C 编程基础入门:友元接口计算机等级考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/645/2021_2022_C___E7_BC_96_E7_A8_8B_E5_c97_645878.htm 限制你的友元类数量 也许你遇到过这样的情况： 在一个类有一组策略，而且这组策略的实现都需要访问A的一些成员，而且这些成员并不希望被其它类所访问。 一般这些成员被期望设置为保护或者私有的，并且这组策略被当作这个类A的友元类。如： Code class Strategy1. class Strategy2. class Strategy3. class A { public: friend class Strategy1. friend class Strategy2. friend class Strategy3. private: void _foo(). int _bar. }. 现在，假如你需要添加新的策略Strategy4为了维持这种微妙的关系，你需要把Strategy4添加为类A的新的友元类。 随着策略的增加，这个过程不断扩展A的友元类，最终你恐怕不会喜欢你看到的代码。并且由于每次增加策略都需要使得依赖A的代码重新编译，这里一定有什么不妥之处。 我想到一种解决方法，可以让你的代码看上去不算太混乱。来源：www.100test.com 首先，既然这组策略以相似的情况出现在A的周围，那么它们可能有相似之处。比如它们可能需要访问A的同一部分成员。 那么假如通过一个代理类来访问这些成员，那么这组策略就不必都是A的友元，只要这个代理是A的友元即可。百考试题论坛 这个代理我称之为友元接口。 Code class IFriendA_Strategy. class A { public: friend class IFriendA_Strategy. private: void _foo(). int _foo2(int p) const. int _bar. }. class IFriendA_Strategy { protected: typedef IFriendA_Strategy Friend. // 统一友元接口的访问方式 // 想必这组函数的实现不必给出了吧来源：考试大 static void _foo(A