Subject: Re: no true Iterator support in Upp???
Posted by mirek on Wed, 29 Sep 2010 07:07:53 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 28 September 2010 16:56recently i came along to look closer at Iterator implementations in containers and i have to admit it is simpler that i expected..

generally i like simple, no doubt, it's most times the best option anyway. but the Iterator question somehow bugs me.. 2 things especially..

* Upp Iterator exist in 2 flavors, Iterator and ConstIterator, depending which one is able to be acessed, depending of the source state...(a const object can only return a ConstIterator). the speraration needs to be there to be able to correctly have mutator methods even on ConstIterator, like advancing the ii index..

No way around that. BTW, STL is the same.

Quote:
* Iterator is *not* container independant, not even the template versions.., its not a 'sourceless' interface.. that you can store together with others from different container type but same element type i.e.

Sure, but that is not even the point of iterators.

Quote:
* Iterator interfaces are not coherent. the template version is not the same as the Vector version i.e. The template version is only used by the BiVector/BiArray containers, offering the rich interface..

The reason there is template version is implementation only.

Quote:
what about a bit more flexible iterator interface?
i'm digging, as soon as done, i might post some ideas, but as i know mirek, he'll be up and done in no time with a cool solution anyway.

To tell the truth, iterators are modelled according to STL.

Gives at least one nice advantage: you can use STL algorithms with U++ containers

OTOH, I do not really like iterators - I appears to me that in many cases, you have to pass a reference to container with each iterator, while all algorithms use pair of iterators as range.

Perhaps the correct solution would be to provide some form Range interface... But whatever, in production code, iterators are almost never used outside Core package, so why bother... (and, they have that nice STL compatibility too, not that I ever used it).