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??

Subject: Re: WHY? "Index:: and ArrayIndex::operator[]" returns const T&

Posted by mirek on Fri, 23 Apr 2010 08:25:43 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 22 April 2010 06:08
in the Map variants, one can change the key refered to an object.

Note "Index::Set" method.

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

This is impossible (both for Index and *Map) - after setting the key, internal structure has to be updated based on the new key.

Note that for *Map, operator[] returns the *value*, not the key.

Subject: Re: WHY? "Index:: and ArrayIndex::operator[]" returns const T&
Posted by kohait00 on Mon, 26 Apr 2010 07:29:08 GMT
View Forum Message <> Reply to Message

yes, i understand your point, this is what i have written above. as for now, the Index is meant to be a container for *immutable* elements, if you want to modify the content of the container, you delete old elementand place a new element. but what if you want to update the container elements in place ? you'd need means to update the internal hash map..

BTW:

i think in any case, we'd need to more clearly outline what each container type is actually for..(in most cases at least), kind of a table... describing what they are used for in most day to day cases.

currently possible:

Vector - moveable elements random acces
Array - arbitrary elements random access
VectorMap - moveable elements hash access via a key
ArrayMap - arbitrary elements hash access via a key
Index - moveable elements hash access over element's value
ArrayIndex - arbitrary elements hash access over element's value

something in the sense of
http://www.cplusplus.com/reference/stl/
should be made for the upp containers, to make selection/comparison easy

Subject: Re: WHY? "Index:: and ArrayIndex::operator[]" returns const T&
Posted by mirek on Mon, 26 Apr 2010 08:12:06 GMT
View Forum Message <> Reply to Message

Like this?

http://www.ultimatepp.org/srcdoc$Core$NTL$en-us.html

Mirek

Subject: Re: WHY? "Index:: and ArrayIndex::operator[]" returns const T&
Posted by kohait00 on Mon, 26 Apr 2010 15:41:47 GMT
View Forum Message <> Reply to Message

yes, something like that thanks