ANSI C++ Committee Meeting, March 12-16, 1990

Brück, Dag M.

1990

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7452)/1–10/(1990)

ANSI C++ Committee Meeting
March 12–16, 1990

Dag M. Brück

Department of Automatic Control
Lund Institute of Technology
April 1990

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
INTERNAL REPORT

Date of issue
April 1990

Document Number
CODEN: LUTFD2/(TFRT-7452)/1–10/(1990)

Author(s)
Dag M. Brück

Supervisor

Sponsoring organisation
ABB Automation AB
Ericsson

Title and subtitle
ANSI C++ Committee Meeting — March 12–16, 1990

Abstract

The first technical meeting of the ANSI C++ Committee (X3J16) was held in New Jersey. The main topics of this travel report are exception handling and the committee working groups.

A more detailed record of the meeting can be found in the minutes.

Key words
C++, Standardization, ANSI, X3J16

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
10

Recipient's notes

Security classification

1. Exception handling

The most interesting part of the March 1990 X3J16 meeting was the technical seminar on exception handling. Three proposals were presented. The proposal by Koenig and Stroustrup [1990] is a major revision of an earlier paper [1989b]. This exception handling scheme is similar to the schemes in Ada and ML. The main features are:

- User-defined exceptions and handlers.
- The block raising the exception is terminated. Execution cannot resume at the point of raising the exception.
- Handlers specify a parameter list, just like functions.
- On raising an exception, the handler is located by searching the chain of function calls backwards.
- Exceptions are objects and can be passed as parameters ("If anything goes wrong, please raise this exception.").
- Grouping of exceptions: `OVERFLOW` and `DIVIDE_BY_ZERO` can both be handled by a handler for `MATHERR`.

The mechanism for finding the "correct" handler is quite similar to the matching of overloaded functions. In particular, it is easy to catch a group of exceptions because implicit conversions from derived class to base class apply. Here is an example:

```
class MATHERR { ... };
class OVERFLOW : public MATHERR { ... };
class DIVIDE_BY_ZERO : public MATHERR { ... };

int f()
{
   try {
      return 1/g();
   }
   catch (const char* s) {
      error("g() goofed:");
      error(s);
      return 22;
   }
   catch (MATHERR) {
      // Handles any mathematical error
      return 0;
   }
}
```

The paper describes the handling mechanism in detail, and suggests two implementations — one portable (implementable in C) and one without run-time overhead. Particularly nice is the use of local variables for resource allocation/deallocation in real-time systems (sections 6 and 10). I have used this mechanism in a small real-time kernel, and the concept appears to work nicely.

Michael Tiemann (father of GNU C++) has developed a slightly different exception handling scheme [Tiemann, 1990]. These two schemes can hopefully be merged some time in the future, but there are still important differences:

- The function signature (the "head" of the function) defines what exceptions the function can raise.

  The advantage is obvious: by looking at the declaration every raisable exception is known, and the compiler can verify that exceptions are consistently propagated.

  I can see three basic problems with this scheme:
  - Functions must either explicitly propagate or explicitly handle a potentially large number of exceptions. This is awkward since (in my experience) recovery is only meaningful at some levels in the call hierarchy. I think there is a risk of using too many "catch-all" handlers just to reduce the amount of handler code to write.
  - If the list of exceptions is part of the function signature, can a virtual function defined in a derived class raise an exception not listed in the base class? If not, the derived class is severely restricted by the base class.
  - Should the list of exceptions be used for resolution of overloaded functions?

- No propagation of exceptions. Unless handled locally, any exception is transformed into a "failed-exception" exception.

  Programming style and language constructs influence each other. Tiemann's exception handling scheme was developed under the assumption that exceptions are normally handled locally and only propagated in exceptional circumstances (pun intended!). In this scenario it is reasonable to turn unhandled exceptions into a "failed-exception" exception, and also to explicitly list a small number of exceptions in the signature.

  My experience from using exception handling in a major C project is different. In this application, a small database manager, exceptions were routinely propagated several levels until handled by an interface layer. Several internal exceptions were then transformed into a few externally visible exceptions, after suitable clean-up.

- There is a special data type for exceptions. The user defines exception objects with different contents, but cannot extend exceptions by using inheritance.

Tiemann's scheme has been used since September 1989 at Sun Microsystems.

William M. Miller (Glockenspiel) has a fundamentally different approach based on resumption which I think is basically flawed. In this scheme, execution can resume at the point where the exception was raised. I believe it is much more difficult both to understand and to implement, and I am very reluctant to accept it until I have seen *good* examples that demonstrate the need for resumption. The examples given are either things that I doubt can be handled locally anyway (device not ready, operator new cannot allocate more memory), or concurrency is disguise (response to asynchronous event).

2. Concurrency and real-time programming

I raised the issue of concurrency and real-time programming at the meeting and was met with great sympathy, although people were reluctant to spend much work on defining yet another new feature. Everybody agreed that the most important job is to make sure we do not standardize something that will

make real-time programming inherently difficult. This issue formally belongs to the working group on language extensions.

I have developed a small real-time library in C++ that I have distributed informally to some interested persons. I hope the "minimalistic" approach will influence future discussion of concurrency and real-time systems. See also [IEEE, 1989].

3. Committee working groups

Several working groups were formed at the meeting:

International concerns. The main purpose is to solicit international participation in X3J16 (at the two meetings so far, I have been the only non-North American representative). The group will maintain one mailing list of committee members that are interested in international issues, and another list of "international" people interested in the work of X3J16. It should be noted that observing members of X3J16 receive all documents but are not required to attend the meetings.

Editorial. The purpose is to establish guidelines for the style and structure of the C++ standard. The base documents of X3J16 are:

- The C++ Reference Manual [Stroustrup, 1990];
- The ANSI C standard [ANSI, 1989];
- The ISO C standard (when it becomes available).

Communications. Because there are only three meetings per year, a fair amount of work must be done between the meetings. An efficient way is through electronic mail, and this working group is responsible for setting up communication and distribution of information via e-mail. A European file archive will be installed at Lund Institute of Technology, Sweden, if needed.

Formal syntax specification. A formal syntax specification is in itself a worthwhile goal, but few people (if any) on the committee seem to possess any competence in this area. On the ISO C committee, this issue was heavily stressed by the British delegation.

Environment. Includes the C/C++ pre-processor, the operating system, implementation dependent behaviour, and what not!

Libraries The goal is to standardize the interface of some commonly used libraries. Likely candidates are current C++ libraries (e.g., stream i/o) and the ANSI/ISO C library.

Language extensions. This group will consider major language extensions, as characterized by:

- a change to the type system;
- a change to the naming system;
- a change to the requirements of the run-time system;
- a change that enables a new style of programming.

The main proposals are parameterized types, exception handling, and perhaps to some extent concurrency. Other proposals are likely to surface. This working group is chaired by Bjarne Stroustrup.

Core language. The core language is the C++ language as defined in the base documents, but excluding libraries, pre-processing and the compilation environment. Chaired by Andrew Koenig.

Compatibility. The main issue is conflicts between C++ and ANSI C. Chaired by William M. Miller. See also [Koenig and Stroustrup, 1989a].

4. International affairs

A so called New Work Item was written for the forthcoming ISO C++ committee, which will be formed late 1990; the organizational meeting is expected to occur in early 1991. Steve Carter, Bellcore (USA) was nominated convener of the ISO C++ committee.

AT&T Unix Software Operation Europe, London, UK, will organize a technical seminar on C++ in June 1990. Bjarne Stroustrup will give the keynote speech, and Steve Carter will probably give a talk on C++ standardization.

5. Meetings and officers

The tentative meeting schedule for the rest of 1990 and 1991 looks like this:

Date	Place	Host
July 9–13	Seattle, WA	Microsoft
November 12–16	Cupertino, CA	Hewlett-Packard
March 11–15, 1991	—	—
June 17–21, 1991	Lund, Sweden	Automatic Control

The list of officers of X3J16 has been completed:

Officer	Name	Affiliation
Chairman	Dmitry Lenkov	Hewlett-Packard
Vice chairman	William M. Miller	Glockenspiel
Project editor	Peggy Quinn	AT&T
Secretary	Daniel Saks	Saks and Assoc.
Vocabulary rep.	Tom Plum	Plum Hall, Inc.
International rep.	Samuel Druker	Zortech, Inc.

It should be noted that approximately 50 people attended this meeting. Members of the ANSI C committee (X3J11) have noted that there are more end-users (as compared to implementors) in X3J16. There are also quite a few independent consultants.

6. References

ANSI (1989): "Programming Language C," American National Standard X3.159-1989.

Brück, Dag M. (1990): "ANSI C++ Committee Meeting — December 15, 1989," TFRT-7421, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

IEEE (1989): "Realtime Extensions for Portable Operating Systems," Draft, P1003.4/D9, December 1, 1989.

KOENIG, ANDREW and BJARNE STROUSTRUP (1989a): "C++: As Close as Possible to C — But No Closer," *The C++ Report*, **1**, 7, July/August 1989.

KOENIG, ANDREW and BJARNE STROUSTRUP (1989b): "Exception Handling for C++," *Proceedings of C++ at Work*, October 1989.

KOENIG, ANDREW and BJARNE STROUSTRUP (1990): "Exception Handling for C++ (revised)," *Proceeding of USENIX C++ Conference*, San Francisco, April 9–11, 1990.

STROUSTRUP, BJARNE (1990): "AT&T C++ Language System, Release 2.1 — Product Reference Manual," Committee document X3J16/90-0020.

TIEMANN, MICHAEL D. (1990): "An Exception Handling Implementation for C++," *Proceeding of USENIX C++ Conference*, San Francisco, April 9–11, 1990.

7. Appendix

The following two handouts at the meeting are enclosed:

- Goals of X3J16, document number X3J16/90-0023.
- Plans for X3J16 working groups, document number X3J16/90-0026.

Proposed Goals (version 3)

1. The standard will be based on the C++ Reference Manual (X3J16/90-0020) and the C Programming Language Standard (ANS X3.159-1989).

   The ISO C Standard will become an additional base document when available.

   The C++ Reference Manual will take precedence when the base documents disagree except where the committee decides otherwise.

2. Standardize the syntax and semantics of C++ programs expressed as a sequence of tokens without preprocessing directives.

   a. Select vocabulary involved in semantic and syntactic specifications.

   b. Resolve syntactic and semantic ambiguities and conflicts between the base documents.

   c. Provide formal specification of the syntax for C++.

3. Define and standardize a minimum set of C++ libraries, their contents, and interfaces.

   To provide a smooth migration path, the C++ libraries will be consistent with the ANSI C library.

   The proposed standard library will contain an I/O stream library as a minimum.

4. Standardize elements of a C++ environment. Issues include

   * preprocessing and lexical analysis
   * linkage
   * defining what it means to say that a given C++ implementation is compatible with a given C implementation
   * interaction with other languages
   * program startup and termination
   * differences between target and host environments
   * differences between freestanding and hosted implementations

5. Consider proposed major extensions to the base documents including parameterized types and exception handling.

6. Ensure that the X3J16 standard is suitable for the international community.

   X3J16 believes in producing a single standard for C++ which is acceptable both nationally and internationally. X3J16 will address all international issues as they are raised and will actively work to solicit members and commentary from the international community.

7. Ensure a very high level of compatibility with ANSI C.

   Complete compatibility is not a goal. Existing levels of

compatibility should be maintained. New incompatibilities with ANSI C should be rejected unless deemed necessary. Each incompatibility will be documented and justified.

8. Establish coordinating liaisons with X3J11 (ANSI C) and NCEG (Numerical C Extensions Group).

   These coordinating liaisons will provide directions for all three technical committees and help maintain inter-compatibility.

9. X3J16 will produce two deliverables: a draft proposed standard and a rationale.

   The rationale will consist of justifications for decisions made by X3J16.

   The standard will be written in English except for the formal grammar specification. This does not preclude work on a formal specification as an optional part of the standard.

10. Priorities:

    1) clear and unambiguous specification
    2) compatibility with the C++ Reference Manual
    3) compatibility with the other base documents
    4) consistency
    5) favorable user and implementor experience
    6) portability, efficiency, expressiveness
    7) ease of implementation including translatability into C

International Concerns Working Group - Steve Carter

Goals:

1. Shepherd the new work item
2. Solicit international participation

Scope: Review and comment on ISO documents referred to the US member body for C++.

Deliverables:

1. SC22 new work item
2. International announcements of C++ work
3. Two international contact lists:
   — X3J16 members with international interest
   — International community members interested in X3J16

Editorial Working Group - Peggy Quinn

Goals:

1. Establish
   — the high-level structure of the standard
   — the fine-grain style of presentation in the standard
   — the structure of the rationale
2. Disband at the next meeting

Deliverables:

1. High-level structure of the standard and rationale
2. Guidelines describing the presentation style

Communications Working Group - Reg Charney

Goal: Facilitate electronic communication between *all* committee and WG members.

Method: Internet mail (including gateways, volunteers, commercial services)

Input:

1. Address list from officers
2. WG membership lists from officers
3. Single communication of change of address or membership status through above lists

Deliverables and Schedule:

1. WG and whole committee mail reflectors (Andrew).
   — approximately two weeks after lists get to Andrew
2. On request, information on using mail facilities
   — ongoing
3. On request, help finding mail facilities
   — ongoing (initial suggestion list 1 Apr 90)
4. European ftp archive (Dag)
   — ongoing
5. Send distribution list to everyone with everyone on it
   — 1 May 90
6. Address/membership request address
   — same as (1)

Formal Syntax Specification Working Group - Jim Roskind

Goal: Provide a formal specification of the syntax for C++.

8

Scope: See below.

Deliverable: Formal specification.

Game Plan:

1. Establish evaluation criteria for a specification
2. Survey possible forms for a specification
3. Solicit and review complete C++ syntax specifications
4. Finalize the specification

Subgoal: 1, 2, half of 3

Deliverable: Reports on 1, 2, half of 3

Environment Working Group - John Vasta

Goals: Produce an outline of the group's proposal for the contents of the standard which relate to the C++ environment. The outline will indicate the group's recommendations for the direction to proceed in each area.

Scope: The outline will cover the areas identified in item (4) of the committee's goals document, with the addition of program structure and C/C++ preprocessor compatibility.

Deliverables: The outline document and supporting rationale.

Libraries Working Group - Michael Vilot

Goals: Produce an outline of the group's proposal for the contents of the standard which relate to C++ libraries. The outline will indicate the group's recommendations for the framework and direction to proceed.

Scope of work:

1. It is not the task of the committee to write or design C++ libraries. Instead, we will describe a library interface and specifications based on existing libraries with implementation experience.
2. The committee will solicit class library specifications from C++ library implementors for inclusion in the draft working paper.
3. The committee will exclude from the standard libraries proposals which are not of widespread value to the entire community of users (e.g., String class vs. Tensor class).
4. Other standards groups may provide C++ bindings for their libraries. Such libraries are outside the scope of this committee.

Deliverables:

1. A document addressing the use of standard header statements from ANSI C
   ```
   #include <stdio.h>
   ```
   and standard functions from the ANSI C library
   ```
   printf("Hello, world!\n");
   ```
2. A document showing procedures for soliciting, considering, and determining the inclusion of C++ libraries in the draft working paper. This includes a proposed document for RFP.
3. A description of some criteria to apply to proposed C++ libraries.

Language Extensions Working Group - Bjarne Stroustrup

Goal: To consider proposals for major extensions while preserving the spirit of C++.

Short Term Deliverables:

For July - a report on a proposal on parameterized types to be submitted by Bjarne Stroustrup.

For later meetings - report(s) on exception handling proposal(s).

A major extension is:

1. a change to the type system
2. a change to the naming mechanism
3. a change to the requirements of the runtime system
4. a change that enables a new style of programming

Long Term Deliverables: We will classify, explain, and evaluate proposals and make recommendations to the committee. If a recommendation is the adoption of a new feature, a draft of changes to the working paper will be provided.

Core Language Working Group - Andrew Koenig

Goal: Prepare the ground for standardization of the syntax and semantics (including items (2a) and (2b) but not (2c) of the goals document).

Scope:

1. Identify the parts of the base documents that fit into the "core language."
2. Define a procedure for handling requests.
3. Create an initial list of requests.

Deliverables: As above.

C Compatibility Working Group - Mike Miller

Goals:

1. Identify as many conflicts as possible between ANSI C and C++
2. Categorize the conflicts according to type and severity
3. Make recommendations for each conflict, either for resolution or for consideration by the whole committee

Scope: X3.159, chapter 3 (Language) except for preprocessing.

Deliverables:

1. List of conflicts
2. List of suggested resolutions
3. List of abdications