

## Re: Is CArray best for this...

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2008-01/msg00841.html>

---

- *From:* Norbert Unterberg <[nunterberg@xxxxxxxxxxxxxxxxxxxxx](mailto:nunterberg@xxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 17 Jan 2008 23:16:33 +0100
- 

David Ching schrieb:

"Norbert Unterberg" <[nunterberg@xxxxxxxxxxxxxxxxxxxxx](mailto:nunterberg@xxxxxxxxxxxxxxxxxxxxx)> wrote in message [news:e36cXXUWIHA.4712@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:e36cXXUWIHA.4712@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

I disagree here. I do not see how in any way a `std::vector` can be "foreign" to windows. The only MFC like thing about `CArray/CList` I see is the naming convention. I can not remember any Windows/MFC API that expects or returns `CArray` or `CList`.

On the contrary, I think that `CArray` is more "foreign" as `std::vector` since `std::vector` is part of the C++ standard and `CArray` is not. What do you think is missing from `std::vector` to make it attractive to windows developers?

Nothing is missing, but what is there appeals to computer scientists and not necessarily to the domain-specific sensibilities of Windows app developers. Iterators have strange syntax, some STL construct (forget which one) needs to reference ".first" and ".second" is totally bizarre, etc. This is completely acceptable, and even desirable, for computer scientists, but not for app writers who are not interested in becoming experts in collections per se, only in that they allow them to write their apps.

Quick, without much thinking, can you write code that iterates a `CArray`, `CList` and `CMap` without looking at the docs, and get it right in the first try? I was used to these classes but I always had to look up the `CList` and `CMap` iterating. But forward or backward iterating through a `std::vector`, list, array, ... is always the same, no need to learn more than one pattern. I always get these right in the first try without looking it up.

Iterating through `std::c++` containers do not look more strange than iterating a `CList` or a `CMap`, IMHO. The difference is that the C++ algorithms implement concepts and not just a single class. So you learn the concept of iterating once, and then you can iterate everything. This is what helps me to get my job done.

See, use `std::vector` and you do not need any work around.  
The std containers and iterators look strange when you start using them, but

Re: Is CArray best for this...

their concept is much better thought out than the MFC collections.

I thought the same when I first saw the std library, but I changed my mind the better technology got me.

The thing of it is: these library writers have nothing else to do except make a good performing, usable class. Is it asking too much for them to make it sensible out of the box?

You mean as sensible as using && for AND and || for OR? I think this learning curve can never be avoided for new technology, and for me it was worth the effort. You get a nice toolbox where most things fit nicely together and snap into place once you got the concepts.

Struggling to use it should not be the required price to make one "change his mind." .NET provides collections that make sense without necessarily becoming an expert in order to use them and persuade one of their inherent superiority.

I am currently working on my first C# and .NET project, and I see what you mean. But even there it took a learning curve to realize that a vector is a List and a list is a LinkedList ;-)  
But seriously, .NET collections have general concept of iterating (by IEnumerable), and so do the std C++ containers (class iterator). For me this is just different syntax for very similar basic concepts, and the .NET containers are very similar to the std C++ containers. In C++ you call this "for computer scientists, but not for app writers", but in .NET you call the same concept "inherent superiority". Hmm...  
I just have to admit that the C# syntax makes the concepts more visible by looking at the class definitions, whereas in C++ these concepts are somewhat hidden in the background. But sometimes we still need native C++.

So it's possible, but for whatever reason, it is not a goal of STL, Boost, or from what I gather, any

BTW, I am not talking about the STL as some external third party library (which is quite dead), but about the current C++ standard.

of the other C++ constructs being worked on right now. It can be perceived as arrogance on their part, and more and more that's what I'm seeing it as.

Well, some of this arrogance slipped into C# version 3, so it can't be that bad. C# now also implements something similar to Boost.Lambda and Boost.Bind.

Ok, I think this is enough on this topic, I think we are far off topic now. I just wanted to push you a little into the C++ standard library direction and give it a try before you put it aside as too complicated.

Norbert

Re: Is CArray best for this...

Re: Is CArray best for this...