Subject: stl-compatibility
Posted by piotr5 on Fri, 12 Apr 2013 19:26:19 GMT
View Forum Message <> Reply to Message

I noticed you have some nice macros for enabling stl-compatibility in upp containers. unfortunately it doesn't always work the way it should. obviously size() you have not added the cv-qualifier const. are there any containers that return GetCount() that could alter the object, or containers that abuse the call for cleaning up?

another more serious problem I noticed when trying the std::includes() template algorithm:

In file included from
/usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/bits/stl_algobase.h:66:0,
from /usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/algorithm:62,
from /home/p/upp/uppsrc/Core/Core.h:212,
from /home/p/MyApps/s6/s6.cpp:1:
/usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/bits/stl_iterator_base_types.h: At global scope:
/usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/bits/stl_iterator_base_types.h: In

/usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/bits/stl_algo.h:3519:2: instantiated from

r2) [with _IIter1 = Upp::Array<Upp::String>::ConstIterator, _IIter2 =

/home/p/MyApps/s6/s6.cpp:95:98: instantiated from here
/usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/bits/stl_iterator_base_types.h:166:53: error:

/usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/bits/stl_iterator_base_types.h:167:53: error:

/usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/bits/stl_iterator_base_types.h:168:53: error:

/usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/bits/stl_iterator_base_types.h:169:53: error:

/usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/include/g++-v4/bits/stl_iterator_base_types.h:170:53: error:

is there any list of what algorithms are supported and what aren't? or will this get better with next gcc version?

Subject: Re: stl-compatibility
Posted by mirek on Mon, 15 Apr 2013 16:53:36 GMT
View Forum Message <> Reply to Message

size is now 'const', thanks.

As for other issue, we do not support iterator_traits (yet). I guess adding support should be possible; perhaps posting a small snippet of code triggering the error would help...

Mirek

Subject: Re: stl-compatibility
Posted by piotr5 on Mon, 15 Apr 2013 21:13:30 GMT
View Forum Message <> Reply to Message

well, since you're adding sorted containers now I guess I'll just wait till you have your own implementation of std::included(), so I deleted my sourcecode. it just took 2 sorted arrays and tried to figure out if one contains the other.

however, this problem inspired me to to start a new project: wrappers for all the different stl-algorithms. maybe I'll find a solution myself. shouldn't be too difficult to write some iterator-traits for the existing upp-containers.

Subject: Re: stl-compatibility
Posted by piotr5 on Tue, 16 Apr 2013 18:11:47 GMT
View Forum Message <> Reply to Message

I instantiated a class making use of each container's stl-compatibility to check what containers can be used. what I found is:

```
 testIterContainers<Vector<V> >();
 testIterContainers<Array<V> >();
/*no conversion from iterator to const_iterator*/ // testIterContainers<Segtor<V> >();
// testIterContainers<Index<V> >();
// testIterContainers<ArrayIndex<V> >();
 testIterContainers<VectorMap<int,V> >();
 testIterContainers<ArrayMap<int,V> >();
/*no conversion from iterator to const_iterator*/ // testIterContainers<SegtorMap<int,V> >();
// testIterContainers<One<V> >();
// testIterContainers<Buffer<V> >();
// testIterContainers<Mitor<V> >();
// testIterContainers<LinkElement<V> >();
// testIterContainers<LRUCache<V> >();
 testIterContainers<InVector<V> >();
/*no operator=(ConstIterator) for InArray*/ // testIterContainers<InArray<V> >();
```

// testIterContainers<SortedIndex<V> >();
 testIterContainers<SortedVectorMap<int,V> >();
/*no operator=(ConstIterator) for InArray*/ // testIterContainers<SortedArrayMap<int,V> >();
/*no conversion from iterator to const_iterator*/ // testIterContainers<BiVector<V> >();
/*no conversion from iterator to const_iterator*/ // testIterContainers<BiArray<V> >();

the containers I used all at least instantiate and return an iterator, the containers I commented out have no stl-compatibility. additional comments are talking of 2 compilation-problems: InArray delegates SetEnd to InVector resulting in

Upp::InArray<T>::SetEnd(Upp::InArray<T>::ConstIterator&) const [with T = Upp::Vector<in

Upp::InArray<T>::end() [with T = Upp::Vector<int>, Upp::InArr

/home/p/MyApps/stlpp/stlpp.h:532:53:   instantiated from

/home/p/MyApps/stlppTest/stlppTest.cpp:50:21:   instantiated from here

 nArray<Upp::Vector<int> >*)this)->Upp::InArray<Upp::Vector<int> >::iv.Upp::InVector<T>::End

/home/p/upp/uppsrc/Core/InVector.hpp:755:2: note: candidate is:
/home/p/upp/uppsrc/Core/InVector.h:200:20: note: Upp::InVector<Upp::Vector<int>*>::Iterator&
Upp::InVector<Upp::Vector<int>*>::Iterator::operator=(con
 st Upp::InVector<Upp::Vector<int>*>::Iterator&)
/home/p/upp/uppsrc/Core/InVector.h:200:20: note: no known conversion for argument 1 from

Upp::InArray<T>::SetBegin(Upp::InArray<T>::ConstIterator&) const [with T = Upp::Vector<

Upp::InArray<T>::begin() [with T = Upp::Vector<int>, Upp::InA

/home/p/MyApps/stlpp/stlpp.h:532:53:   instantiated from

/home/p/MyApps/stlppTest/stlppTest.cpp:50:21:   instantiated from here

 nArray<Upp::Vector<int> >*)this)->Upp::InArray<Upp::Vector<int> >::iv.Upp::InVector<T>::Begin

/home/p/upp/uppsrc/Core/InVector.hpp:749:2: note: candidate is:
/home/p/upp/uppsrc/Core/InVector.h:200:20: note: Upp::InVector<Upp::Vector<int>*>::Iterator&
Upp::InVector<Upp::Vector<int>*>::Iterator::operator=(con
 st Upp::InVector<Upp::Vector<int>*>::Iterator&)
/home/p/upp/uppsrc/Core/InVector.h:200:20: note: no known conversion for argument 1 from

stlppTest: 1 file(s) built in (0:05.84), 5848 msecs / file, duration = 5855 msecs, parallelization 0%

minimal testing program is

```
struct InArrayDebug
{
 InArray<int> c;
 InArray<int>::ConstIterator end() const {return c.end();}
 InArray<int>::ConstIterator begin() const {return c.begin();}
};
```

the problem with segtor is

/home/p/MyApps/stlpp/stlpp.h:75:2: note: no known conversion for argument 2 from

please note, I haven't updated svn yet,
svn st -v |sort |tail returns 5977 as the revision number on googlecode.

Subject: Re: stl-compatibility
Posted by mirek on Sat, 20 Apr 2013 16:59:54 GMT
View Forum Message <> Reply to Message

It should be now fixed, except Segtor, which is totally obsolete...

This now compiles:

```
#include <Core/Core.h>

using namespace Upp;

template <class T>
void Check()
{
 T a;
 std::includes(a.begin(), a.end(), a.begin(), a.end());

 const T& b = a;
 std::includes(b.begin(), b.end(), b.begin(), b.end());
}

CONSOLE_APP_MAIN
{
 Check< Vector<int> > ();
 Check< Array<int> > ();

 Check< BiVector<int> > ();
 Check< BiArray<int> > ();

 Check< Index<int> > ();
 Check< ArrayIndex<int> > ();
 Check< VectorMap<int, int> > ();
 Check< ArrayMap<int, int> > ();

 Check< InVector<int> > ();
 Check< InArray<int> > ();

 Check< SortedIndex<int> > ();
 Check< SortedVectorMap<int, int> > ();
 Check< SortedArrayMap<int, int> > ();
}
```