[**C++接口与实现分离的2种方法**](http://www.cppblog.com/mzty/archive/2007/08/06/29441.html)

方法一：使用另一个实现类分装类的私有成员和函数，这种方法称为Pimpl方法。  
  
test.h

http://www.cppblog.com/Images/OutliningIndicators/None.gif#pragma once  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include "shared\_ptr.hpp"  
http://www.cppblog.com/Images/OutliningIndicators/None.gifclass CTest  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifpublic:  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    CTest(void);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    ~CTest(void);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void DoSomething();  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifprivate:  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    class CTestImp;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    boost::shared\_ptr<CTestImp> pimpl\_;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif};

test.cpp

http://www.cppblog.com/Images/OutliningIndicators/None.gif#include "Test.h"  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <iostream>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifclass CTest::CTestImp  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifprivate:  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    CTestImp(CTestImp const &){}  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    CTestImp & operator=(CTestImp const &){}  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifpublic:  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    CTestImp(){}  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void DoSomething();  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif};  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifvoid CTest::CTestImp::DoSomething()  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    // do something.  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    std::cout<<"Imp class do something."<<std::endl;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifCTest::CTest(void)  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    boost::shared\_ptr<CTestImp> pImp(new CTestImp);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    pimpl\_ = pImp;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifCTest::~CTest(void)  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifvoid CTest::DoSomething()  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    pimpl\_->DoSomething();  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

方法二：使用抽象类来实现接口与实现的分离。  
  
x.h

http://www.cppblog.com/Images/OutliningIndicators/None.gif#pragma once  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <stdio.h>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include "shared\_ptr.hpp"  
http://www.cppblog.com/Images/OutliningIndicators/None.gifusing namespace boost;  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifclass X  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifpublic:  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    virtual void f() = 0;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    virtual void g() = 0;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifprotected:  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    ~X() { printf("~X\n");}  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif};  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifshared\_ptr<X> createX();

x.cpp

http://www.cppblog.com/Images/OutliningIndicators/None.gif#include "X.h"  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <stdio.h>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifclass X\_impl: public X  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifprivate:  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    X\_impl(){};      
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    X\_impl(X\_impl const &);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    X\_impl & operator=(X\_impl const &);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifpublic:  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    ~X\_impl(){printf("~X\_impl\n");};  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    virtual void f()  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif      printf("X\_impl.f()\n");  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    virtual void g()  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif      printf("X\_impl.g()\n");  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifprivate:  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    friend shared\_ptr<X> createX();  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif};  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifshared\_ptr<X> createX()  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    shared\_ptr<X> px(new X\_impl);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    return px;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

总结：  
       接口与实现的分离，有助于我们对代码实现的保护，特别是如果我们开发lib共别人使用使，更要注意。在实现分离的过程中，最好采用上面的第一种方法的智能指针boost::shared\_ptr的实现，简单安全。

#### [#](http://www.cppblog.com/mzty/archive/2007/08/06/29441.html#32209) re: C++接口与实现分离的2种方法 2007-09-14 14:10 [梦在天涯](http://www.cppblog.com/mzty/)

第二种方法的create方法可以作为X class接口类的static方法！   
第一种方法中，如果pimpl class需要访问CTest接口类基类的话，可以把pimpl class申明为CTest接口类的friendly类。

