

# Re: inhibit compiler warning C4624 for a class hierarchy

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2007-01/msg00432.html>

---

- *From:* "Ben Voigt" <[rbv@xxxxxxxxxxxxxx](mailto:rbv@xxxxxxxxxxxxxx)>
  - *Date:* Tue, 16 Jan 2007 09:21:14 -0600
- 

"Charles Wang[MSFT]" <[changliw@xxxxxxxxxxxxxxxxxxxxxx](mailto:changliw@xxxxxxxxxxxxxxxxxxxxxx)> wrote in message [news:IJdfiAAOHHA.2080@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:IJdfiAAOHHA.2080@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Hi rbv,

The undefined behavior means that the behavior is decided by the compiler. Different compilers may cause different behaviors. C++ standard does not define a consistent behavior for this situation.

For your four requirements, I am afraid that it is hard to implement unless you do some changes. Even if you use template allocator/deallocator, your none-virtual Destroy method is defined in your base class OpNotification and it uses "delete this". This only release your base instance space,

I would change that call to invoke the deallocator.

however if your derived class instance includes additional data members, this may cause memory leak due to some left spaces not released.

No allocator I'm aware of ever frees less than the entire block. Leaks could appear due to member destructors not being called, for this reason this hierarchy will permit only POD members.

Since my template allocator/deallocator would invoke (`new BYTE[] / delete []`), it's guaranteed to free the entire block originally allocated, even though the deallocator didn't know the original type. I'd lose member initializers.... well, I could invoke a member function through the template typename argument to initialize the buffer, ala two-step construction. What I lose is the ability to for each buffer initializer to define a set of mandatory arguments. I guess I can make the two-step initialization explicit.

Then there wouldn't be any mismatch between constructor and destructor calls, because neither would be called. The dynamic type of every object

Re: inhibit compiler warning C4624 for a class hierarchy

would be BYTE[]. Compiler behavior would be well defined by the standard.

The really nice thing with constructors, though, is the way they aren't implicitly inherited.

What I ultimately want is a family of structures sharing a common header so that they can be used polymorphically, with a convenient initialization syntax. Multi-level (not multiple) "inheritance" is a plus, but not needed at the moment. Just like the ENHMETARECORD family of structures, but not as ugly.

Why do not you use a macro for releasing your object? It might be the least work in your situation.  
For example:  
#define \_\_ReleaseB(pB) \  
delete pB;\n pB=NULL;

That really doesn't solve any problems. I need to be able to polymorphically destroy the request objects if I encounter an error condition, such as the port being closed.

Hope this helps! Please feel free to let us know if you have any other questions or concerns.

Thanks for all your help with this. Getting this done right is really going to help future maintainability.

Sincerely yours,  
Charles Wang  
Microsoft Online Community Support

=====  
When responding to posts, please "Reply to Group" via your newsreader so that others may learn and benefit from this issue.  
=====

This posting is provided "AS IS" with no warranties, and confers no rights.  
=====

Re: inhibit compiler warning C4624 for a class hierarchy