Report from the Library working group
Work done 9-13 Jul 90

This report presents a summary of the work the Library working group did during the second meeting. It is organized along the lines of the slides from Thursday's presentation. The details are enclosed as a separate section.

Libraries

Tuesday's presentation of the Library group's discussions revealed some concerns from the members of the X3J16 committee. The major issues were the large scope (there was some feeling that the group would not be able to address so many items) and the lack of focus (after 3 months' discussion, there were no written proposals). A particular item of concern was the lack of progress on the hardest and (to many) the most important concern: compatibility with the C library.

The directions from the X3J16 committee were to formulate a prioritized list (similar to the approach taken by the Extensions working group) and a plan for completing work on the most important items before starting work on the rest. Dan Saks suggested that the group work out a statement of guiding principles for inclusion into the Rationale.

The Library group worked during the week to address these concerns. Thursday's presentation to the full committee presented the results of the group's work in the following areas:

Criteria
Priorities
Table of Contents
— string
— streams
List of Issues
Plan / Schedule

Criteria

Universal
Experience
Quality
Spirit of C++
Consistent with existing libraries
Economy
Portable
Not conflict with other standards

Tim O'Konski agreed to coordinate the group's ideas on criteria. This list reflects the results of that effort. A statement elaborating this list should eventually appear in the Rationale for the library.

Priorities

1. Language support (small)
2. ANSI C (small - medium)
3. Input / Output (large)
4. String (small)
====================================
etc.

This list reflects the approach taken by the Extensions group, and indicates the sentiment expressed by the committee after Tuesday's presentation. The main point is to focus on the most important items first.

The string class made it on to the list for two reasons. First, it is small, and can serve as the pilot effort to work out the details of how to present the contents of Chapter 18 of the Standard. Second, it is an example of how a standard class can be of some benefit — different existing libraries provide slightly different definitions of a string. Providing a single accepted definition would simplify the difficulties faced by users when combining libraries.

Table of contents

18. Library
18.1 Language Support Packages
18.2 ANSI C Library
18.3 Input/Output and Formatting Package
18.4 String Package

Michael McLay agreed to coordinate the input to the initial table of contents. One interesting aspect of the discussion was whether the group should draft the full, final table of contents immediately, or let it evolve as proposals are adopted. The list presented here favors the latter approach. This implies that if the sections of Chapter are listed in alphabetical order, the section numbering will change as new sections are added.

String Example

18.4 String Package
18.4.1 Header <string.hxx> ?
18.4.1.1 Class String
    Synopsis:
18.4.1.1.1 Behavior of String()
    Description:
18.4.1.1.2 Behavior of String(char pc)*
    Description:
18.4.2 Exceptions
    Global
    Within package

Aron Insinga volunteered to work up a draft of how one section of Chapter 18 would look. He developed an example presentation of the string class more to illustrate the format of the presentation than as the best design of the class.

Bjarne Stroustrup pointed out that some people might have difficulty with the term "package."

Streams

Retain:
```
    class ios;
    class istream;    cin
    class ostream;    cout, cerr

    class streambuf;
    class filebuf;
    class strstreambuf;

    template class IAPP, OAPP;
    template class IMANIP, OMANIP;
```

Jerry Schwarz took on the task of simplifying the AT&T 2.0 iostreams classes. His proposal keeps the basic design intact, and eliminates some of the awkward and less-used features. The presence of templates and exceptions will change some aspects of the iostreams design. The issues of access to (and interaction with) the stdio portion of the ANSI C library will have to be considered.

List of Issues

template:

Working paper
Proposal (for vote)
Ratification (in document)

Thursday's presentation concluded with a list of the items the Library working group expects to deliver at the next meeting. The basic idea is to have all proposals from the group go to the full X3J16 committee for consideration in three stages:

First, a working paper will present the proposed addition. The paper will indicate what existing library(ies) form the basis for considering the proposal as part of the Standard. It can also indicate what changes were made from the existing version(s).

The second stage is a specific proposal in manual form. This would include numbered sections and wording complete enough so that only editorial changes are needed to incorporate the text. Once the full X3J16 committee has voted to incorporate the proposal, the Editorial group will prepare a revised working draft of the Standard.

The last stage is a check that the final wording of the Standard correctly incorporates the proposal. A proposal might go through all three stages in the course of one meeting, but that is unlikely.

The Library working group expects to have nine working papers for the next meeting in November. Four of them relate to contents of the Standard, and should evolve into concrete proposals. The others address some issues that impact several such proposals.

Contents

C Compatibility
Streams
String
Multibyte Characters

Shawn Nash agreed to coordinate input regarding the difficult and important issue of compatibility with the C library. Steve Clamage could not attend the July meeting, but expects to be at the next meeting. Between them, and with the input of all interested parties, there should be a solid presentation in November.

Jerry Schwarz agreed to work out more details on the `streams` portion of the Library, and Mike Vilot agreed to write up the working paper. One issue to be addressed is a consistent approach to initializing objects in the library (such as `cin`, `cout`, and `cerr`).

Aron Insinga agreed to put together a proposal for the `string` portion of the Library.

Wen-Yau Hsieh agreed to write up a report on possible library support for multibyte characters. The report will present an analysis of the "Draft Proposed Multibyte Support Extension of ANSI C" currently proposed to ISO SC22/WG14.

Guidance/Procedure

Motivation / Philosophy / Principles
Conformance / Extensibility
Name Space
Formal Specification
Solicitation

Tim O'Konski agreed to continue refining the statement of principles for the library and produce a proposal for the introduction to Chapter 18 of the Rationale.

Pete Becker volunteered to write up a summary of the conformance and extensibility issues. This is an interesting topic, because it tests the ability of a Standard to provide common agreement — for a library that is designed to be extended!

Wen-Yau Hsieh also agreed to summarize the input on naming conventions and name space management. This is a particularly difficult aspect of combining libraries — including the C and C++ libraries. Consider that the class `string` should be found in a header file named `string.h`, but that the name is already preempted by the C library.

Aron Insinga volunteered to write up a proposal for using formal specification to document the semantics of the Library. His opinion is that library semantics are easier to describe than language semantics. Given the interest in formal specification repeatedly indicated by the International concerns group, the use of formalisms in describing the Library portion of the Standard might be appealing.

Aron Insinga also agreed to document the solicitation procedures the Library group is using. This includes the informal process currently at work, and research into to any more formal process that X3 may require. The more formal process may be necessary if the X3J16 committee decides that a formal canvass of existing C++ libraries is warranted.

The following section is a copy of the document assembled by Michael McLay during the meeting, from input provided by all members of the Library working group.

7-11-90 6pm DRAFT

1. MOTIVATION AND PHILOSOPHY — LIBRARY

A. Top level objectives

1. Make sure that the users of C++ have enough functionality that is guaranteed to be the same across vendor libraries to do the bulk of common programming tasks.
2. Ensure that the library functionality codified in the standard represents (close to) the state of practice.
   a) Note that this implies controlled change in the library standard over time.
      1) To track new language features
      2) To track new library technology

B. Challenges in meeting these objectives

1. Templates not in any existing library in public use
2. Exception handling not in any existing library in public use
3. Numerical exceptions not currently detailed; does this committee specify them? (Requires reconciliation with other standards group(s)).
4. Name space pollution: Collisions between class names from different vendors are already occurring.

C. Approach to be taken

1. Specify one class, strings, in detail, as a way of flushing technical difficulties in specification to the surface.
2. Specify the input and output stream package, which will be based on the existing iostream package [a brief description of a first pass at how this process will take place will be presented today].
3. Create a document to examine the issues of how the ANSI C library should be included as a part of the standard C++ library, for the purpose of forming a basis for a future proposal.
4. Specify
   (1) methods for acquiring candidate library documentation and specifications,
   (2) criteria for selecting what goes in the standard library, and
   (3) the mechanism for how it gets incorporated in the evolving standard.
      a) Primarily by scanning what is out there and standardizing existing practice.
      b) Except for new language features which we might add to existing practice.
5. Use templates and exceptions where appropriate in the C++ library specifications that this committee proposes.
6. Write a proposal to the whole committee on name space management for libraries (including taking a position on whether we preempt the intuitive names of provide vendors with a means for ensuring that their class names are unique).

2. Priority List of C++ Library Activities

1. Language Support Packages
   General discussions of classes like new and delete that are used to support the base language will be discussed here.

2. Initialization
   A description of the order of initialization of things like cin and cout will be discussed here.

3. ANSI C Library
   Three levels of interfacing to the C library were discussed. No conclusion was reached as to which of these levels should be done.

4. Input/output and Formatting Package
Specification of an I/O package.

5. String Package
Specification of a character string representation and manipulation package.

6. Container Packages
Lists, bags, sets, ...

7. Memory Manager Package
zones?

8. Garbage Collection Package

9. Locale Package
The environmental settings like time zone, currency, language... that specify the current state of a program.

10. Date Package
Implements a date class that works in conjunction with the locale package.

11. Time Package
Implements a time class that works in conjunction with the Date and Locale packages.

12. Currency Package
Implements a currency class that works in conjunction with the Locale package.

13. Task Package
a la Simula67

14. Task Package
Concurrent execution or parallel processing package?

15. Bit Package
Specification of a bit stream or bit map representation and manipulation package.

16. Math Package
Should include functionality of the C math library and the existing C++ complex class. Other miscellaneous classes include vectors, matrices, float, and infinite precision math.

3. TABLE OF CONTENTS FOR REFERENCE MANUAL

4. String PORTION OF REFERENCE MANUAL

18.4 String Package

18.4.1 Header <string.hxx>

18.4.1.1 Class String

Synopsis:

```
class String
{
public:
String();
String(char*);
~String();
void operator=(const String&);
String(const String&);
friend String operator+(const String&, const String&);
int length();
// (etc, etc, etc)
};
```

18.4.1.1.1 Behavior of String()

Description:

Constructor for a String. The initial value of the constructed String is the sequence of no characters.

Post Conditions:

(*this).length() == 0

(*this) == ""

18.4.1.1.2 Behavior of String(char* pc)

Description:

Constructor for a String. The initial value of the constructed String is the sequence of characters in the array pointed to by the parameter pc up to, but not including, the first character equal to '\0'.

(etc, etc, etc)

18.4.2 Exceptions

Global

Within package

5. PRELIMINARY SKETCH OF ITEMS IN Iostream LIBRARY

```
class ios {
  class Init;
}
class istream;
class ostream;

extern istream cin;
extern ostream cout;
extern ostream cerr;

class streambuf;
class filebuf;
class strstreambuf;

template<class T> class IAPP;
template<class T> class IMANIP;
template<class T> class OAPP;
template<class T> class OMANIP;
```

Items explicitly omitted:

```
/ ——  iostream;
o     extern ostream clog;

      stream_withassign;
      fstream;
      {i,o}fstream;
      {i,o}strstream;
      stdio{stream,buf}

      S{MANIP,APP}
      Input/Output{MANIP,APP}

      streambuf::setbuf
      streambuf::doallocate
      streambuf::unbuffered
      streambuf::allocate

      {i,o}stream::seek{g,p}
```

6. WHITE PAPER ON USING THE ANSI C LIBRARY IN C++

Create a document to examine the issues of how the ANSI C library should be included as part of the standard C++ library, for the purpose of forming a basis for a future proposal. Discussion to date indicates three levels of integration:

- lowest level
  Do nothing. Application writer can call C library as they see fit. (No safety belt.)
- ——> middle level
  Minimal work to cover minor holes in type checking. (Only fix what is easy)
- high level
  Create C++ classes that provide all ANSI C library functionality. (closes all type checking holes.)

7. CONFORMING LIBRARIES

We need to define what it means for the library packages in an implementation to be in conformance with this standard. At the moment, it looks like we're going to have to agree on some new terminology to be able to specify this accurately. The proposed rules, generally stated, are as follows:

1. The implementation must provide as a minimum the interface specified in the standard;
2. The implementation may provide additional non-member functions with the same name as non-member functions specified in the interface, provided that these functions have parameter lists which make them distinguishable from the required non-member functions.
3. The implementation may provide additional class members as desired.
4. The implementation may provide extensions to enumerated types, in the form of additional enumerators, as desired.

8. NAME SPACE MANAGEMENT

Problem:
external/internal name collision

Issue:

- existing names in ANSI C library (as is)
  - — case sensitivity and number of significant characters (Environment WG)
  - — header file names
  - — class names
  - — external names
- names in ANSI C++ class library

- names in vendors' libraries
- names in users' libraries/programs for multiple vendor platforms

Proposal:

1. ANSI C++ library defines a set of unique names used without the assumption of case sensitivity.
2. ANSI C++ library defines implementors' rules for internal names
3. Implementations should guarantee that all names generated by a class with a unique class name are unique.
4. No solution discussed to vendors' own class libraries and users' libraries (nested class names?)

9. FORMAL LIBRARY SPECIFICATION

The precision of the standard may be enhanced in some cases by the use of mathematical notation, axioms, assertions, trace specifications, structured English, tabulation, etc. over and above what could be achieved with prose alone. We invite everyone to submit illustrations of the application of their favorite techniques to C++ class documentation.

Formal notation and prose description could appear together in either the actual standard, an appendix, or the rationale.

10. CRITERIA FOR INCLUSION IN C++ LIBRARY

1. Universality of use in programming

2. Extensive existing practical experience

3. Quality of design

4. In keeping with the spirit of C++ (object oriented tone...)

5. Consistent with other existing library classes

6. Economy of expression

7. Platform independence

8. Not in conflict with other standards