/94-059]

Jaeschke opened the discussion by noting that there were the 6 original principles and 5 new ones.

Gwyn asked if we were committed to publishing a Rationale. There was considerable sentiment to do this.

Ream asked if the committee really endorses changes that would require all implementations to change? In the ensuing discussion, it was noted that there were several options–e.g., we could reaffirm the Standard as is, or just fix "fuzzy" things. Ideally, no existing C code would break, but if it does, it must be a noisy, not a quiet change.

Gwyn stated that these principles give us a justification for revoking proposals. They are limiting the scope of extensions, but not precluding them.

Guilmette added that some persons have already misconstrued principle 10 (Minimize incompatibilities with C++) to imply that C will become C++.

Gwyn and Plauger argued that there are tradeoffs between these principles, that complying with all of them may not be possible. Rationale is needed, however, whenever we deviate from one principle. It was suggested that wording to reflect this be added as a fifth "observation" under section 4.

Guilmette asked if "existing practice" must be in C. No, could use prior art from other languages, including C++! Suggested that wording clarify this by adding that "Existing practice is not limited to C compilers." Note: can't limit to just C prior art since any extension is non-conforming and therefore not C!

SV In favor of not limiting prior art to just C implementations.
20 Yes. 1 No.

MacDonald suggested that point 3 under section 4 be reworded to be stated in a positive way.

With respect to section 5, Gwyn noted that Amendment 1 and TC1 are already part of the standard. Also, would prefer that we strike references to particular languages/groups.

On section 6, Jaeschke noted that a submission must be sponsored; he will assume the role of filtering submissions from J11 before sending them to Plauger.

On section 8, Plauger will try to get more precise schedule after reviewing Sam Harbison's detailed schedule for WG21. He will take this schedule to the Plenary in September.

Mooney asked for clarification of the procedures and meaning for each milestone.

- For CD registration, must have all major parts in place and the document must be in ISO format. There is typically a 2 month ballot.

- For CD ballot, must consider document done, submitting to SC for ballot. If there are No votes, must be with objections. If the objections are met, the No voters are obliged to change to Yes. There is a 2 to 4 month ballot.

- DIS ballot is for final polishing–answering the No votes.

Tydeman asked when does public review happen? Plum responded that the last two ballots have ANSI public comment periods; however, for the second of these, no comment is really expected.

Plauger noted that we can use shortened, accelerated review periods, although this makes for very tight schedule. Several persons expressed mixed feelings about this since the public comments really improved the quality of the original Standard.

Plauger added that the Internet is better now, and we can use various channels to get early distribution of the document, close to CD registration. Farance said this raises the issue of making the document available electronically. Plum stated that we should be sympathetic with X3's service considering their budget. Plauger agreed there is a delicate balance but we have to err in favor of more widespread dissemination of the document.

Getting back to the schedule in section 8: it is admittedly very ambitious. Gwyn noted that it limits the amount of technical work that can be done;

December 1995 is the last time to bring new proposals. Plauger agreed–we need to get the shape down by the end of the Copenhagen meeting. Plum spoke in favor of delaying exact timelines until we see how well we define this shape over the next two meetings. Plauger affirmed that we want to delay publishing a schedule to the Plenary until September. Everyone agreed that the goal of this schedule, however, is to limit changes to the Standard.

*** Plauger will come to Copenhagen with more precise schedule document for the C revision.

*** Jaeschke will revise the C revision charter document as discussed.

5.2 Selection of Editors

Plauger stated that there are only two official ISO positions: convenor and project editor. If no editor is appointed, it is up to the convenor to do the work. ANSI proposes, ISO disposes on the issue of who should be project editor.

Jaeschke stated that only one candidate has expressed interest so we should vote to confirm him.

MSP Move we nominate Farance as project editor. (Meyers, Mooney) 17/0/0/3/20

Gwyn asked for clarification on what constitutes a necessary vote. Plum stated that a 2/3 of the membership was required.

*** Jaeschke will do the necessary work to process the appointment of Farance as project editor.

5.2.1 Submission Guidelines [N380/94-065]

Jaeschke stated that the guidelines will impose organization on the proposals submitted (and therefore reduce submissions!) by requiring the cover letter be filled out.

Farance suggested that a revision level be added. Jaeschke noted that there should be a new document number for a revised proposal with a reference to history document numbers. He will add a line near the top for revision history.

Farance also suggested that each proposal should contain a development plan–e.g., where a proposal is sounding out interest, but not detailed. There was no clear sentiment for this as it was seen as an unnecessary extra rule.

Mooney suggested that the E-mail address should be an Internet address. Jaeschke will amend to include a template.

Gwyn suggested that we add an "Other" category for Area of Standard Affected–e.g., document format, which affects the whole document.

*** Jaeschke will revise the submissions guidelines as discussed.

5.3 Technical Proposals

Jaeschke opened the discussions with suggestion that we determine by the end of the week which of the NCEG proposals we intend to include in the revision process.

C++ features

Walls stated that we need to decide what features of C++ to include in the revision. We need to consider Bob Jervis' proposal, N298, to add classes to C. Plauger cautioned that we need implementations to evaluate to avoid "Gedanken experiments masquerading as international standards."

Jaeschke suggested the question is whether we need OOP support in C as Jervis asks. Plauger asserted that it is not our intention to provide a complete OOP environment; we should put an appropriate disclaimer to limit what we are promising, but the Jervis proposal is a good first cut.

Ream reported that he has been simulating the Jervis proposal, less virtual functions, in his implementation–i.e., trying to represent classes in C. He noted that class-oriented is different paradigm from OOP.

Plum proposed that we should limit inheritance to common prefix, and allow a pointer to a derived class to get assigned a pointer to a base class.

MacDonald asked what problem in C do classes solve? Plauger suggested that it solves limitations in C such as its flat external name space; there are a half dozen conceptually and performance cheap methods that have been proven in C++ that would be useful; but we should draw a line where complexity starts to be expensive (e.g., name mangling, namespaces). MacDonald asked if we can solve these problems with structures rather than classes.

Kwan expressed concern about knowing what to cut out of the Jervis proposal.

Ream thinks the major strength of the Jervis proposal is that it focuses on class design.

Meyers stated that it is possible to stop eating after one peanut: we can adopt some features of C++ without others. For example, single inheritance was successful in Smalltalk. He also argued that virtual functions are not expensive to implement and are a real win for programmers.

Farance hopes that we can use the experience of C++ to add better understood and engineered features so that C remains a well-defined subset of C++. He doesn't think big vs. little features is necessarily a good measure of what we should add, but we should add features to gain experience and let things evolve.

Gwyn believes consensus is on drawing the line pretty much where the Jervis proposal does, modulo some tweaks to satisfy other constraints.

Plum reminded us of current C principles: overt semantics, no runtime baggage, acknowledged building block or "universal foundation" for other languages. We should maintain these virtues. C++ features that violate them include name mangling and overloading.

Ream agreed and added that the features we adopt should be as ugly as necessary...but no uglier!

Meyers suggested that some form of name mangling may be necessary, for example, to support member functions. Some featured depend on others.

*** Meyers will produce a paper describing dependencies of features of C++.

Meyers stated that he would anticipate our being questioned on why we chose single inheritance—e.g., doing a subset language later. His answer is that this can serve as basis for other things.

Farance asked how run-time costs of C++ differ from ordinary abstraction techniques. He suggested that the abstraction tools are not being used correctly if no savings is realized.

Gwyn believes programmer discipline can be substituted for class-oriented semantics, so perhaps we could just make a few extensions for structures.

Thomas said we should be able to get the most out of the C++ experience and not make the same mistakes in adding what we want.

Plauger noted that the costs of some features (such as virtual functions) are incurred only when they are used, but the costs of other features (such as multiple inheritance) are incurred everywhere. The cost for getting the initialization of objects right needs to be made clear to the users, too. With respect to evaluating features using a dependency tree, he noted that this still leaves open the possibility of buying into a subset of the features.

Ream noted there are intellectual costs of adding new concepts that should also be considered.

Meyers stated that building all programs from Turing machines is possible, but not best idea. There is a fine line between when to build out of simpler

concepts vs. adding new concepts. However, simulating features leads to ugliness and interferes with optimization.

MacDonald reminded us that we need to weigh any proposals against the C9X charter item 2.6.

Gwyn said some features are just for convenience but we should determine what concepts we really want to support.

Ream made an appeal that we eliminate surprises.

Crigger said we should keep in mind side effects of each feature.

There was balloting on several issues to determine how to direct Jervis to proceed with his proposal.

SV In favor of using the Jervis proposal on classes in C as the starting point for including some C++ features in C for the revision.
10 Yes. 8 No. 5 Don't know/care.

We repeated the same straw vote, but at Walls request, asked for voting members only. The results were:
6 Yes. 7 No. 4 Don't know/care.

MacDonald qualified his no vote with "Can we do it with structures?"
Guilmette qualified his no vote with "Can we have separate proposals for each feature?"

Plum asked if we should vote on who believes we should add some OO features. We should state the question in such a way as to make it clear whether or not we want C to evolve to support OOP. Meyers suggested that we leave it open for others to categorize whether C9X supports OOP. Guilmette added that we can't vote on this since we don't know what OOP means.

Gwyn suggested that we rephrase the vote in terms of explicit features in the Jervis proposal. Meyers identified the following four features:

(1) access control on members
(2) inheritance
(3) member functions
(4) virtual member functions

SV Should one or more of these features be included in revised C?
13 Yes. 0 No. 6 Don't know/care.

Walls asked if there was strong opposition to any feature. Guilmette said he only wants (2). Meyers argued that all of these have low implementation cost and are useful; he believes (1), (3), and (4) are needed.

Plauger said, "Fifty years ago, there were no stored program computers, nuclear weapons, or salad shooters. We have to decide which kind of things these are." Farance asked, "What's a salad shooter?" Plauger replied, "I rest my case."

Plum believes it is too early in the process to see implications of these features with other features that we may want to add.

SV Are you strongly opposed to:
(1) 1 Yes.
(2) 0 Yes.
(3) 1 Yes.
(4) 4 Yes.

Implicit Int

Plum asked if anyone wanted to direct him to tell C++ not to ban implicit int. He hoped not. He reminded us that the vote for the first Standard was very close on this issue.

Jaeschke clarified that banning implicit int means getting rid of the rule in declarators that says if there is no type specifier, int type is assumed.

Meyers thinks this will have only a minor impact to implementors. Gwyn stated that theoretically it matters, depending on context of user/customer base. MacDonald has sympathy for banning implicit int, but believes that practically it would cause lots of problems.

SV In favor of deprecating the rule that no type specifier is equivalent to int type in the next revision.
20 Yes. 0 No. 1 Don't know/care.

******* Gwyn will draft a proposal for deprecating implicit int for the next revision.

// comments

MacDonald said he'd been assuming this was universally favored for including in the next revision.

Gwyn said that if C++ has precise semantics for what happens to characters after // we should use their specification without deviation. Ream agreed we should make this clear as there are variations in C++ implementations.

Meyers added that it is a quiet change so we need to include some rationale.

Jaeschke asked if it would require change to phases of translation. Plum indicated that C++ decided not to make any changes, so if you use a macro with // as the last characters before a continuation \ you will be very

unhappy. Meyers asked if C++ would agree to allowing \ to terminate a comment. Plum thought this would be unwelcome at this point in their standardization process.

Gwyn wondered if the actual impact could be more than we imagine. Plum said we can always reconsider if new information is discovered.

SV In favor of supporting // comments in the next revision.
Lots Yes. 0 No. 2 Don't know/care.

******* Gwyn will draft proposal for adding // comments to the next revision.

Mooney and Jones both expressed opposition to any changes in the phases of translation. This seemed to be the consensus of the committee as a whole.

Signed integer arithmetic

MacDonald wants to fix signed integer arithmetic so that the modulus operator works right; make it work the same as in FORTRAN.

Gwyn said we want a mathematically uniform step function; truncating toward 0 doesn't do this as it produces a leap around the origin.

MacDonald said we only need to fix division with negative integers which is where we should remove the implementation-defined behavior currently allowed. The rule for modulus will then just fall out.

Gwyn said if we start opening these issues, we need to revisit other areas as well and fix other implementation-defined gray areas. Guilmette agreed that we should revisit these areas and remove past compromises for strange architectures. Gwyn asked if all architectures agree now, and will they forever? And do languages agree?

Meyers asked for clarification that this was just standardization of integer division, but not necessarily of the remainder operator %. This raised the question of whether % would become a modulus operator. Plauger argued that it should remain a remainder function, and should still be defined in terms of division. Meyers was then generally supportive, but wanted time to reflect on this. Gwyn added that one objection is that this precludes someone from making % a modulus operator.

SV In favor of / being well-defined for signed integers in the next revision.
13 Yes. 1 No. 9 Don't know/care.

******* MacDonald will draft proposal to make integer division well-defined for the next revision.

#line directives in macro invocations

MacDonald presented David Gay's request, which had been circulated on the WG14/J11 email reflector. Essentially, he has asked that either we allow `#line` directives to occur in macro invocations or that we extend the logical line length to accommodate very long lines generated by macro invocations.

Plauger reminded us that historically we didn't want to allow preprocessing directives in macro invocations because "therein lies madness." However, some preprocessing directives are sufficiently benign that they could have been allowed. One last point is that the `#line` directive was intentionally a weak directive to aid in debugging support. Walls said we should clarify, however, that this is a compiler issue and not a debugging issue.

Plum would like to move away from describing source text position in terms of `#line` and `__LINE__` to allow more flexibility for future descriptions. Gwyn agreed; we shouldn't add complexity to antiquated descriptions.

So MacDonald asked if the consensus was in favor of extending the logical line length. Jones asked if that is really what Gay wants. Yes. Gwyn was concerned that this wouldn't work since if there were errors in the macro definition, could still get wrong line numbers.

Plauger said the alternative was to partition all preprocessing directives into those which could occur in a macro invocations and those that could not.

Plum said that C++ has proposed 64K as the expansion buffer limit, so we should consider using that if we change the logical line limit. Benito argued that the C++ limits are huge–much too large for suitable minimum limits. Plauger said we should balance compatibility with reasonable limits for C constraints which are different from C++ requirements.

Ream asked for clarification on whether more than one `#line` directive could occur in a logical line, for example with spliced lines. This can't happen since the preprocessor is a line-oriented mini-language; each directive lasts to the end of a logical line.

There was then considerable discussion on what is a reasonable limit. Historically, limits were chosen to fit in a 64K address space. Plum suggested we should consider 64K systems obsolete now, and we should identify one or more architectures for minimum system requirements to determine limits. MacDonald suggested 32K as a trial balloon. Plauger suggested we poll the committee on whether a half-megabyte machine is considered minimal. Guilmette said we should ask about implementations versus architectures. Crigger added that since we allow 8 levels of nesting through `#include` file, we may have to consider 8 times the limit we choose, depending on the buffering methods used. Plum said we should phrase the limit as total line buffering requirement.

SV In favor of identifying which preprocessing directives are recognizable in macro invocations.
3 Yes. 9 No. 10 Don't know/care.

SV In favor of increasing the minimal translation limits from a 64K conceptual machine to a 500K conceptual machine, and determining new limits for all translation limits.
15 Yes. 2 No. 4 Don't know/care.

*** Benito will investigate new minimum translation limits for a 500K conceptual machine.

*** Jaeschke will communicate our decisions regarding translation limits and `#line` directives to David Gay.

Designated Initializers [N356/94-041]

Jaeschke gave a brief history of this proposal: it originated with Ken Thompson, and was originally proposed for inclusion in the NCEG TR by David Prosser. Keaton is now the focal point for this proposal.

MacDonald spoke in favor of this extension, saying it satisfies criticism from the FORTRAN community, although they still would like to see a repetition count included. Plauger said this is an elegant proposal, and would just want to be sure that a repetition count proposal would not collide with the syntax of this one.

Plum discussed possible C++ liaison issues. J16 POD's are C objects that can be walked byte by byte (e.g., `memcpy`'d), but this is not true for C++ objects (e.g., virtual base classes). As long as the designated initializer proposal applies only to POD's, no anticipated liaison problems. But we need to be careful if we expand C objects during the revision such that we introduce non-POD's. As an aside, he noted that the Jervis proposal adds virtual functions but not virtual base classes.

Gwyn said the fact that the source of the proposal is AT&T, conceptual minimalists, speaks well for the proposal: it is not a gratuitous extension.

Plum would like to see direct personal liaison with Ken Thompson; doesn't want to see it repudiated by him without our knowledge.

MacDonald recalled that the repetition count proposal had been waiting on array syntax issues. Maybe now that DPCE is completed, it could proceed. Gwyn would like to see a precise proposal before deciding to incorporate.

SV In favor of incorporating the designated initializers proposal into the next revision.
11 Yes. 0 No. 1 Don't know/care.

*** Keaton will draft proposal for adding designated initializers to the next revision.

SV In favor of seeing a repetition count proposal extension for designated initializers.
18 Yes. 0 No. 5 Don't know/care.

*** Keaton will draft proposal for adding a repetition count to the designated initializers proposal.

*** Keaton will liaison with Ken Thompson regarding modifications to the designated initializers proposal.

Compound Literals [N357/94-042]

Keaton reviewed the editorial changes made since the last version which had been approved at the San Jose meeting.

Keaton argued that the reason for including this proposal in the next revision is that it gives the convenience of constructors without the associated overhead. It can be used for creating expression temporaries, and this feature is also in Thompson's implementation.

Plum expressed concern that the interaction with any struct/class extensions adopted would have to be worked out in detail before we adopt this–e.g., how would it work with respect to private members. Plauger said the onus would be on Jervis to work out those details, but he agrees that we should not adopt this until those details are provided.

There was further discussion on how to add interdependent proposals. We need to determine a process for handling this. We can't postpone decisions on proposals that could interact with others, burden should be on others if subsequent proposals. But we need a conceptual, not a temporal ordering.

Keaton suggested that we have to consider the proposal both independently as well as in concert with the Jervis class proposal. Gwyn said that the compound literal proposal has more history with the committee so should have somewhat more weight, but still have to consider all the proposals together as much as possible. Meyers stated that if there is a tradeoff between a strong proposal for classes and a weak proposal for compound literals, he believes classes are more fundamentally useful.

Plauger lamented this is another case of Occam's Razor: of course it would be simpler if there were only one method chosen. But there are already two ways to do initialization in C++, constructors or initializers. It would be worse, however, to add compound literals now and then have to battle to take them out later.

Plum returned to the issue of how private class members would get initialized, which is unspecified in the Jervis proposal. Could be specified as a compiler-generated constructor which provides default initializations. Also, compound literals interact with array syntax–i.e., arrays can be constructed in an expression. If we adopted a new, clean array syntax, we would have multiple abstractions.

Farance suggested that we need to consider that there is no one programming paradigm as we may have thought 15 years ago. We now recognize that there are really multiple paradigms. Gwyn was sympathetic to this view, but added that we have to balance with constraint in our charter that there be just one way to do an operation.

Plauger suggested that making incremental improvements to something that works is the best approach, but committees have to use others as well. The odds are good that the compound literals extension will be reconcilable with classes so we should proceed with the understanding that we must be committed to reviewing this again during the revision process.

Gwyn agreed–this is already in the TR, so we should go ahead and vote.

SV In favor of pursuing the compound literal approach in the next revision. 14 Yes. 0 No. 7 Don't know/care.

*** Keaton will draft proposal for adding compound literals to the next revision.

Jaeschke suggested it would be worthwhile adding some rationale to reflect this discussion to that proposal.

Thomas asked for clarification that we have just decided to pursue this approach, but haven't adopted the compound literal proposal as yet. Yes.

Floating Point C Extensions [N364/94-049]

Thomas gave a brief overview of the proposal and its goals, which are to make floating-point programs more predictable. What he desired to get as feedback from the committee is what and how to propose in parts from this proposal for C9X. The proposal contains two levels of specification: one for basic C implementations and a second for IEEE implementations. The goals of FPCE were also twofold: to provide predictability of programs without loss of efficiency and to provide support for IEEE. For predictability, if you know the hardware and the inputs to an operation, should be able to predict the result, within parameters allowed for flexibility in implementations. For IEEE, this is a real and pervasive hardware standard, but without software standard, can't serve user community.

Thomas asked what role should the IEEE specification have in a C revision. Should it be part of the Standard, an Appendix, or remain as a TR? Tydeman prefers that it be physically part of the Standard, even if in a nonnormative

appendix. MacDonald pointed out that there is hardware that supports IEEE representation, but doesn't really support IEEE arithmetic. Gwyn added that the IEEE part of FPCE contains too specific architectural requirements for all vendors. Tydeman indicated that we may also have a requirement to provide an appendix for LIA. Meyers is unsure of the user demand for full IEEE conformance–e.g., the DEC Alpha machine can run in mode which does not support full IEEE arithmetic, and that is overwhelmingly the preferred mode. Thomas said that the specification was useful for various levels of IEEE conformance. MacDonald said we should look at the FPCE proposal separate from the IEEE specification; he would not like to put the burden of making the IEEE part normative on implementors.

Thomas then asked what sort of granularity of proposal would be most useful, given that predictability is the major goal and not all features can achieve this.

Gwyn said we need the details of the overloaded/intrinsic function semantics since this requires general linguistic support. Thomas noted that you only get these if you include `<fp.h>`, and that the semantics are within the range of acceptable results for `<math.h>` functions. If a general overloading mechanism was available, however, would use that instead of the intrinsic functions. It is also negotiable to back off of this approach and revert to using suffixes to distinguish functions.

Meyers reminded us that the goal of NCEG extensions was to allow and promote experimentation. However, the FPCE proposal is too big to swallow all at once, and some of it is designed for a "boutique" audience. It should be broken up to consider what has already been proved good.

Keaton added that this would prevent throwing out everything just because some features were not palatable to everyone.

Plum returned us to the subject of LIA–we have to take this seriously. Plauger reported that he had promised at the Plenary that we would address it but it is not necessary to do anything for it with the revision. MacDonald asked what are the implications of LIA? Plauger replied that we need a separate report or appendix showing how to comply with LIA, providing a specific mapping.

Meanwhile, the discussion continued on the needs of the "boutique" audience that uses IEEE. Plauger stated that his experience confirmed Meyers'–even the boutique audience doesn't usually want full IEEE. Gwyn said even a boutique audience needs a standard. Stanberry agreed and asked what are the available mechanisms for getting a standard for a specialized audience such as IEEE or DPCE so that both can write portable programs. Martin indicated that ISO publishes "guideline standards" which don't have the full force of a standard: "shall" -> "should."

Farance spoke in favor of separating features so that we could pick and choose among them.

Plum suggested that we assume that the FPCE proposal will be a nonnormative annex and then decide what parts to make normative so we can make progress. Meyers agreed with this approach. Gwyn added that more persons are served by having all the details in the Standard.

Thomas developed a preliminary spreadsheet of features X goals, which he presented. Each feature was characterized by which of the following goals it achieved (could be more than one):

- Major goals
  - Predictability with efficiency
  - Predictability with efficiency for IEEE
- Lesser goals
  - IEEE binding/support
  - Reasonable code/data portability (e.g., so IEEE could do something reasonable)
  - Other value–e.g., really convenient
  - Separable (i.e., could be entirely separate proposal)
  - Separable to IEEE specification
  - Value not clear (yet)

Thomas will use this information to try to chunk pieces of FPCE into proposals for the revision.

Gwyn asked if evaluation modes can be set for translation time. No.

Guilmette asked why use overloaded functions. MacDonald said this avoided name explosion and provides generic capability users expect. Thomas pointed out that the built in operators are already overloaded. Gwyn added that it is easier to change types in a program; don't have to change function names. Guilmette thought this was a good argument for a general overloading feature. Gwyn noted that FPCE proposes overloading only for a fixed set of know functions. Keaton said the overloading was important for "widest-need" functionality: if the functions were not generic, could risk undesirable narrowing of arguments. Plum said if we are aiming this for the numeric market and think overloading is worth doing, then we shouldn't do it half way; alternatively, if we don't want to add overloading, C programs could invoke C++ functions which can be overloaded to provide "intermediate overloading." Stanberry added that if the overloading functionality is useful to language specification, should allow users access to the same functionality. Gwyn was agreeable provided we find a better way to specify overloading than the way C++ does it.

Meyers suggested that those things done with pragmas in the TR would be best done by keywords or macros for the revision.

Thomas asked for feedback on what approach to take while separating FPCE into reasonable slices. Jaeschke suggested that he should try to leave the proposal as intact as possible for ease of logistics; use straw votes on partitions as guidance for future submissions without trying to determine every possible slice now. Also should describe new proposals as slices of the existing FPCE proposal.

SV In favor of forwarding the FPCE document for consideration for the revision, with the understanding that Thomas will prepare proposals for reasonable slices.
19 Yes. 0 No. 3 Don't know/care.

Meyers said we need to get direction on interacting proposals early, such as overloading. Guilmette wondered how many more sausages were in the bag...

There was more discussion on how to handle the IEEE specification. If put in an appendix, would still give high status to IEEE. Thomas will separate IEEE issues into features (kept with other FPCE specifications, "if IEEE, behaves...") and binding (separated to an appendix).

Restricted Pointers [N334/94-019]

MacDonald briefly reviewed the proposal: basically, restricted pointers work "like an array"–i.e., accessing a unique object, which allows optimization. Prior art includes implementations by CRI, Convex, and Edison Design Group.

Gwyn asked what are the C++ issues? Just another keyword.

Guilmette asked if `restrict` was syntactically or semantically a type qualifier. Both–it obeys all the semantics of type qualifiers.

Jaeschke quoted an excerpt from Stroustrup on why restricted pointers were rejected for C++. Plum noted that the restricted proposal was rejected in the extensions subgroup and never brought to the full committee for discussion; do we want to try again, or just leave to implementors to add? The consensus was to not pursue this again in C++ now, although there was some sympathy for repudiating Stroustrup's comments to J16.

Meyers observed that Ritchie supports the proposal, which is important because of his opposition to the earlier "noalias" proposal.

Tydeman reported that IBM has a competing proposal, which is a "`#pragma disjoint`" directive. Guilmette noted that pragmas are not standard.

Plauger stated that the restricted pointer proposal represents the pragmatic solution to the "noalias" problem–the part of the semantics that we understand. We really should try to think this through to a complete "noalias" solution for the future.

SV In favor of considering the restricted pointer proposal for the next revision. 20 Yes. 0 No. 2 Don't know/care.

*** MacDonald will draft a proposal for adding restricted pointers to the next revision.

6. Defect Reports

6.1 Review of Proposed Responses [N377/94-062]

Plauger said that at its Tokyo meeting WG14 had postponed an official vote on the proposed responses to defect reports until J11 had reviewed them. All responses would become Record of Response 2 (RR2), with the subset of proposed corrections to the Standard becoming Technical Corrigendum 2 (TC2), if approved.

Martin indicated that BSI has some additional 25 DR's to submit from Clive Feather. These would become RR3.

Plauger indicated that we need to (1) review N377 for correctness, (2) vote to submit N377 to ISO as RR2 and TC2, and (3) decide how to handle future DR's–publish another RR/TC or just tackle them as information for the new revision.

Martin suggested it is easier to edit them in as part of the revision. Jaeschke suggested an RR3/TC3 be issued in two years, corresponding to the revision. Plauger informed us that the project editor is required to keep a document with all the changes folded in. Guilmette objected to postponing release of RR/TC's for such a long time; the Standard needs to be updated regularly just as software is. Plauger then suggested that we may in fact be required to issue RR/TC's more regularly, but perhaps we could be more officious in enforcing formal procedures for submitting DR's. Plum believes the requirement is to respond within two years; he would also like to question Feather about misusing the DR method–should be used for commercial need, not editorial fixes–and it would be better to handle editorial fixes as part of the revision process.

Plauger agreed that WG14 needs to respond in reasonable time to DR's, but we could postpone issuing the current responses for another year, for example. Stanberry suggested that we issue what we have, rather than risk spending committee time revisiting the same DR's; also we should actively encourage no more DR's, suggesting proposal be submitted for the revision instead.

Plauger reviewed who can sponsor DR's: WG14 member bodies and the ISO project editor, clarifying that he is the project editor for the 1990 Standard. Plum said we should clarify that we will still deal with real commercial

problems with the 1990 Standard in the interim while encouraging editorial issues to be handled as part of the revision.

SV In favor of closing out and publishing RR2 and TC2 of the current DR log.
Lots Yes. 0 No.

Plauger asked to determine the sentiment of the committee about the current practice of encouraging DR's. Walls asked for clarification that we would still handle all DR's as input to the revision. Yes. Plum added that we want to encourage revision proposals instead of DR's.

SV In favor of giving the C90 project editor more latitude to redirect submissions to national bodies rather than to sponsor as DR's.
Lots Yes. 1 No. 2 Don't know/care.

Martin reported that the published TC1 has errors! Six defect reports are missing, and one has a wording change that makes it invalid. Plum confirmed that the responses had been reorganized to be arranged by document section number rather than by DR sequence; we should insist that it be published in the form in which it was balloted.

Plauger said we should express concern/outrage that DR#22 was misworded and that the six DR's are missing. He clarified that this is an administrative error: the SC22 published version is correct but the Geneva (ISO) version is incorrect. He will handle as an administrative detail.

DR#67

Gwyn would like the second paragraph of (c) removed.

There was no objection. It was believed that was the intent from San Jose.

DR#69

Gwyn presented some objections to the rewording that had happened in Tokyo to parts (d) and (e) of the response. He stated that it is important to show some "holes" in the examples cited.

On part (d), there was discussion of whether all bits participate in char types. For example, are there bits that are not part of the value but are copied by **memcpy**? Gwyn argued that there is a fundamental understanding that there can't be any unused bits in an unsigned char type that are used in a larger integral type. The response needs fine bit of clarification to guarantee that you must be able to copy a value as if copying a sequence of unsigned char values.

On part (e), the issue is whether all bit patterns represent values. Plauger stated that he is convinced after many discussions that there can be bits in an object representation that are ignored in the value representation. Gwyn

agreed that the value bits are a subset of the bits of an object representation; he would like to see "value representation" changed to "object representation" in the response.

Plum introduced discussion of representations used in some implementations for uninitialized values–i.e., are there "special" values that might use bits of an object representation that are not part of the value bits? With respect to unsigned representations using n bits, the answer is no, unless outside the C model, there are no leftover values. But with signed representations, there is a left over value that some implementations use for a special value (e.g., -0 for one's complement, or the most negative value in two's complement).

SV How many persons believe that, in signed integer representations, all of the 2^n bit patterns for n bits must represent values?
5 Yes. 10 No. 7 Don't know/care.

Plauger stated that in our response to (e) we should be honest and say that we don't know/agree that the Standard gives clear guidance on this.

There was no objection to changing "value" to "object" in response (e), and Gwyn will draft text to include additional wording about special values in signed integer representations. Secretary's note: I didn't record this as an action item, and I didn't find the proposed text for this to include in the minutes; further, my notes don't indicate any changes agreed to for part (d); was I sleeping again??

Gwyn added that we should look at the wording in the "mem" function descriptions as well, especially for the revision.

Plum indicated there was a C++ liaison issue with respect to value collapse and signed chars. C++ will require no value collapse when copying char arrays with one's complement representations. (That is, C++ does not want loops that copy char arrays to convert `-0` bit pattern to `0` bit pattern in one's complement.) Gwyn suggested that C++ should define the semantics of such loops as if calling mem functions and then define the mem functions to work as if with unsigned chars. Plauger strongly opposed any endorsement of C++ using signed chars where they expect unsigned char semantics. Plum asked if anyone knew of any implementations that would have a problem with the C++ semantics for chars; could he report no empirical evidence found that there was a problem with this? Plauger, Farance, and MacDonald all strongly argued that we should not endorse this because there are theoretical problems, and we shouldn't give C++ license to assume otherwise. Gwyn added that cautious programmers should not write:

```
char *p, *q;
while (*p != 0) *q++ = *p++;
```

since the loop could terminate early. The consensus was not to tell WG21 that there is no problem with this.

DR#76

Plum reported that the current wording states that `&a[10]` requires a diagnostic while `a + 10` does not. We should either fix the Standard to relax the constraints on `&` such that one is allowed to get the address of "one too far" for an array, or we should say that we are deferring this as too murky of an issue.

MacDonald believed that the intent of the Standard was to require that an implementation be able to represent the byte following the array, but not necessarily the one before.

Gwyn stated that this is surprising and difficult to explain; we do not require an object to be there, but we could fix this by making `a[10]` an lvalue denoting an object.

Plauger clarified that all that is required is one byte that is addressable but not accessible beyond the array. Also, we left it deliberately OK to use `*` with an lvalue that is not necessarily an accessible object in the type system. The issue is addressable vs. accessible.

Gwyn added that `&*` is always allowed.

Meyers agreed with the conceptual model, but we need to be careful with correcting 6.3.3.2. Our response should be phrased as "We meant to say ..." and then make corrections so that a diagnostic is no longer required.

Plum crafted the following correction to the first sentence of the Constraints for 6.3.3.2:

> The operand of the unary `&` operator shall be either a function designator or an lvalue that does not designate a bit-field object and is not declared with the **`register`** storage-class specifier.

Plum believes this correction would require the following change to the definition of "lvalue" as well: the "one-too-far" location was meant to be addressable, but there is not an "object" at that location. The first sentence of 6.2.2.1 should be changed to:

> An *lvalue* is an expression (with an object type or an incomplete type other than a void type) that might designate an object.

Gwyn suggested that a qualification be included in the response that the Standard is currently clear that the example program is not a strictly conforming program, and that a diagnostic is required; then we should explain that this was not our intent.

Plauger suggested we should think about this and not fix it now.

Mooney noted that this response would invalidate our response to DR#12 in RR1.

Meyers expressed concern about the change to the definition of lvalue.

Jaeschke asked if the consensus was to go with this correction as the response with Gywn's qualification. Plum and Benito both argued in favor of a TC to remove the requirement for a diagnostic. After more discussion leading no closer to consensus, Gwyn suggested leaving this as an open issue.

SV In favor of leaving DR#76 as an open issue in RR2?
Lots Yes. 1 No. 4 Don't know/care.

Plauger added that there was precedence for this in RR1.

DR#78

Gwyn suggested that we fix the example by including `<string.h>` to make this a valid program. There was no objection.

DR#84 and DR#104

Gwyn suggested we revise the response to DR#84 to clarify why it refers to "the size required" when that is not mentioned in the original question.

Guilmette pointed out that we need to consider DR#104 as well. DR#84 would actually be withdrawn if his proposed TC for DR#104 is accepted. His suggested TC would add words to the appropriate subclause to the effect that "Declarations of formal parameters and/or abstract declarators appearing in prototyped formal parameter lists other than those used to declare formal parameters for function definitions do not declare objects."

Plauger considered it rash to say that these are not objects; need to fix in same manner as we did in allowing the return type of a function to remain an incomplete type until called.

Meyers reminded us of where incomplete types are tolerated (§6.5.2.3, footnote 63). Plauger said we missed a few places where we should have allowed incomplete types, such as allowing parameter types to be incomplete. Meyers said if carried to extreme of "don't require type information unless it's used" this would require care that we don't introduce errors; for example, unused parameters in variable argument list where size is needed to locate the following argument.

Guilmette withdrew his proposed TC, and Gwyn withdrew his proposed edit to the given response in lieu of Plauger helping to find correct words. The suggestion is that the following TC be given as the response to DR#84.

Add to subclause 6.5.4.3, after "A parameter type list specifies the types of, and may declare identifiers for, the parameters of the function."

> A parameter may be declared with an incomplete type, so long as the type is completed before the function is defined (6.7.1) or called (6.3.2.2).

Jones pointed out that DR#84 and DR#104 differ only by having an identifier in the declarator, and that hence this fixes DR#84 but DR#104 is still undefined behavior. Plauger disagreed since this fix extends the kinds of things that can be in parameter lists.

There were no objections to the proposed TC.

DR#88

The responses to this DR were changed in conjunction with revising our response to DR#139. See the discussion under that item for the revisions.

DR#101

Guilmette requested that the response be clarified by adding references to the appropriate subclauses of the Standard: for initialization, 6.5.7, and for function returns, 6.5.3.

Benito noted that the citation from Subclause 6.3.2.2 is misquoted; it should read "... has a type that includes a prototype, ..."

There was no objection to making these corrections.

*** Guilmette will draft wording for DR#101 corrections and send them to Plauger.

DR#103

Guilmette requested that a TC be used to clarify the intent of the Standard. Plum wanted a strong show of hands to endorse the response as is with no TC.

SV In favor of changing the response to DR#103.
1 Yes. Lots No. 7 Don't know/care.

DR#106

Guilmette argued that the proposed response fails to explain what is meant by a "use" of the value of an expression.

Benito stated that "use" is used throughout the Standard and he believes it is well-understood. Jones believes this is an excellent question, but can't be

answered in the short term. Plauger agreed that this needs fixing but not now. Plum added that "evaluation" demand constitutes use but there are lots of ripples in getting that right everywhere in the Standard.

Jaeschke suggested that the DR response be fixed by changing the last statement of the proposed response.

SV In favor of striking the last paragraph of the response to DR#106.
14 Yes. 4 No. 3 Don't know/care.

Plum suggested that we agree among ourselves to consider this in the revision process.

DR#107

Guilmette suggested some editorial wordsmithing for the response to this DR. For (b), change "may be" to "results in" in the second sentence.

There was no objection to this edit to response (b).

For (c), change "...in its treatment of programs that violate this requirement." to "... with regard to such conversions, which may or may not take place." In the discussion on this, it was proposed that response (c) just be changed to "No." We voted on this, but this was inconclusive. Then we voted on Guilmette's proposed wording change. When that was equally inconclusive, we voted another time on the proposal to change response (c) to "No."

SV In favor of changing response (c) to "No."
7 Yes. 4 No. Lots Don't know/care.

SV In favor of Guilmette's proposed wording change for response (c).
5 Yes. 4 No. Lots Don't know/care.

SV In favor of changing response (c) to "No."
14 Yes. 2 No. 4 Don't know/care.

DR#108

Guilmette reported a typo in the example. Need to insert

```
#undef double
```

after the include of `<math.h>`. There was no objection.

DR#109

Guilmette asked if someone could explain why the given response is correct? MacDonald agreed that it doesn't answer the question of whether or not the code must be successfully translated.

Guilmette suggested that the third sentence of the response was irrelevant and should be stricken; it presumes that this code is not strictly conforming, and that is not the case.

Gwyn asked what was wrong with the San Jose response? It was reread:

> The presence in a translation unit of an expression which would trigger undefined behavior *if evaluated* may not be used as an excuse for an implementation to fail to "successfully translate" the give translation unit unless the expression in question appears in a context where a constant expression is required. (See subclause 6.4; Semantics, and the associated footnote.)
>
> For an expression which is defined to yield undefined behavior (e.g. 1/0), if the expression appears in some context where a constant expression is *not* required, the undefined behavior actually arises *only if* the expression in question would be evaluated (at run-time) according to the abstract machine semantics. (See subclause 5.1.2.3.)

Plauger stated that this was OK so long as we allow a translator to not translate if it can determine that the code would be executed. Plum suggested that we strike the third sentence and reinsert the San Jose wording. Walls recalled the Derek Jones in Tokyo had objected to requirements on the quality of implementation, which is why the third sentence was added.

Guilmette added that he would like the first two paragraphs of the Tokyo meeting deleted as well. There was discussion of "undefined behavior" and "indeterminate value" which concluded that the first paragraph of the Tokyo response was needed. Farance and others objected to the wording of the first paragraph from the San Jose response. Meyers and others wanted to keep the second paragraph of the San Jose response since it is useful to allow implementations to issue a diagnostic, and it was also clarifying with respect to phases of translation.

Guilmette, Gwyn, and Farance drafted revised wording from the San Jose wording to replace paragraph three of the Tokyo response. Before presenting this to the full committee, however, Jones disagreed with the revised response and proposed a revised revised revised response:

- Delete the third paragraph of the WG14/Tokyo response.
- Add the following to the bottom of the WG14/Tokyo response:

> If an expression whose evaluation would result in undefined behavior appears in a context where a constant expression is required, the containing program is not strictly conforming.

> Furthermore, if *every* possible execution of a given program would result in undefined behavior, the given program is not strictly conforming.
>
> A conforming implementation must not fail to translate a strictly conforming program simply because *some* possible execution of that program would result in undefined behavior.

Martin asked if "every" possible execution should be "any" possible execution. Jones clarified that "every" was the correct qualification.

There was discussion of whether every possible execution path could be determined. Jones noted that a strictly conforming program could not change because of input value. Plum asked what about "`1/(argc - 3)`" where `argc` is 3? Proper way to describe "strictly conforming" is as characterization of input plus the program; the current definition of "strictly conforming" does not have any words about the input, so we should consider adding these for C9X.

MacDonald wanted the response to clearly answer the question with an explicit statement saying the example must be translated. After more discussion, it was agreed to add:

> Because `foo` might never be called, the example given must be successfully translated by a conforming implementation.

There was no objection to the Jones/MacDonald revision.

Walls observed that "`i`" needs to be declared in the example program.

Guilmette asked if WG14 would accept the revised response or if we needed to vote here and then wait for their approval. Plauger said that if we don't reverse their decisions, it's incumbent upon us to vote out these responses. Gwyn added that the Tokyo WG14 responses were improvements to the previous responses!

DR#116

Guilmette asked that the response give rationale for the answer. It was proposed that "See 6.3.3.2 Constraints" be added to the response.

There was no objection.

DR#118

Guilmette argued that the response does not address the primary issue of this DR which is, "At what point does an enumeration type become a completed type?" He believes we need an equivalent statement for enums as

for structs and unions, namely, that the type is completed at the closing brace. Also, `sizeof` is irrelevant here.

Gwyn stated that the relevance of `sizeof` is that it is only defined when the type is complete. Also, saying the type is complete at the closing brace is too limiting; we need to give some latitude to implementations that may in fact complete the type before the closing brace, depending on how they determine the representation.

Guilmette would accept "no later than the closing brace"; it doesn't have to be "exactly at".

There was general consensus that we should fix this, so the issue became whether we should fix it now or as part of the revision.

SV In favor of fixing now.
9 Yes. 5 No. 8 Don't know/care.

Plum said the proposed wording goes too far; it would require a diagnostic for sizeof in the given example. Gwyn agreed, and it would also require a diagnostic for the example for DR#13.

SV In favor of intent of enumeration types being complete by the closing brace but undefined whether they could be complete before then.
21 Yes. 0 No. 1 Don't know/care.

There were several attempts at wordsmithing. Then it was decided to leave this to a drafting committee (of one).

******* Plum will draft appropriate wording for a TC to resolve DR#118, reflecting the intent that an enumeration type is complete by the closing brace of its declaration.

DR#120 and DR#122

Guilmette asked if the committee members believe 3-bit integer types exist? Plauger replied that this is too vague; bit field width is like lvalueness property. Gwyn noted that there is no such thing as a bit field type in the C90 Standard (6.1.2.5). Plauger asked, what is a type? That which determines representation and values and operators; and that includes bit width.

There was debate on whether to leave these two DR's as open issues. Plum suggested that we should leave the responses as is (there is little debate on the functional answers) and just acknowledge that the wording in the Standard could be improved. Gwyn agreed but suggested that we amend response to DR#120 so that the second sentence reads "... can be informally described as ..."

Guilmette agreed since everyone else agreed.

DR#125

Guilmette proposed that the clause "as explained in the response to Defect Report #012, Question 1" be stricken from the response. There was no objection to this change.

However, there was discussion about Guilmette raising these (and similar issues) at this time in the process. Guilmette noted that at the San Jose meeting he was not a voting member and had been restrained from voicing some of these concerns then, and that since he could not attend the Tokyo meeting, this was his first opportunity to raise some of these issues.

DR#132

Gwyn suggested that we restore a cross reference from DR#132 to DR#109 with the revised response to DR#109, and this was accepted. Add "The response to DR#109 addresses this issue." to the response for DR#132.

DR#139

Jones expressed a strong desire to change our response so that we fixed the Standard. He argued that we forgot to put into the Standard that an incomplete struct type is compatible with any struct type declared in a separate translation unit; it was always the intent that this was allowed, but never stated; and this type compatibility is useful for information hiding.

Meyers agreed with the proposal to fix this, and gave an example of typical use of this assumed compatibility. Stanberry also agreed–codes are using this now, but if it isn't in the Standard, it isn't portable.

MacDonald wanted clarification that the access to the struct members is through pointers and access functions. Yes.

Gwyn asked if we allow incomplete array types to be compatible with complete array types across translation units. Yes.

Plauger stated that he had reviewed 6.1.2.6 and agrees with the sentiment of the proposed TC. However, both Plauger and Plum suggested that we wait for C9X to fix. There was repeated discussion about the current use and not wishing to wait to fix it. Jones voiced particular objection to the response saying the behavior was explicitly undefined.

There was discussion of the fix not addressing rules for structure tags in type compatibility. Gwyn said that we don't want all struct pointers to be compatible. MacDonald added that we don't want to cripple optimizers with wording such that an implementation can't identify when two struct pointers

point to different structs. Plauger said we need to be careful not to give away the store to attain the middle ground.

Jones drafted a proposed fix which would change the third sentence of 6.1.2.6 as follows:

> Moreover, two structure, union, or enumeration types declared in separate translation units are compatible if at least one is an incomplete type or if they have the same number of members, the same member names, and compatible member types; for two complete structure types, the members shall be in the same order; for two complete structure or union types, the bit-fields shall have the same widths; for two enumeration types the members shall have the same values.

Jones reiterated the following things to consider:

- Tags are not part of the current type system–an incomplete struct type would be compatible with any struct type declared in another translation unit, complete or incomplete, regardless of whether the tags are the same or different.

- Since the only thing you can do with an incomplete struct type is to declare a pointer to it, this is an explicit statement of the "all struct pointers are the same" rule.

- This also implies there are no optimization issues since creating or accessing an object requires a complete type and the rules for compatible complete types would not change.

- Type compatibility is already nontransitive: two incompatible types may both be compatible with a third.

- Would not preclude changing the type system in the future to include tags, but no desire to do so now.

MacDonald repeated his concern for the effect on optimizers–allowing possible aliasing of structs. Others observed that the proposed TC doesn't sneak in any tag equivalence, although allowing an incomplete struct type to be compatible with any struct type in another translation unit blows a large hole in the purely conceptual type checking mechanism.

Plum noted that C++ uses tag equivalence so we don't want to introduce any problems with their type checking.

MacDonald asked for clarification that the following example would be allowed with the proposed fix.

y.c

```
f(struct foo *,
  struct bar *) {
...
}
```

x.c

```
struct tag *p;
struct foo;
struct bar;
f(p,p);
```

Yes. It was agreed that this was the best we can do with the current situation.

SV In favor of accepting Jones' proposal for a TC as our response to DR#139?
17 Yes. 0 No. 4 Don't know/care.

SV In favor of stronger tag checking in next revision?
17 Yes. 0 No. 4 Don't know/care.

Jones then discussed the impact of a TC for DR#139 on our responses to DR#88. We would need to change the following responses as indicated:

(a) No.
(b) Yes, see response to DR#139.
(e) Yes, yes, and no.

SV In favor of these revised responses to DR#88?
16 Yes. 0 No. 3 Don't know/care.

*** All edits for RR2 and TC2 need to be sent to Plauger by the end of the year.

*** Benito, Farance, Guilmette, Gwyn, and Martin will serve as review committee for RR2/TC2.

6.2 New Defect Reports

No new defect reports have been received; N377/94-062 represents the current defect report log.

7. Review of X3J11 Technical Reports

Jaeschke introduced this agenda item by reiterating that he would like to close out the TR by this meeting. Our charter expires at the end of the year and we don't want continued work on NCEG issues to interfere with the revision process. Gwyn added that the TR will have more visibility than proposals for actual revision.

Variable Length Arrays [N317/94-001]

MacDonald reminded us that there were other documents besides the CRI proposal. Cheng's proposal, N384/94-069, provides a good introduction to the issues.

Jaeschke reviewed some history for this subgroup: we adopted the CRI proposal as "a" contribution to the TR. Do we want to include other contributions in the TR?

MacDonald stated that the problem with Cheng's proposal is his description is in terms of other C^H features. Thomas asked if Cheng has put forward his proposal as a candidate for the TR. There was discussion of what was required to get proposals in the TR. Gwyn said the proposal should be acknowledged. Tydeman suggested that then fat pointer proposal should likewise be acknowledged.

Plum said that from a technical rather than a procedural view, having more than one VLA proposal in the TR implies that we still haven't made up our mind. Stanberry added that we have to decide what is the weight of the TR; if it is to give guidance to vendors, it is confusing if we can't decide. Gwyn believes there is no problem if two proposals are orthogonal; they would be technically compatible.

Kwan expressed concern that we need some compelling reason to change our mind this late.

MacDonald clarified that the CRI VLA extension follows the FORTRAN 77 model, but the FORTRAN 90 model uses dope vectors.

Farance said he is withdrawing his prior VLA work from consideration for the TR, but he will propose it for consideration for C9X.

Gwyn asked for clarification on the use of C^H. It is not widespread, used mostly at UC Davis.

Plum said we should decide what we are going to do with the C revision before closing the TR. At the Kona meeting, he thought we should move to the dope vector model. For parameter passing, isn't the no-dope-vector solution just an optimization of the dope vector solution? The only functional difference between the two is with VLA's as struct members. MacDonald clarified that VLA's are not allowed as struct members in the CRI proposal. Plum suggested that if the proposal was not overly specified, could leave it open to allow a dope vector to be stored as part of the value of a VLA struct member. He also noted that C++ has added a valarray class and a vector template to its library.

Farance argued that the real issue with argument passing is whether passing by value or by reference. The CRI proposal requires shape information to be

passed as separate arguments; the fat pointer proposal has the compiler pass the information for you. Another possibility is that the shape information is external or global.

MacDonald believes the dope vector approach is orthogonal to the CRI proposal, and could coexist; FORTRAN 90 supports both kinds of VLA's. Gwyn said that it is good to recognize there is more than one way to do some things and it may be better to allow options rather than insisting on only one way.

Plum stated that C++ could never adopt the no-dope-vector approach, but if C and C++ are really going to diverge, it won't matter. VLA's, DPCE, and ALO's address different markets; it's too bad we couldn't find one solution for all. Farance said that we wanted separable features, though. MacDonald said the goal of NCEG was to answer "What would C look like for numerical programming?" Gwyn said there are enough users who need better arrays than what C currently offers; we should look at general solutions if possible, but maybe there aren't any. Plum wondered if it would be possible to find "the" array proposal for the revision process.

There was a discussion of implicit versus explicit shape passing. There is overhead for implicit handling, but it was generally agreed that it is relatively small for most functions.

Meyers expressed his preference for the dope vector approach since he believes the explicit shape passing is more error-prone.

Farance commented that he really wants a facility to describe what he needs for performance/optimization/etc. One problem with dope vectors is that they don't fit into a void *. With respect to whether we should allow more than one paradigm, he said we already do in allowing both pass by reference and pass by value.

Gwyn clarified that C only has pass by value.

Stanberry suggested that we need to chose a sufficiently low level paradigm with tools to build the complexity that is needed specific to each market.

Plum thinks we should clarify that, as with classes, we are just inviting proposals but are still debating what to accept.

SV In favor of seeing the CRI VLA proposal considered for the next revision. 13 Yes. 0 No. 5 Don't know/care.

*** MacDonald will draft a proposal for including the CRI VLA proposal in the next revision.

Complex [N397/94-082]

Thomas gave a presentation on the state of this proposal.

- History
  - Original NCEG focus group
  - Early drafts followed traditional FORTRAN style
  - IEEE consistency/efficiency problems: `∞ * i => NaN + ∞i`
  - Imaginary types: `∞ * i => ∞i`
  - FPCE consistency
  - Mathematical notation: `-I*asin(I*Z)`
  - C++ compatibility
- Imaginary types
  - Natural modeling for complex analysis
  - IEEE consistency for special values: `∞ NaN -0`
  - Definitional efficiency: `real * imag => imag, imag/imag => real`
  - Storage/algorithmic efficiency
- Overview
  - **<complex.h>** adds
    - new basic types, where complex types are represented with two reals and imaginary types are represented by one real:

`float_complex`	`float_imaginary`
`double_complex`	`double_imaginary`
`long_double_complex`	`long_double_imaginary`

    - Imaginary unit `I` such that `I*I == -1`
    - Conversion among complex, imaginary, and real types
  - Usual arithmetic conversions for operand parts, but no automatic conversion among kinds
  - Expression typing to appropriate kind (real, imaginary, or complex)
  - Expression evaluation methods apply
  - Complex functions
  - Branch cuts and ranges
  - No special cases specification for library functions
  - Overloading as in FPCE

Gwyn asked how were the included complex functions chosen. Used FPCE plus Tydeman's earlier document plus other recognized useful ones. Some functions were omitted because it was unknown how to specify them. Gwyn hoped that as experience is gained, other functions might be added. MacDonald observed that some vendors might object to functions that are not in FORTRAN being included.

Guilmette asked if it was over-specifying to say that the macros defined in **<complex.h>** are in the implementation's namespace? Not in the context of TR which is guidance to the implementor. Both Guilmette and Gwyn suggested that this wording be moved to rationale so that we don't constrain an implementation. Note: have to leave the expansion of `I` to `_Imaginary_I`

in the implementation namespace so that a user can `#undef` it if needed. Thomas agreed to make this editorial change.

Tydeman discussed his working document showing the code necessary to implement this proposal for complex. He is also working on a full IEEE implementation of complex, but expects it will be too costly in terms of performance to be desirable. There was discussion of the complexity of implementing some operations. For example, Tydeman has four algorithms for complex multiply for IEEE conformance. Need a flag to detect which one is invoked; gives different edge case results with respect to exception flags but not with respect to values. This affects portability.

- C++ compatibility issues
  - Type names
  - C complex > C++ complex
  - Mixed mode?
  - Conversions?
  - Wide evaluation?
  - Integral arguments?
  - Single source for common subset
  - Imaginaries

Plum said now we have two implementations of complex–C++ and this.

Plauger noted that the C++ complex proposal with templates and implicit conversion has broken the C++ complex model. We shouldn't try to be compatible with this moving target.

- To do
  - Incorporate Plano comments
  - Review mathematics
  - Update references
  - Write foreword
  - Get editorial review
  - Submit TR
  - Draft C9X proposal

Gwyn said this is not as polished as other TR proposals, but he is confident that the math pieces that need to be completed are in place and that the proposal after review will be correct. Tydeman agreed since the complex proposal is consistent with the FPCE work.

Guilmette was concerned that endorsing this proposal will be construed as also endorsing overloading. Others shared this sentiment; this should not be construed this way.

Jaeschke asked if, given that the math review would not lead to substantive changes but only editorial ones, does the committee feel comfortable empowering Thomas to continue?

MacDonald stated that he has been instructed to vote against this proposal since CRI believes this is more than what is needed and is too costly for the perceived benefit.

Plum asked what would prevent `typedef`'ing imaginary to complex, assuming that an application was not worried about special values. He suggested that any parts of the proposal that would preclude this should be removed. He hopes that this would prevent the committee from remaining hopelessly polarized on this issue. Several instance were noted, alas, where it is not just a representation issue, but real semantic problems would result.

MSP Move we accept N397 as amended by the edits agreed to at this meeting and subject to review committee approval for the TR. (Thomas, Tydeman)
15/3/0/2/20

*** Guilmette, Keaton, and Tydeman will serve as review committee for edited complex proposal.

*** Thomas will produce an edited Complex proposal.

MacDonald reiterated that without NaN's, infinities, and signed zeroes, an optimizer can handle the edge cases without an imaginary type. Also these values don't behave the same in the real plane as they do in the complex plane. Gwyn said there were other reasons, such as representation, for including imaginary types, not just edge cases.

Meyers stated that the TR allows experimentation but we need more time to evaluate before making this part of the Standard; need feedback from real users. Gwyn agreed but we need to start work now and gain the experience while we're in the revision process.

Plum stated that he is sensitive to efficiency issues as well as IEEE implementations. Jaeschke observed that if we start with a subset, it will be easier to evolve to add more features rather than to fight to remove some. MacDonald stated that he would support considering a complex type with an imaginary type for the revision. Guilmette added that he would support a complex proposal without overloading.

Thomas believes it might be possible to try the `typedef`'ing of imaginary to complex, although there would be some subtle differences in behavior. But removing imaginary type would not be acceptable as it would be too far-reaching.

SV In favor of adopting the complex proposal as in the TR for consideration for the next revision.
9 Yes. 5 No. 7 Don't know/care.

SV In favor of some form of complex support considered for the revision. 19 Yes. 1 No. 2 Don't know/care.

There was discussion on how to proceed. We need more experience and possibly more proposals. Martin suggested that we revisit these votes after time to discuss and think about these issues. Stanberry would like to see a proposal addressing the typedef mapping.

Floating Point C Extensions [N364/94-049]

Thomas presented six edits to this proposal. The first five were purely editorial. There was no objection to these.

The sixth edit changes the promotion rules so that integral arguments are promoted to the wider of the minimum evaluation format or the widest floating argument. This yields better performance on two-argument functions. There was some discussion of cases before and after the change. The motivation for the change is to be more consistent with the model. There was no objection to this edit.

Jaeschke will consider these edits as part of the document to be considered for the TR.

*** Thomas will produce an edited FPCE document.

Data Parallel C Extensions [N383/94-068]

Keaton reported that the DPCE subgroup had a very productive meeting in Boulder in September, basically resolving the remaining open issues with the proposal.

Stanberry presented the changes to the document as a result of the decisions reached at the Boulder meeting. The following new features were added in the indicated document sections:

- (3.3.1) The "." was added as a primary expression, allowed only in parallel indexing expressions, as a synonym for an appropriate call to `pcoord`.

- (3.3.3.6) Parallel indexing was extended to allow slicing. Slicing was based on the FORTRAN 90 model, but with some restrictions.

- (3.3.3.7) Specification of the result of reduction operators when no positions of an operand are active; a default result for each operator was added.

- (3.5.4.1) Pointer declarators are allowed to be qualified as `elemental`. This makes the referenced type parallel when the pointer type is parallel.

MacDonald asked for the motivation for elementally-qualified pointers. This was added to support parallel linked lists. Gwyn asked if this implied any run time burden. No.

In further discussion of elemental functions, MacDonald asked how parallel pointer arguments are handled. Keaton explained that this is handled by compiling elemental functions twice: once for a parallel version, and once for a serial version. The compiler can detect which should be invoked for a given call.

• (3.5.4.4) New run-time terminology was added for shape objects (fully assigned, partially assigned, and fully unassigned) to aid in defining semantics of library functions.

MacDonald suggested that "acquired" might be a better term than "assigned." Gwyn asked how shape objects acquire their values. This is done either at declaration time or via assignment.

• (4.14) New types `dpce_layout_specs_t` and `dpce_layout_t`, and new functions `layoutof`, `nodeof`, and`nodepositionsof` were added to `<dpce.h>`.

• (4.14.1) The redefined functions from `<stdlib.h>` were extended to include elemental versions of `calloc`, `malloc`, and `realloc`.

Gwyn asked for clarification of whether DPCE requires changes to these standard header files. No, the functions redefined in `<dpce.h>` are defined to be compatible with the functions defined in the standard header files. Gwyn asked if one can invoke, for example, `cos` on both parallel and nonparallel? Yes. Keaton added that the syntax for elemental and nodal function qualifiers uses the C++ function prototype syntax, which allows qualifiers after the parameter specifications. Gwyn asked if any existing functions get changed by DPCE. No.

Stanberry summarized revisions to the following document sections from decisions reached at the Boulder meeting:

• (3.3.2.2, 3.5.4.3, 3.6.6.4, 3.7.1) Completed nodal and elemental function specifications and examples.

• (1.6, 3.1.2.5, 3.2.2.3, 3.3.2.1, 3.3.3.2, 3.3.4, 3.3.6, 3.3.8, 3.3.9, 3.5.4.1) Revised and completed "parallel pointer" and "pointer to parallel" specifications and examples.

• (3.1.2.5) Completed extending types to include parallel and shape types, and pointers to parallel types.

• (3.1.2.6) Revised and completed specifications for compatible and composite types. Defined the composite type of an elemental-qualified function and a nonelemental-qualified function to be the elemental-qualified function.

• (3.2.3, 3.3) Clarified required compile-time shape compatibility checking (constraint violations that require diagnostics) versus run-time shape equivalence requirements (that result in undefined behavior).

• (3.3) Completed constraints sections for all operators.

• (3.3.2.1) Corrected and clarified subscripting of arrays to allow both parallel and nonparallel integral subscripts.

MacDonald asked what results from a parallel subscript of an array? A parallel subscript with an array results in a parallel value, essentially a multiple selector, similar to a slicing operation.

• (3.3.3.5) Changed **shapeof** to be an operator rather than a function.

• (3.5) Changed/restricted syntax for parallel object declarators.

• (3.5.4.4) Clarified **block** and **scale** layout specifiers.

• (3.6.4, 3.6.5, 3.6.6) Completed specifications for selection, iteration, and jump statements.

• (4.14.1.3) Replaced **salloc** with **newshape** function.

• (A) Completed Appendix.

Farance asked if **memcpy** could be used to copy a parallel object to/from another parallel object. Although there is an **elemental memcpy** that allows copying between parallel objects of the same shape, there is no support for changing layout through this function. Dynamic layout changing requires copying elements explicitly. Farance believes dynamic layout support through **memcpy** should be supported, but the DPCE subgroup chose not to do this.

Plum asked about commercial interest in the DPCE proposal, and what has been implemented. Besides the implementation by Thinking Machines Corporation of the C* language, there are implementations at University of New Hampshire and at Oregon State University. Pacific-Sierra Research is providing a preprocessor that will translate DPCE into C, similar to how they have implemented HPF. And Maya Gokhale of David Sarnoff Research Center is working on an implementation. All features except nodal functions have been implemented so far.

Plum also reported that he had been satisfied from a response from Phil Hatcher to questions regarding the role of DPCE: it is not intended to be on

the evolution path of future C, but is clearly aimed for a specialized audience. Hence, it is more important to be compatible with C* than with C++. Keaton confirmed that the DPCE proposal would be included in the TR but not considered for the next C revision. Plum added that if any new array syntax is proposed for the next C revision, Hatcher suggests that this can be made illegal to mix with old array syntax.

Farance asked what had happened to the proposal to change the syntax of parallel or left indexing as suggested at the San Jose meeting. Although this had been discussed at the Boulder meeting, the issue of C++ compatibility was not compelling (using function-call-like syntax changes the precedence of parallel-indexing and is only one point of many where DPCE/C++ would differ) and no other acceptable syntax that provided a clear distinction of this operation from array subscripting was found. Further, this type of change would break existing C* code. Hence, the syntax for parallel indexing was maintained as in the C* base document. Farance still considers this to be a major problem stylistically with DPCE. Gwyn suggested that if this is a matter or preference, it is not a proper subject for debate here.

Gwyn asked if there were any ongoing substantive changes to be made to the document before it is submitted for the TR. Stanberry reported that there were some minor, but substantive, changes to be proposed at this meeting for the layout directives, but no other open issues were still to be debated.

As technical editor, Stanberry requested latitude to add clarifications as well as fix typos, grammar errors, and other errors of a nonsubstantive nature in preparing the final copy for the technical report, including incorporating any suggested changes from this meeting. The consensus was that this was assumed for all editors. A review committee will be requested to check all such changes.

Martin raised two issues. He asked if the committee had considered changing the maximum size of an "object" (2.2.4.1). Since parallel objects are derived from C objects, the maximum size of a C object does not need to be changed. He also asked why many of the library functions could not return, say, -1 instead of having undefined behavior; this would allow users to recover from errors gracefully. Although desirable, these functions have undefined behavior because of an incompletely assigned or acquired shape object's value; since there is no way to detect that the shape value is unassigned, the best that can be said about these functions is that the behavior is undefined.

Jaeschke suggested that the "history" statement in the introduction be removed from the DPCE (and other TR documents) as he will provide a suitable history overview in introductory cover sheet for the TR.

Stanberry presented suggested edits to document section 3.5.4.4, to revise the block and scale layout specifiers.

- Add constraint on **scale** expressions (page 72, line 30):

  All **scale** expressions shall be greater than zero.

Gwyn observed that the constraints on the values of **block** and **scale** expressions are really part of the semantics. It was agreed that they should be moved from the constraints section to the semantics section.

- Add constraint on mixing **block** and **scale** specifiers (page 72, line 25):

  If any *shape-dimension-and-layout-specifier* includes a **block** layout specification, then no other *shape-dimension-and-layout-specifier* shall include a **scale** layout specification, and vice versa.

- Revise **scale** semantics (page 72, line 53):

  The **scale** specifier suggests that data be partitioned into maximal blocks with block dimensions proportional to the scale expressions in the corresponding dimensions; each partition receives the indicated block size, if the shape is multidimensional; or a single block if the shape is one-dimensional. Partitions of blocks are distributed in round-robin fashion to the nodes of the target architecture.

- Add rationale (page 73, line 3):

  Note: if the number of partitions is not a multiple of the number of nodes, not all nodes receive an equal number of partitions; if the data size is not a multiple of the determined block size, not all partitions are necessarily the same size.

MacDonald stated that CRI had abandoned its layout directives which DPCE had adopted here; they turned out to require a lot to implement and it wasn't really used by applications, so basically it was viewed as overkill. Stanberry noted that these directives are, like **register** or **volatile**, ignorable by any implementation. Plum asked if C* had layout construct? C* does have an **allocate_detailed_shape** function, but doesn't have these layout features. Keaton indicated the layout directives DPCE has chosen are similar to what is in HPF. MacDonald wondered if this would be a portability issue. Layout is essentially implementation dependent, but providing these directives is an attempt to make this more portable.

Farance stated that a remaining issue is combining of features: have to buy all or get nothing. He also wants to see nodal and elemental functions integrated with sequence points. (This was briefly discussed at the evening meeting of DPCE. Stanberry and Keaton believe nothing additional is needed for nodal and elemental functions–they behave as do ordinary functions with respect to sequence points. Synchronization implied on return of nodal functions is already described in sections 3.3.2.2 and 3.6.6.4.)

*** Stanberry, Keaton, Van Sickle, Ream, and MacDonald volunteered to review (all or some of) the edited DPCE document.

MSP Move we accept N383/94-068 as amended and approved by reviewing committee for the TR.
15/1/0/4/20

*** Stanberry will produce edited DPCE proposal (version 1.6, N395/94-080).

Secretary's note: thanks to Tydeman for providing notes for the minutes for the DPCE discussions!

Extended Integers [N381/94-066, N386/94-071]

Kwan presented the latest version of the **<inttypes.h>** proposal, N381. He reported a typo on page 7: the definition of macro `PRId64` should be `"lld"`. He presented a couple of motivating examples.

```
main()
{
long L = -1;
unsigned int ui = 1;
if (L > ui ) printf("L greater than ui\n");
else printf("L not greater than ui\n");
}
```

This prints `"L greater than ui"` if `sizeof(int) == sizeof(long)`, and prints `"L not greater than ui"` if `sizeof(int) < sizeof(long)`.

```
main()
{
unsigned int mask = 0xFFFFFFF0;
int i = -3;
int j;
j = i ^ mask;
printf("j=%d\n",j);
}
```

The value of j will depend on the size of `"int"`.
A better way is to set `mask = ~0xF`.

Then Kwan reviewed the elements of the proposal.

- Purpose
  - Make C the programming language of choice
  - Provide a way to write portable code by:
    - Defining new integer types
    - Providing consistent properties and behavior
    - Easily implemented
  - Takes a minimalist approach
    - implementation limited to a single header

- little or no extensions to the language

- **<inttypes.h>**
  - Integers of exactly n bits

```
typedef ? int8_t
typedef ? int16_t
typedef ? int32_t
typedef ? int64_t
typedef ? uint8_t
typedef ? uint16_t
typedef ? uint32_t
typedef ? uint64_t
```

  - Integers of "at least" n bits
    - Signed integral types of at least n bits:

```
typedef ? int_least8_t
typedef ? int_least16_t
typedef ? int_least32_t
typedef ? int_least64_t
```

    - Unsigned integral types of at least n bits:

```
typedef ? uint_least8_t
typedef ? uint_least16_t
typedef ? uint_least32_t
typedef ? uint_least64_t
```

  - Other types
    - Most efficient (fastest) integer type

```
typedef ? intfast_t
typedef ? uintfast_t
```

    - Integer data types that are large enough to hold a pointer to `void (void *)`

```
typedef ? intptr_t
typedef ? uintptr_t
```

    - Largest integer supported

```
typedef ? intmax_t
typedef ? uintmax_t
```

There was discussion of the "fastest" type. Thomas recalled that we wanted the fastest types to be "at least" types. Keaton agreed–we had previously decided that we need to separate the "at least" types into "smallest" and "fastest-at-least" types. Kwan agreed to make this change.

There was also discussion of whether, if this becomes part of the Standard, it is strictly conforming to use. OK if mapped only to standard types. Note that for the TR version, however, a program which includes **<inttypes.h>** would not be strictly conforming.

- Limits
  - Signed limits

```
#define INT8_MIN  (-128)
#define INT16_MIN (-32767 - 1)
#define INT32_MIN (-2147483647 - 1)
#define INT64_MIN (-9223372036854775807 - 1)
#define INT8_MAX  (127)
#define INT16_MAX (32767)
#define INT32_MAX (2147483647)
#define INT64_MAX (9223372036854775807)
```

  - Unsigned limits

```
#define UINT8_MAX  (255u)
#define UINT16_MAX (65535u)
#define UINT32_MAX (4294967295u)
#define UINT64_MAX (18446744073709551615u)
```

Mooney noted that there were no "max" values for `intptr_t` and `uintptr_t`. These are needed as feature tests in case the architecture can't represent a pointer in any standard integral type.

Guilmette asked if we should be introducing macros when C++ is eschewing adding new macros. Yes, this is the only reasonable way to do this.

Plum asked if all the proposed types can map into those in Farance's type system. Yes.

- Formatted I/O
  - Provide a list of macros for output (**printf**) integers of various length

```
#define PRId8 "d"
#define PRId64 "lld"
#define PRIo8 "o"
#define PRIo64 "llo"
...
```

  - Corresponding macros for input (**scanf**) integers of various length

```
#define SCNd16 "hd"
#define SCNd64 "lld"
#define SCNx32 "x"
#define SCNx64 "llx"
...
```

  - Example

```
main()
{
uint64_t u;
printf("u=%016" PRIx64 "\n",u);
}
```

- Conversion functions
  - Only 2 conversion functions are needed:

```
extern intmax_t strtoimax();
extern uintmax_t strtoumax();
```

- Constants
  - Provide macros to define constants of a certain type

```
#define __CONCAT__(A,B) A ## B
#define INT16_C(c) (int16_t) c
#define UINT16_C(c) ((uint16_t) __CONCAT__(c,u))
#define INTMAX_C(c) ((int64_t) __CONCAT__(c,ll))
```

  - The suffix for `int64_t` constants is non-standard

- Notes
  - Promotion rules are the same as the underlying integral type
  - No support for endianess (bit and byte ordering)
  - May need the same for C++ compatibility
  - There could be more than one "most efficient" integer type in some implementations
  - libc library must use the same header as the application
  - Not all pointers are of the same size in some implementations

Plum asked if anyone noticed any C++ incompatibilities. Meyers said that they ship a version of **<inttypes.h>** now with both their C and C++; they also **typedef** to extended types to provide type checking. Plum said this suggests that a strong **typedef** (**typedef!**) be used for defining these types.

Gwyn suggested that the proposal use actual types rather than ?'s.

Thomas asked what can he do with these types? Convert to float? Typedef to other types? Yes, all the ordinary things you can do with integral types. The proposal should say that.

Plum stated that the proposal is acceptable as is for the TR, but should be reviewed more carefully for consideration for the revision.

*** Farance, Keaton, Meyers, Mooney, Thomas, Tydeman, Ziebell will review an edited **<inttypes.h>** proposal.

MSP Move we approve an edited N381 (Kwan's extended integer) proposal as "a" contributing component for the TR. (Kwan/Meyers)
17/0/0/3/20

*** Kwan will produce an edited **<inttypes.h>** proposal.

Farance presented his extended integer proposal, N386.

- What is the problem?
  - 64-bit types?
  - create long long?
  - Agree on mapping of sized to standard types?

None of these is the real problem!

- Concepts
  - Signedness. unsigned or signed.
  - Specified Precision. The number of bits required (specified) by the programmer.
  - Exactness of Precision. The specified precision requires exactly N bits or at least N bits. Conforming code uses at least precision and strictly conforming code uses exact precision.
  - Actual Precision. The number of bits that participate in the value. For exact types, actual precision == specified precision.

- Performance Attributes
  - Optimization for space. Provides "smaller" storage. Implementation-defined optimization for space.
    - Useful for applications where data must be stored in smallest storage, e.g., data bases.
  - Optimization for time. Provides "faster" speed. Implementation-defined optimization for time.
    - Useful for applications where fastest execution performance is desired, e.g., inner loops.
  - Unoptimized. No space or time optimization specified.

- Examples - what we believe is characterization of existing C types
  "short" is:
  ```
  int atleast:16
  ```
  "int" is:
  ```
  fast int atleast:16
  ```
  "long" is:
  ```
  int atleast:32
  ```

- Examples - programmer needs "at least 40 bits" precision (specified=40, exactness=atleast)
  - For API's, programmer uses the type:
    ```
    int atleast:40
    ```
    - compiler implements this type as 48-bit integer
  - When storing this value in data bases, the programmer uses the type:
    ```
    int small atleast:40
    ```
    - compiler implements as 64-bit integer
  - When used in inner loops, use the type:
    ```
    int fast atleast:40
    ```
    - compiler implements as 64-bit integer

  Performance attributes yield different actual precision.

- Loss of information
  - There are 96 possibilities (#specified bits X exactness X conformance X addressability X performance), ignoring signedness.
  - These get mapped into 23 scenarios in C. Loss due to:
    - C not having specification-based types
    - Different mappings for different implementations and architectures

- These scenarios get mapped into 5 "types": char, short, int, long, bit fields. Loss due to:
  - C not having specification-based types
  - Portability issues (type documents itself)
  - Reverse Engineering issues

- Code for some 32-bit implementation
  - Programmers RARELY document the intention of type. More often, programmers document the purpose of the variable.
  - Implementation-defined mappings to types imply that the code is not portable.

- Other proposals
  - "long long" – doesn't solve problem
    - makes it WORSE
      - With long long, 30 scenarios mapped to 6 "types" represents 20% coverage
      - With Standard C, 23 scenarios mapped to 5 "types" represents 22% coverage
  - Kwan Proposal
    - eliminates one scenario, adds 7 scenarios, adds 1 type
    - 29 scenarios => 6 "types"
  - C++ class libraries
    - implementation of all promotion rules requires several thousand prototypes in scope
  - Only solution is specification-based approach
    - no loss of information
    - allows programmer to get close to hardware

- Algorithm for determining correct type (simplified - ignoring signed vs. unsigned)
  - Inputs:
    - Specified Precision
    - Actual Precision of available types
    - Exactness – atleast/exact
    - Performance – space, time, none
    - Conformance – strict conformance/conformance
    - Addressability – can take address/not needed
  - Process:
    - see paper
    - determine which of 23 scenarios
  - Output:
    - The C "type", one of: char, short, int, long, bit fields
    - Access operation type:
      - usual (assign, arith ops, etc.)
      - bit mask (must do explicit mask)
  - Algorithm executed by: programmer, preprocessor, experiment program

- SBEIR - Type System – Axioms
  - Actual precision of two types differing only in their signedness must be the same.
  - Actual precision of types differing only in performance attribute:
    actual(small) <= actual(none) <= actual(fast)
  - Actual precision of two unoptimized types differing only in specified precision:
    if specified(N) >= specified(M),
    then actual(N) >= actual(M)
  - For two small types, the same is inferred
  - For two fast types
    if specified(N) >= specified(M)
    does NOT imply actual(N) >= actual(M)
    - Example:
      fastest at least 16 => 64 bits
      fastest at least 32 => 32 bits
  - Programmer can create new type based on actual precision
- C types mapped to SBEIR types
  - Each implementation must map the C types char, short, int, long (and their unsigned versions) to an SBEIR type.
  - Usual C relation must hold:
    short <= int <= long
  - must be true for both actual and specified precision
  - examples of mappings to specific machines are in the paper

- Arithmetic Promotion Rules – These promotion rules are the same as the C promotion rules. The words "size of integer" are replaced with "actual precision of integer".

(1) Integer specification is determined for both operands. Determine:
- actual precision
- specified precision
- exactness of precision
- performance attributes
- signedness

(2) The actual precision of the result is at least the maximum of the actual precisions of the operands.
(3) If both operands are exact, the result is exact. Otherwise the result is atleast.
(4) The specified precision of the result must be at least the specified precision of the operands.
(5) If both operands are signed, the result is signed. If both operands are unsigned, the result is unsigned. Otherwise if the operands differ in actual precision, the result is the signedness of the operand with the greater actual precision. Otherwise if the operands differ in specified precision, the result is the signedness of the operand with greater specified precision. Otherwise, the result is unsigned.
(6) If the operands have different actual precision, the result has the performance attributes of the operand with greater actual precision.

Otherwise, if either operand is optimized for time, the result is optimized for time. Otherwise, if both operands are optimized for space, the result is optimized for space. Otherwise, the result is unoptimized.

- Supporting Services
  - including **<stdint.h>** activates SBEIR
  - precof(x) – returns prec_t value that is the precision of expression or type
  - EIR_* Macros – exact information from an object of type prec_t.
    EIR_BIT(prec_t t) => actual precision
    EIR_SBIT(prec_t t) => specified precision
    ...

- Example of Multiplication

```
typedef ... fact1_t;
typedef ... fact2_t;

#define fact1_len EIR_BIT(precof(fast1_t))
#define fact2_len EIR_BIT(precof(fact2_t))
#define prod_len(fact1_len + fact2_len)

signed int atleast:prod_len
my_prod(fact1_t f1, fact2_t f2)
{
   signed int atleast:prod_len p;
   p = f1;
   p *= f2;
   return p;
}
```

- The **fprintf** Function
  - Add "?" character prior to conversion specifier
  - The argument prior to the value must be "precof" applied to the value.

```
int atleast:32 x;

printf("%?d\n",precof(x),x);
```

- The **fscanf** function is similar
- The **strtoint** function

```
#include <stdint.h>
void *strtoint(const char *nptr, char **endptr,
               int base, void *buf, prec_t type);
```

  - If **buf** is not a null pointer, the result is stored in the object it points to, otherwise, **malloc** is used to allocate space for the object.

- Prior Art
  - Several public domain multi-precision libraries.
  - PL/1
  - IEEE Standard 1596.5

Big Aligned Unsigned Quadlet
- Big Endian
- Align on "quadlet" (4 bytes)
- Unsigned
- Quadlet (4 bytes, exactly)

Unsigned Quadlet
- Byte ordering/alignment not specified
- Quadlet (4 bytes, exactly)

Unsigned Quadmin
- Byte ordering/alignment not specified
- Quadmin (4 bytes minimum)

Farance outlined a bridge proposal that would allow the **<inttypes.h>** proposal to coexist with his proposal.

- Implementing Kwan's types as SBEIR types:

```
typedef signed int exact:8 int8_t;
typedef signed int exact:16 int16_t;
typedef signed int exact:32 int32_t;
typedef signed int exact:64 int64_t;

typedef unsigned int exact:8 uint8_t;
typedef unsigned int exact:16 uint16_t;
typedef unsigned int exact:32 uint32_t;
typedef unsigned int exact:64 uint64_t;

typedef signed int exact:EIR_BIT(precof(void *))
  intptr_t;
typedef unsigned int exact:EIR_BIT(precof(void *))
  uintptr_t;
```

Gwyn stated that the bridge proposal would be necessary if both the Kwan and Farance proposals are in the TR.

Several possible flaws with the proposal were discussed: promotion rules, bit-field types, punning. Plum asked for clarification on addressability: are all types in this scheme addressable? Yes. If extended to bit fields? No, still can't take **&** of bit field. Plum also expressed concern on how this proposal would interact with overloaded functions, if we decide to allow them.

MacDonald said CRI had decided against this proposal as it costs too much to implement. It also requires bit fields of up to 128 bits. Although "at least" types are useful many times, the "exact" types can be achieved through use of

a struct with a bit field. Stanberry asked for clarification of costs–increases complexity of compiler itself. MacDonald added that some features can't be implemented on some architectures, and adding this much to the compiler introduces possible errors.

Gwyn argued in favor of the proposal as it in general encourages development in architectures and programming languages.

Ream asked about the PL/1 experience. Tydeman reported that PL/1 has only exact, not at least, types, and that programmers end up using the size of the architecture.

Mooney stated that he would approve this for the TR, but it needs further study to determine what violence this does to the C type system. Meyers asked if we could specify a less rich "exact" set; couldn't approve for the TR today. Kwan also expressed reservations about the future of this in C.

Plum reminded us that the TR is for experimental purposes. Thomas was skeptical, though, because this is very large, very late to be adding to the TR; the other proposals have been held to fairly high standards. Jaeschke agreed with that this proposal is quite different from the others. Gwyn said some leniency is in order because we promised to give it fair consideration at the San Jose meeting; also, this proposal solves long term problems which the Kwan proposal doesn't address. Stanberry agreed and believes it should be considered for the revision but shouldn't have competing proposals in the TR. Plum suggested that what we need is to find out what the market wants.

MSN Move we approve an edited N386 for the TR. (Farance, Kwan)
7/6/0/7/20

SV In favor of considering an edited N386 for the next revision.
12 Yes. 3 No. 4 Don't know/care.

*** Farance will draft a proposal for including an edited N386 in the next revision.

SV In favor of considering an edited N381 for the next revision.
16 Yes. 2 No. 1 Don't know/care.

*** Kwan will draft a proposal for including an edited N381 in the next revision.

<u>Technical Report Wrap Up</u>

Jaeschke reviewed the rules for TR's: basically, there are no rules. He therefore determined the following logistics: subgroup editors will forward all their edited documents to Jaeschke by April 1st.

*** Jaeschke will draft a preface for the TR.

The preface will include the history of NCEG and a table of contents.

Specifics for layout of the TR:

- Editors should provide all references using ISO Standard section numbering. Should refer to any ANSI numbers as "X3.159" rather than "ANSI". Kwan asked how to determine ISO numbering since he only has the ANSI numbering. The SGML version of the Standard has the ISO numbering, N385.

- Credits, acknowledgments can be included. The technical editor's name and email address should be clearly shown.

- Preface will identify which items are being considered for the C9X revision which is currently underway. Meyers suggested that appropriate disclaimer wording be used to clarify that even though submitted, those items will not necessarily be included in the revision.

- Add an appendix or introduction that talks about prior art, real or pseudo-implementations. Should verify with implementors that it is OK to be named.

- Chapter numbers were assigned to each subgroup:
  1 – Restricted Pointers
  2 – Variable Length Arrays
  3 – Designated Initializers
  4 – Compound Literals
  5 – Floating Point C Extensions
  6 – Complex
  7 – Data Parallel C Extensions
  8 – Extended Integers

- Editors should include chapter identification with page number–e.g., "chapter–page#".

Thomas asked if he should include prior FPCE comments. Jaeschke said if they were no longer relevant (i.e., addressed), just delete them; otherwise, keep in appendix.

There was discussion of whether copies of the TR should be made available to committee members. Plum suggested it could be placed on Keaton's ftp site to make available. There may be problems with this (copyright issues) after it becomes owned by ANSI, but would like the TR widely distributed before that happens.

Gwyn asked when the TR would actually be available from ANSI. Jaeschke was not certain, but expected it would be by the end of 1995.

Jaeschke gave some further instructions to the editors regarding page numbering, headers, and cover pages:

- page numbers should be at center bottom of each page
- chapter title should be at center top of each page
- cover page title should be "Chapter *n* <document title>"
- cover page should include editor + email address
- cover page should be page 1 (for table of contents) and will be a "right" hand page .

8. Other Business

Jaeschke thanked Plum for years of service as Vice Chairman of X3J11. He also thanked Stanberry for serving as reluctant secretary.

Plauger thanked MacDonald and CRI for handling the mailings for WG14.

Jaeschke suggested that we determine a voting requirement for approving proposals for the revision. He recommended a 2/3 vote, at least for the US committee, to make a change to the base document, as guidance for how the IR should vote. If it is harder to vote a change in, this will avoid fights to remove a feature. Plum said ISO already instructs the convenor not to accept changes with a bare majority vote.

Jaeschke said he would be happy with this as an informal understanding. Farance questioned utility of straw votes; if we really want a proposal in the base document, should be given a formal vote. Jaeschke clarified that a proposal becomes a change to the base document only when officially sanctioned by an ISO vote.

Gwyn argued that he doesn't want a sharp definition of consensus, but we need guidelines to deal with control of agenda time when proposals are not reaching consensus; we should give clear precedence to those reaching consensus. Plum said it is the responsibility of the IR to work to achieve consensus when a vote is between 1/2 and 2/3.

Jaeschke said we will use a 2/3 vote as guiding principle for now. Gwyn asked if the chair decides when to cut off debate? Jaeschke said this was guidance to the head of the US delegation only. Meyers suggested that we rephrase as "substantial majority" rather than exactly 2/3; also, should establish an ending date for accepting new proposals.

Farance cautioned that members should not vote "No" on a late proposal on which they earlier abstained.

Gwyn said the ending date should be the CD registration date, but we should not impose any other limits, even up to the last date. However, we should also make it clear that late proposals will have less chance of approval.

Plauger said it is more important to get the sentiment right than to develop formal rules.

Martin suggested an alternative would be to require confirming votes at two successive meetings.

Jaeschke said we may revisit this issue at the next meeting.

US TAG Meeting

Benito said we need to appoint delegation to the Copenhagen meeting. The current US delegation is Benito, Farance, Jaeschke, Plum, and Walls.

Jaeschke indicated he would give up his position on the delegation to allow Keaton to be on the delegation.

Kwan stated that he needed to know the exact business to determine if he can attend. Plauger indicated that they would begin formal work on revision now that the preliminaries are done.

MSP Move we approve Benito, Farance, Keaton, Plum, and Walls as our delegation to the next two meetings. (Jones, MacDonald)
18/0/0/2/20

MSP Move we approve Benito as head of the US delegation. (MacDonald, Jones)
18/0/0/2/20

The next item of business was approval of the Project Proposal [N394/94-079]. Jaeschke reviewed the document, and the following suggested changes were incorporated:

- (2.3) Reword last sentence to read: "Considerable prior art exists for C++ and the numerical extensions."
- (3.7) Correct the meeting location for the second meeting (Nashua, NH) and the meeting date (February) for the third meeting.
- (4.1) Strike the last sentence.
- (5.2, 5.5) Revised to get right words about types of projects and other committees.
- (5.7) Strike X3T2. "None."
- (5.8) Just X3J16.

There was discussion of whether X3T2 should be added under 5.8. We decided not to include unless told to do so by OMC.

Jaeschke will recirculate the proposal with these corrections.

MSP Move we approve the Project Proposal as amended. (MacDonald, Benito) 18/0/0/2/20

9. Future Meetings

9.1 Future Meeting Schedule

The schedule for co-located meetings of WG14 and X3J11 is:

6/12-16/95	Copenhagen	DS
10/16-20/95	Nashua, NH	DEC
2/5-9/96	Orange County, CA	Unisys
6/24-28/96	University of Amsterdam	NNI
10/21-25/96	Toronto, Canada	IBM

We also reserved the following dates: 2/3-7/97, 6/23-27/97, and 10/20-24/97.

Plauger spoke about scheduling considerations: by 10/96, we need to vote on CD registration. He wants us to maintain a believable schedule with the understanding that we slip four months every time we miss a deadline.

Plum added that when we reach the DIS date, no changes are allowed (except for clear typos); otherwise we have to start all over again. He strongly suggested that the meeting for the DIS be strictly an edit session. Therefore, no extensive changes would be allowed at that meeting, and that implies that the previous meeting would be the last time to make such changes. With respect to the public review, we should hope that this is on the CD, not the DIS. It is just not possible to do edits/edit review/60-day public review in a four month cycle. (J16 may shorten their public review period to 45 or even 30 days.) Therefore the last CD meeting may also need to be a 5 day edit session if we want a full 60 day comment period.

9.2 Future Agenda Items

This will be business as usual, but we need more detailed agenda for the second mailing. Plauger requested that Jaeschke provide a more accurate skeleton to him by the first mailing.

*** Jaeschke will work with Plauger to expand the agenda structure.

9.3 Future Mailings

All document numbers must be obtained from Plauger:

P. J. Plauger
398 Main Street
Concord, MA 01742

pjp@plauger.com
phone: 508-369-8489
fax: 508-371-9014

Items for mailings should have the document number on the cover of the document, and then should be sent to Plauger. He prefers hard copy, then fax, then email.

Deadlines for mailings are:

Post-Plano	Friday, COB	1/6/95
Pre-Copenhagen	Friday, COB	4/28/95
Post-Copenhagen	Saturday	7/8/95
Pre-Nashua	Saturday	8/26/95

*** Jaeschke will talk to CBEMA (ITI) about the timeliness of mailings.

10. Resolutions

10.1 Review of Decisions Reached

Stanberry reviewed the votes taken at the meeting, which are marked in the left margin of these minutes (SV, MSP, MSN).

10.2 Formal Vote on Resolutions

Plauger said we need just one formal vote by WG14 to approve all the results of the straw polls.

MSP Move we endorse the straw votes and resolutions from this meeting. (Benito, Martin)
2/0/0/??/??

Secretary's note: I don't know how many eligible voters there are for WG14. I recorded this vote as 2/0/0/0/2, but I think that is incorrect since I believe there are >2 eligible voters, even though there were only 2 attending.

Plum noted that the decision to forward TC had already been approved at the Tokyo meeting, and decisions at this meeting are subject to reconsideration at future meetings (i.e., they don't bind other national delegations).

10.3 Review of Actions Items

Stanberry reviewed the action items from the meeting, which are marked in the left margins of these minutes (***).

10.4 Thanks to Host

Jaeschke thanked Meyer and Convex for hosting the meeting!

11. Adjournment

The meeting was adjourned at 12:07 PM, CST, 9 December.

SECRETARY COPY

Attendance Roster - X3J11

Voting members	NAME	Mon	Tue	Weds	Thu	Fri
1 Convex	Randy Meyer	✓	✓	✓	✓	✓
2 Cray Research	Tom MacDonald	✓	✓	✓	✓	✓
3 DEC Professional	Rex Jaeschke	✓	✓	✓	✓	✓
4 Digital Equipment	Randy Meyers	✓	✓	✓	✓	✓
5 Farance Inc	Frank Farance	✓	✓	✓	✓	✓
6 Hewlett-Packard	John Kwan	✓	✓	✓	✓	✓
7 IBM Corp	Dove Mooney	✓	✓	✓	✓	✓
8 Keaton	David Keaton	✓	✓	✓	✓	✓
9 Lawrence Livermore	Linda Stanberry	✓	✓	✓	✓	✓
10 OSF						
11 Perennial	John Benito	✓	✓	✓	✓	✓
12 Plum Hall	Tom Plum	✓	✓	✓	✓	✓
13 RG Consulting	Ron Guilmette	✓	✓	✓	✓	✓
14 SDRC	Larry Jones	✓	✓	✓	✓	✓
15 Sun	Douglas Walls	✓	✓	✓	✓	✓
16 Taligent	Jim Thomas	✓	✓	✓	✓	✓
17 Thinking Machines						
18 Tydeman	Fred Tydeman	✓	✓	✓	✓	✓
19 Unisys	Johnathan Ziebell	✓	✓	✓	✓	✓
20 US Army	Douglas A. Gwyn	✓	✓	✓	✓	✓
Watcom	Fred Crigger	✓	✓	✓		
Tuple	Edward Ream	✓	✓	✓	✓	✓
Motorola	T. Van Sickle	✓	✓	✓	✓	✓

242

P.J. PLAUGER NV
398 MAIN ST.
CONCORD MA 01742
+1-508-369-8489
PJP@PLAUGER.COM

REX JAESCHKE V.
DEC PROFESSIONAL
2051 SWANS NECK WAY
RESTON, VA 22091 USA
+1 (703) 860-0091
rex@aussie.com.

LINDA STANBERRY V
LLNL
+1-510-422-9006
lstanberry@llnl.gov

Douglas Walls V
SunSoft
2550 Garcia Av, MS UMTV12-33
Mountain View, CA 94043-1100
+1 (415) 336-3454
douglas.walls@eng.sun.com

FRED TYDEMAN V
3711 DEL ROBLES DR
AUSTIN, TX 78727-1814
+1 512 255-8696
tydeman@netcom.com

FRED CRIGGER NV
WATCOM
415 PHILLIP STREET
WATERLOO, ONT N2L3X
(519) 883-6307
Fred@watcom.on.ca

Jonathan Ziebell V
25725 Jeronimo Rd
Mission Viejo, CA 92691
+1 714 3806504
ziebell%mp068@eccsa.tredydev.unisys.com
Unisys Corp.

John Benito V
PERENNIAL
4003 Mission St
SANTA CRUZ, CA
(408) 457-3915
jb@peren.com

Frank Farance ✓
Farance Inc.
555 Main Street
New York, NY
10044-0150 USA
Phone: +1 212 486 4700
Fax: +1 212 759 1605
E-mail: frank@farance.com

Douglas A. Gwyn (U.S. Army) ✓
801-L Cashew Ct.
Bel Air, MD 21014
+1 410 838 2765
gwyn@arl.mil

NEIL MARTIN (WG14) UK HOD
MAKE IT SO

Randall P. Meyer ✓
Convex Computer Corporation
3000 Waterview Pkwy
Richardson, TX
+1-214-497-4224
meyer@convex.com

Dave Mooney V
~~844 Don~~
IBM Toronto Lab
844 Don Mills Rd
North York ON M3C 1V7
CANADA
+1 416 448 4474
dmm @ vnet. ibm. com

Edward Ream, Tupple, ~~Inc~~ NV
1617 Monroe St, Madison WI 53711
edream @ aol. com

Ted Van Sickle NV
~~Motorola~~ SPS.
12254 Hancock
Carmel, In 46032
Phone 2175717011 - Fax 317 575 8278 email RSFV40@WAECVM.NOT.CO

LARRY G. SULLIVAN (OBSERVER) C NV
858 WESTLINE ST.
EL PASO TX 79904
915/757-3243

Randy Meyers, Digital Equipment V
110 Spitbrook Rd
ZKO 2-3/N30
Nashua, NH
603-881-2743

215

John Kwan ✓
Hewlett-Packard Company
19447 Pruneridge Ave
Cupertino, Calif.
(408) 447-6442
jkwan@~~f~~cup.hp.com

Ron Guilmette ✓
RG Consulting
396 Ano Nuevo Ave. #216
Sunnyvale CA 94086
(408) 732-7839
<rfg@segfault.us.com>

MacDonald, Tom	Cray.	✓
Jones, Larry	SDRC	✓
Plum, Tom	Plumhall	✓
Keaton, David	Keaton	✓
Thames, Jim	Taligent	✓

David Keaton (independent) voting
1630 30th St., #311
Boulder, CO 80301
(303) 589-9DMK
dmk@dmk.com