



BABAR Database Workshop 8th-9th July 1997

Objectivity Collection Classes

David R. Quarrie

Lawrence Berkeley National Laboratory

DRQuarrie@LBL.Gov



Overview

- Objectivity Intrinsic Collection Classes
- Persistent Rogue Wave Tools.h++ collection classes
- Standard Template Library (STL) persistent classes
- RD45 approach
- Summary



Intrinsic Classes - ooVArray(T)

- Variable size
- Zero-based (indices from 0 to N-1)
- Can contain objects or references to objects
 - Either transient or persistent if contained
- Cannot create an ooVArray of ooVArray
- Not directly persistent-capable
 - Embed within other persistent-capable objects
- Complete ooVArray mapped into memory at once
 - Size limitation
 - Page-based ooVArray has been requested



Intrinsic Classes - ooMap

- Persistent hash table
- Elements are object name & object ID (OID)
 - Key is C++ string
 - ▶ *char* key*
 - Dynamic sizing
 - ▶ Adjustable parameters
 - Initial number of bins
 - Maximum average density of elements in a bin
 - Growth factor



Intrinsic Classes - ooItr

- Iteration over sets of objects
 - All objects of a particular class & subclass
 - ▶ In FDDB, Database, Container
 - Belong to a on-to-many or many-to-one relationship
 - ▶ Uses association link
 - Are one level lower in the storage hierarchy
 - ▶ *e.g.* over databases in FDDB
 - Have a scope name for an object
 - Are named in a given scope
 - ▶ *e.g.* named objects within a database



Intrinsic - Queries

- Queries

- Adjunct to iterators - selections on meeting some condition
- Conditions specified by a predicate
 - ▶ *char* predicate*
 - ▶ *Restricted to data members*
 - “(x > 1) && (y < 5)”



Persistent Tools.h++

- Persistent-capable version of Tools.h++ classes
- Based on Tools.h++ version 6
 - c.f. transient Tools.h++ version 7
- Support being dropped by Objectivity
 - Replaced by STL-based classes
- Not a good candidate for long-term use
- Some classes have been tested on some platforms
 - In particular RWODTRefVector<T>
 - ▶ Equivalent to RWTPtrVector<T>
 - Used in prototype for persistent-capable collection
 - ▶ Needs to be replaced at some point



STL-based collections

- Based on STL from ObjectSpace
 - Standards<ToolKit> 2.1
- Both Transient & persistent-capable
- Will be part of Objectivity V5.0
 - Initial ship date end of September 1997 (Solaris & NT)
 - Additional platforms added over next 3-4 months
 - Should be available for our supported platforms 1st Qtr 1998
 - ▶ HP support is a problem
- Too late for our use?



STL classes

- *d_vector*
 - Extensible vector
- *d_list*
 - Lists with constant time insertions & deletions at any position
- *d_set* & *d_multiset*
 - Set or bag (duplicate entries)
- *d_map* & *d_multimap*
 - Keyed storage (*d_multimap* allows duplicate keys)



STL classes (2)

- *stack*
 - Stack: first-in, last-out
- *queue*
 - Queue: first-in, first-out
- *priority_queue*
 - Queue with sorting
- In all cases, there is a persistent-capable version with name *d_pc_xxxx*
 - e.g. *d_pc_stack*



Iterators

- 4 sorts of iterators
 - `::iterator`
 - ▶ Traverses objects sequentially, returning mutable objects
 - `::const_iterator`
 - ▶ Traverses objects sequentially, but returns const objects
 - `::reverse_iterator`
 - ▶ Reverses the direction relative to `::iterator`
 - `::const_reverse_iterator`
 - ▶ Reverses the direction & returns const objects



Transient/Persistent interoperability example - .ddl file

```
#include <d_list.h>
#include <deque.h>
#include <vector.h>
#include <algorithm.h>

typedef d_list< int, d_allocator<int> > int_list;

// Template instantiation for d_list of ints
template class d_list_node< int >;

class P_int_list : public ooObj, public int_list {};
```



Transient/Persistent interoperability

example - .cc file

```
ooHandle(P_int_list) list_h;
deque< int, allocator<int> > t_deque;
[...]
// Assume list_h is a valid handle
list_h->push_back(1);
list_h->push_back(2);

t_deque.push_back(3);
t_deque.push_back(4);

// Construct a transient vector of ints; allocate enough storage to hold list & deque
vector< int, allocator<int> > result(list_h->size() + t_deque.size());

// Merge list & deque into vector
merge(list_h->begin(), list_h->end(), t_deque.begin(), t_deque.end(), result.begin());

// Output the contents of the transient vector via ostream iterator
ostream_iterator<int> iter1( cout, " " );
copy( result.begin(), result.end(), iter1 );
cout << endl;
```



Tools available to migrate from Objectivity/DB to Objectivity/STL

- Iterators
 - VArrays (ooVArray), map dictionary (ooMap), associations
- Some restrictions on referential integrity with STL collections



RD45 approach

- HepVector
 - Interface to ooVArray(T) with Tools.h++ style interface
- HepRefVector
 - Interface to ooVArray(ooRef(T)) with Tools.h++ style interface
- Name changes being contemplated
 - Currently HepVector clashes with CLHEP class name



Summary

- Painful situation for us
 - Falling between two stools
 - ▶ Persistent Tools.h++ going away & persistent STL not quite here
 - Spring 1998 for support of all our platforms
- We are part of the beta program for Objectivity V5.0 & STL
- Difficult decision what to use
 - Potentially significant impact on design