C++ Compiler, Lib, Std Issues

- **Standard = near final draft form since December 1996**
  Downloadable form (txt, html, ps, ...): http://www.setech.com/x3.html
  Directly WWW navigable: http://www.cygnus.com/misc/wp/dec96pub

- **Books beginning to appear which are up-to-date**
  [Stroustrup3] is even more up-to-date than the posted draft. [Ladd], [Nelson]

- **Compilers beginning to appear which are up-to-date**
  Kai C++ (compiler and library) is very nearly completely conforming.
  SGI MipsPro 7.2 compiler is nearly conforming, but library is in poor shape.
  We are investigating using Rogue Wave Standard C++ Library in its place.
  SGI MipsPro 7.1 is usable. **GNU C++ 2.7.X** is becoming more and more stale.

- **Hidden major difference between compilers -** How to treat template classes to avoid redundant copies = Template Instantiation Problem.

Some Known Problems...

- **G++ 2.7.X strstream class:** Leaks memory (tested by Insure++), often returns invalid string (garbage characters at end) from str() method.
- **G++ 2.7.X pointer output:** operator <<() mistreats pointers as bool. To get correct output, one must cast pointers to be void *.
- **G++ 2.7.X static_cast<>():** Many compilers (G++) ***mistakenly*** treat static_cast<>() as being ***exactly*** the same as a C-style cast. Since there is no built-in or user-defined conversion between pointers to built-in types, correct =

```
int *p_int_rep = &_int_rep ;
char *p_chars = (int *) p_int_rep ;                  \\ Compiles. C-style casting
char *p_bytes = static_cast<int *>(p_int_rep) ;  \\ Standard: fails compilation
```

- **Stdlib namespaces:** Compliance varies. Nearly all stdlib classes should be in the "std" namespace, in extension-free headers. A few template methods to help define operator !=() in terms of operator ==() are in "relops" namespace.
- **Official FNAL source:** http://www-oss.fnal.gov/~mengel/c++wg

STL Example: vector

- **STL vector = "array" data structure:** Good for random access and for fixed size. Poor for insertion or deletion in middle or for varying size.
- **Random access via index: just as with built-in arrays**
- **Iterators provide simplified sequential access:** The special value methods begin() and end() return iterators pointing to the first element of vector and to one past the last element of vector, respectively.
- **Large suite of STL algorithms can be applied to a vector**
- **"All at once" assignment, comparison operators, and other specialized methods defined for vector:** For example, one can perform an "all-at-once" swap() on two vectors which takes advantage of knowledge of vector implementation to perform an optimally efficient swap.

Intermediate C++
18-26 August 1997
Rob Kennedy
Liz Sexton-Kennedy

STL Example: vector (2)

- **STL vector example:** Declare and use two vectors of doubles.

```
vector<double> dvec(4) ;         \\ 4 elements, init value = 0.0
vector<double> vec2(4, 2.0) ;    \\ 4 elements, init value = 2.0

dvec[0] = 0.0 ; dvec[1] = 1.0 ; dvec[2] = 2.0 ; dvec[3] = 3.0 ;
vec2 = dvec ;

if (dvec == vec2)
  { cout << "As expected, two vectors are now equal" << endl ;
  }

for (vector<double>::iterator iter = dvec.begin() ; iter != dvec.end() ; ++iter)
  { cout << " " << *iter ;
  }
cout << endl ;
```

STL Example: list

- **STL list = "doubly-linked linear list" data structure**
  Good for insertion or deletion in middle or for varying size. Not as good for random access or fixed and small size requirements.

- **No random access via index and operator[]**
- **Iterators provide only means of element access:**
  The special value methods begin() and end() return iterators pointing to the first element of list and to one past the last element of list, respectively.

- **Large suite of STL algorithms can be applied to a list**
- **"All at once" assignment, comparison operators, and other specialized methods defined for list:** For example, one can perform an "all-at-once" swap() on two lists which takes advantage of knowledge of list implementation to perform an optimally efficient swap.

Intermediate C++
18-26 August 1997
Rob Kennedy
Liz Sexton-Kennedy

STL Example: list (2)

- **STL list example:** Declare and use a list of strings.

```
list<string> arglist ;          \\ No need to specify initial size or values for elements

for (int index = 0 ; index < argc ; ++index)
  { arglist.push_back(argv[index]) ;        \\ append each command line arg to list
  }

sort(arglist.begin(), arglist.end()) ;  \\ By default, sort is done alphabetically here

arglist.push_front(string("SortedArglist=")) ;      \\ pre-pend new element to list

for (list<string>::iterator iter = arglist.begin() ; iter != arglist.end() ; ++iter)
  { cout << " " << *iter ;
  }
cout << endl ;
```

Other STL Containers

- **STL deque:** A double-ended queue. Good for insertion at begin or end, but not in middle. Low storage overhead relative to a list, but can be less efficient.
- **STL associate containers: set, map, multiset, multimap**
  Storage and retrieval of data based on a key. Set classes store only the key. Map classes store key and associated data. Non-"multi" variants permit only one occurrence of a key in container, "multi" variants allow multiple occurrences. In our work, use of multimap classes will be common (eg. CDF Run 2 Tracking).
- **STL container adapter examples: stack and queue**
  Some data structures adapt an existing STL container to have the correct interface for their abstraction. Stack and queue adapt deque to have the correct FIFO (first-in, first-out) or LIFO (last-in, first-out) characteristics.
- **STL extension example: SGI's hash table**
  A hash table is a container adapter using multimap as its implementation. This is available in SGI's version of the STL... could be ported to Standard C++ lib.

STL Notes

Intermediate C++
18-26 August 1997
Rob Kennedy
Liz Sexton-Kennedy

- **Many algorithms ready to re-use:** Entire catalog of algorithms exist for finding, sorting, counting, applying a method to each element, ...
- **STL "Framework" includes provisions for memory allocation optimization:** Allocator template arguments give user a hook to implement, for example, bulk allocation of many small list nodes, etc.
- **New containers can be added to framework:** If a container follows STL guidelines, then appropriate algorithms are ready for re-use. Examples listed on WWW: http://www.ipmce.su/~fbp/stl/components.html
- **Iterators are not robust to container modifications**
  A surprise to most STL users. If you modify a container in a loop driven by an iterator pointing into that container, then that iterator may become invalid. There is a work-around for one iterator in use at a time, but not for more.