深度探索C 对象模型(3) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/133/2021_2022__E6_B7_B1_E5_BA_A6_E6_8E_A2_E7_c97_133877.htm 介绍 多态是一种威力强大的设计机制,允许你继承一个抽象的public接口之后,封装相关的类型,需要付出的代价就是额外的间接性--不论是在内存的获得,或是在类的决断上,C 通过class的pointer和references来支持多态,这种程序风格就称为"面向对象". 大家好，雷神关于《深度探索C 对象模型》笔记终于又和大家见面了，速度慢的真是可以。好了不浪费时间了，直接进入主题。 这篇笔记主要解决了几个常常被人问到的问题。 1、C 支持多重继承吗？ 2、结构和类的区别是什么？ 3、如何设计一个面向对象的模型？ C 支持多重继承（JAVA和C#不支持多重继承），虽然我想我可能一辈子用不到它这一特性（C 是雷神的业余爱好），但至少我要知道它可以。典型的多重继承是下面这个： //iostream 从istream 和 ostream 两个类继承。 class iostream:public istream,public ostream {.....}. 结构struct和类class到底有没有区别？VCHELP上前几天还看到一个帖子在讨论这个问题。其实结构和类真的没什么区别，不过我们需要掌握的是什么时候用结构好，什么时候用类好，当然这没有严格的规定。通常我们混合使用它们，从书上的例子，我们可以看出为什么还需要保留结构，并且书上给出了一个方法： struct C_point{.....}. //这是一个结构 class Point { public: operator C_point(){return _c_point.} //.... private: C_point _c_point. //.... } 这种方法被成为组合（composition）.它将一个对象模型的全部或部分用结构封装起来，这样做的好处是你

既可以在C 中应用这个对象模型，也可以在C中应用它。因为struct封装了class的数据，使C 和C都能有合适的空间布局。面向对象模型是有一些彼此相关的类型，通过一个抽象的base class（用来提供接口），被封装起来。真正的子类都是通过它派生的。当然一个设计优秀的对象模型还必须考虑很多的细节问题，雷神根据自己的理解写出一个面向对象模型的代码，大家可以看看，高手请给指出有没有问题。雷神先谢了。 思路：我想要实现一个人员管理管理的对象模型，雷神一直在思考一个人员管理的组件（当然最终它会用C#实现的一个业务逻辑对象，并通过数据库控制对象和数据库进行交互，通过WEB form来显示界面）。这里借用一下自己的已经有的的想法，用C 先进行一下实验，由于只是为了体会面向对象的概念，我们采用面向对象的方法实现一个链表程序，而且没有收集信息的接口。信息从mina()函数显式给出。这个对象模型应该可以实现对人员的一般性管理，要求具备以下功能： 创建一个人员信息链表 添加、删除人员信息 显示人员信息 //********** //PersonnelManage.cpp //创建人：雷神 //日期：2002-8-30 //版本： //描述： //********** #include #include //基类,是此对象模型的最上层父类 class Personnel { friend class point_list. //用来实现输出链表,以及插入或删除人员的功能. protected: char serial_number[15].//编号 char name[10].//名称 char password[15]//口令 Personnel *pointer. Personnel *next_link. public: Personnel(char *sn,char *nm,char *pwd) { strcpy(serial_number,sn). strcpy(name,sm). strcpy(password,pwd).

next_link=0. } Personnel() { serial_number[0]=NULL. name[0]=NULL. password[0]=NULL. next_link=0. } void fill_serial_number(char *p_n) { strcpy(serial_number,p_n). } void fill_name(char *p_nm) { strcpy(name,p_nm). } void fill_password(char *p_pwd) { strcpy(password,p_pwd). } virtual void addnew(){} virtual void display() { cout}. //下面是派生的子类，为了简单些我在把子类进行了成员简化。 //思路:由父类派生出成员子类，正式成员要求更详细的个人资料,这里省略了大部份. 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com