Subject: WHY? "Index:: and ArrayIndex::operator[]" returns const T&
Posted by kohait00 on Thu, 22 Apr 2010 10:08:21 GMT
View Forum Message <> Reply to Message

hi there,

the Index/ArrayIndex containers are pretty helpful. their current use compared to VectorMap/ArrayMap is as I understood as follows:

the VectorMap/ArrayMap offer the way to find an object's position inside the container via an object *independant* hash value (key).

the Index/ArrayIndex offer the way to find an object's position inside the container via an object *dependant*, own hash, computed typically over the "value" of an object. therefore objects used in Index/ArrayIndex Container either have to have an own GetHashValue() function, or the container needs to be supplied a HashFn class, which can compute hash values for the object (i.e) if one can not manipulate the class (not own sources..). Indexes are not meant to be used for LAARGE objects, since another objects "value" (for hashing) is used to retrieve the position of an object in Index (temp object creation topics)

thus Index variants are *NOT* replacements for Map container variants.

now a problem arises:

in the Map variants, one can change the key refered to an object.
the Index cant do it because objects provide the hash value themselves, thats why Index may not have a mutator operator[] (like the Maps) to return a changable object. >> i understand that Index/ArrayIndex are meant as immutable readonlys in that case.

But consider the case of Indexes, where we need the objects to be mutable, so we would need means to update the hash table. to still have it pointing to the correct object.

some additions would be to update the hash of an element in the Index/ArrayIndex, and ofcorse the mutator operator[]

Index.hpp:127

```
void    SetKey(int i)           { B::hashfn(key[i]); }

T&      operator[](int i)       { return key[i]; }
```

opinions??