Subject: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Thu, 29 Jul 2010 20:17:28 GMT
View Forum Message <> Reply to Message

just a short question:

i noticed that all containers dont return a ref on the item created inside a container, when a copy Add is used: (here Array is example)

```
T&      Add();
void    Add(const T& x); //why void?
void    AddPick(pick_ T& x); //why void?
T&      Add(T *newt);
```

so why not having

```
T&      Add();
T&      Add(const T& x);
T&      AddPick(pick_ T& x);
T&      Add(T *newt);
```

the added elements in any case end up in the container, so their ref could be returned...

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Tue, 03 Aug 2010 09:15:01 GMT
View Forum Message <> Reply to Message

any comments on this?

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by andrei_natanael on Tue, 03 Aug 2010 10:53:49 GMT
View Forum Message <> Reply to Message

Hi,

That would be possible but we will lose some performance for nothing(or less). Taking the following code from Array implementation:

```
void    Add(const T& x)          { vector.Add(DeepCopyNew(x)); }
void    AddPick(pick_ T& x)       { vector.Add(new T(x)); }
```

to return the reference to added element, we have to find the element in vector, and that will take some time.

Andrei

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Tue, 03 Aug 2010 14:06:24 GMT
View Forum Message <> Reply to Message

why then, not do the same as in

```
 T&      Add()                  { T *q = new T; vector.Add(q); return *q; }
//like this
 void    Add(const T& x)          { T *q = DeepCopyNew(x); vector.Add(q); return *q; }
 void    AddPick(pick_ T& x)       { T *q = new T(x); vector.Add(q); return *q; }
```

the reference does not change, it's stored on heap, there should be no performance hit on that i think.

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by andrei_natanael on Tue, 03 Aug 2010 20:44:48 GMT
View Forum Message <> Reply to Message

That seems ok.

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Wed, 04 Aug 2010 07:57:37 GMT
View Forum Message <> Reply to Message

i changed things for Add semantic, take a look at it. it was no much work. maybe mirek has had something special in mind with void return.. i'll ask him.

another thing to be changed is maybe the void Set(int i, const T& x, int count = 1); behaviour. which also could be split up just like Insert..

EDIT: no one has downloaded previous version, so i just change it for one, where Insert and Set behaviour is already changed accordingly. plrese review it. maybe it can go upstream if nothing stands in way.

File Attachments
1) Core.rar, downloaded 413 times

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Thu, 05 Aug 2010 06:35:06 GMT
View Forum Message <> Reply to Message

any comments on that one? (need it for my app , like always...)

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mrjt on Thu, 05 Aug 2010 07:55:17 GMT
View Forum Message <> Reply to Message

I'd like to hear Mirek's reasons for doing it the old way because there is usually a good reason, especially for the NTL stuff.

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Thu, 05 Aug 2010 08:10:06 GMT
View Forum Message <> Reply to Message

ofcorse. i also was trying to imagine what problems there might be to have prevented this stuff to already be there. nevertheless wanted to provide already a solution to think visually about .

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mirek on Fri, 06 Aug 2010 09:27:46 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 29 July 2010 16:17just a short question:

i noticed that all containers dont return a ref on the item created inside a container, when a copy Add is used: (here Array is example)

```
T&      Add();
void    Add(const T& x); //why void?
void    AddPick(pick_ T& x); //why void?
T&      Add(T *newt);
```

Mostly because of standard usage pattern...

It might be a little bit confusing as those variants that are taking parameter make a copy of this parameter (and would return a reference to this copy).

Also note the existence of Top() - only one more line...

But I am not strongly opposed to changing this either....

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Fri, 06 Aug 2010 09:48:20 GMT
View Forum Message <> Reply to Message

the usage pattern is not changed too much in fact. all the functions changed (in code provided) returned void by before. so it's more of an extension..

the 'new' usage pattern would be, get direct access to the object, that 'somehow' has been added / replaced (Set) / or inserted. because it exists in container after the call, beeing it the same or a copy, the user wouldnt need to care. it is the one that remains in the container (maybe could even be more logic).

this would spare the usage of Top() or even operator[](GetCount()-1) like in some places, after having void Add(const & T x). this by the way uses the rather 'implicit' assumption that an added item is always placed last...

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Mon, 09 Aug 2010 06:50:28 GMT
View Forum Message <> Reply to Message

to which conclusion have you guys come? should it be extended?

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mirek on Fri, 13 Aug 2010 07:38:03 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 09 August 2010 02:50to which conclusion have you guys come? should it be extended?

I am barely positive about this, but before I waste time fixing documentation, I still cannot imagine usage scenario which I would like - I think that you either use Vector to store "values" and then you do not need those references as return values, or to create objects inside.

I mean, just tell me little where it does help Practical example is most welcome.

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Mon, 16 Aug 2010 07:06:44 GMT
View Forum Message <> Reply to Message

it's actually the same behaviour as with T& Array::Attach(T* newt);
and merly a logical unification of interface, that anything that ends beeing an object, no matter newly added or as copy, should be instantly available, without the need to again access the container to get the same. here, it actually doesnt matter the container type, it's same situation for Vector and Array.

a practical use is this:

crating new container objects, based on some 'template' objects, and remodifying stuff that is actually different, on the new created object, pushing it somewhere to do something. this would use in case of Array:
void Array::Add(cibst T&),
then
T& Array::operator[](int i) with Array::GetCount()-1.
actually 3 invokations, that could be done in one.

it's maybe more of estetic use but could again add to Ultimate's short and reading friendly code

---

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Mon, 30 Aug 2010 08:46:52 GMT
View Forum Message <> Reply to Message

hey mirek, what is your final decision on this one?
sorry if disturbing in vacations

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mirek on Mon, 06 Sep 2010 08:53:08 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 16 August 2010 03:06then
T& Array::operator[](int i) with Array::GetCount()-1.
actually 3 invokations, that could be done in one.

Could be done with Top().

I am likely to make this change, but I am still waiting for 3rd party opinion...

Subject: Re: why not "T & Add(const T & x)" in all containers

Posted by kohait00 on Mon, 06 Sep 2010 09:52:11 GMT
View Forum Message <> Reply to Message

who is 3rd party?

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mirek on Wed, 08 Sep 2010 07:00:52 GMT
View Forum Message <> Reply to Message

Not me and not you (That makes it 3rd, right?)

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by rylek on Wed, 08 Sep 2010 08:24:30 GMT
View Forum Message <> Reply to Message

Hi there!

It is a pleasure to play the third party, though the semantics of the term are highly disputable here. I think it's possible Mirek originally didn't include the return reference in Vector::Add to emphasize the fact that such references into Vectors are volatile in principle (specifically they are periodically invalidated by the Add function itself while reallocating the physical Vector data). But, of course, the same argument holds for Add() which does return the reference. Moreover, Array::Add(T *newobj) also returns the reference, albeit for different reasons.

U++ also decidedly avoids returning references in pick assignment operators, which is natural because a "chain" assignment (a = b = c) in such cases exhibits undesirable behaviour (by destroying b). But Vector::Add(const T&) doesn't have this problem; the only thing that has to be avoided is rather artificial constructs of the form

vector.Add(vector.Add(obj))

exactly because of the periodical Vector reference invalidation. But then again you can run into exactly the same problems by writing, e.g.

vector.Add(vector[5]);

so that this is no specific of Vector::Add(const T&) either. To sum it all up, I currently see no practical reasons against modifying Vector::Add to include the return reference. I would rather say that it's like updating old code to match interface standards adopted / developed later on, in fact I believe Vector is one of the very oldest things in U++ (although it's been rewritten quite a few times since its inception).

Regards

Tomas

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Wed, 08 Sep 2010 08:54:44 GMT
View Forum Message <> Reply to Message

thanks, nice outline

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mirek on Wed, 08 Sep 2010 09:30:02 GMT
View Forum Message <> Reply to Message

rylek wrote on Wed, 08 September 2010 04:24
exactly because of the periodical Vector reference invalidation. But then again you can run into exactly the same problems by writing, e.g.

vector.Add(vector[5]);

Actually, above code is OK, this was already improved (because improvement is possible by changing the order of operations in implementation).

The only last one in this zone is Insert... (which right now ASSERTs, but I guess it should be fixed too).

Mirek

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mirek on Fri, 17 Sep 2010 06:11:22 GMT
View Forum Message <> Reply to Message

OK, it now returns T&...

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Fri, 17 Sep 2010 07:15:15 GMT
View Forum Message <> Reply to Message

hi mirek, thanks.
just a question aside: why does DeepCopyConstryct return T& instead of T*? it kinda differs from the other 'patterns' DeepCopyNew, ::new(p) T() etc..which all return T* at that level.

EDIT: i think i understand why..there occure strange compile errors for ArrayCtrl Vector< Vector< Value > > things. and i kind of cant figure out why and how to potentially fix it.

nonetheless, Add is done, what about Insert and Set behaviour? should do the same..

i've added the current proposal, based on your changes.

File Attachments

1) Core.rar, downloaded 358 times

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Thu, 21 Oct 2010 07:02:13 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Fri, 17 September 2010 09:15hi what about Insert and Set behaviour? should do the same..

just a reminder, since i've still got the changes around in my code, just to know if they once might become upstream. or what your position is..

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mirek on Thu, 21 Oct 2010 09:10:42 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 21 October 2010 03:02kohait00 wrote on Fri, 17 September 2010 09:15hi what about Insert and Set behaviour? should do the same..

just a reminder, since i've still got the changes around in my code, just to know if they once might become upstream. or what your position is..

Insert/Set has sort of problem as it can insert more than single element, and even worse, it can insert ZERO elements. (Note that for single element it already returns T&).

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Thu, 21 Oct 2010 09:54:17 GMT
View Forum Message <> Reply to Message

i thought of it. and there is a solution in the code.
i just split the function, i.e. Set:

original:

```
void    Set(int i, const T& x, int count = 1);
```

split:

```
T&      Set(int i, const T& x);
void    Set(int i, const T& x, int count);
```

which is syntactically the same, the omited count defaults to setting only one element and returns the ref..no changes in user code. when using count, the other function evaluates.

insert is same:

```
void    Insert(int i, const T& x, int count = 1);
```

split:

```
T&      Insert(int i, const T& x);
void    Insert(int i, const T& x, int count);
```

i'm actually working with the code for some while now, and haven't noticed any misbehavior, so the sub layers of upp deal well with it. consider it again. i think it can contribute to the comfort using upp containers, and of corse again a little bit of speed optimization at user level..and shorten the code a bit.

attached the current source based snapshot..

File Attachments

1) containers_Core.rar, downloaded 357 times

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Wed, 30 Mar 2011 08:54:02 GMT
View Forum Message <> Reply to Message

just wanted to poke gently w.r.t that issue..
i've got some fixes in my local tree and am cleaning up

is there interest/chance for this to be upstream sometime or should i consider it lost? (future coding needs to take account of it)

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mirek on Sat, 16 Apr 2011 18:31:57 GMT
View Forum Message <> Reply to Message

OK. I really dislike this change, but I cannot bring any rational argument against it.

So it is now in Core

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Sun, 17 Apr 2011 12:46:56 GMT
View Forum Message <> Reply to Message

cool..

just for completion sake..there are those changes for Map and Index, that can expose their elements now as well, just by using the api..was it your intention?

for easy handling, i just packed them, so you can replace them for svn review. to current revision 3346

there is also the Insert(int i, T* newt) that seems to be overseen.
thats why Vcont.h and .hpp are in the rar as well.

it'd be nice to have all together

thanks for your time

File Attachments
1) Core.rar, downloaded 389 times

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by mirek on Sun, 17 Apr 2011 19:43:00 GMT
View Forum Message <> Reply to Message

Sight... great, another change I really hate.

OK, it is there. I have even "fixed" some inserts you forgot...

Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by kohait00 on Mon, 18 Apr 2011 07:00:49 GMT
View Forum Message <> Reply to Message

thats definitely cool, thanks a bunch..