c++ prime plus第十四章

私有继承

私有继承是C++面向对象编程中的一种继承方式。与公共继承和保护继承不同的是，私有继承派生类的对象无法访问基类的非公有成员。具体来说，私有继承将基类的公有和保护成员转化为派生类的私有成员。这样，在派生类对象中，基类的公有和保护成员就不能直接访问了，只能通过派生类提供的公共接口（即成员函数）来间接访问。而基类的私有成员在派生类中是不可访问的。私有继承的主要优点是可以隐藏基类的实现细节和公共接口，同时继承了基类的接口和实现。这样，可以在派生类中重新定义基类的接口，从而实现新的功能或更好地适应新的需求。总之，私有继承是C++面向对象编程中的一种继承方式，可以隐藏基类的实现细节和公共接口，同时继承了基类的接口和实现。在使用私有继承时，需要注意访问权限的限制，以及重新定义基类接口带来的影响。

使用包含还是私有继承

在C++中，继承和包含都是实现类的复用的两种主要方式。在确定是否使用继承和包含时，需要考虑以下几个因素：

类之间的逻辑关系：如果派生类是基类的具体化，或者属于“is-a”关系，则应该使用继承；而如果派生类只是需要调用基类的某些功能，则应该使用包含。

可访问性：继承会将基类的公共接口暴露给派生类和其他类，而包含只有通过对象的接口才能访问。如果派生类需要访问基类的私有成员，应该使用私有继承；而如果不需要访问，则可以使用包含。

设计的灵活性：继承会将基类的设计限制透露给派生类，而包含则允许更灵活的设计。如果基类可能会发生变化，并且这个变化会影响到派生类，则应该考虑使用包含。

总之，对于需要复用其他类的功能的时候，可以通过继承和包含来实现，具体应该使用哪种方式，需要根据具体情况进行选择，综合考虑类之间的关系、可访问性和设计的灵活性等因素。

保护继承

保护继承是C++中的一种继承方式，与私有继承和公共继承不同，它将基类的共有和保护成员转化为派生类的保护成员，而将基类的私有成员仍保持为私有成员。这样，在派生类中，对于基类的共有和保护成员，外部不能直接访问，只能通过派生类的成员函数来访问。保护继承通常用于实现继承和封装的共同作用。派生类通过保护继承，继承了基类的接口和实现，同时可以通过再定义方法来重新定义接口。这提供了更好的控制派生类对基类接口访问权限的能力，从而实现更安全的封装。

多重继承

C++支持多重继承，即从多个基类派生出一个派生类。 在多重继承中, 派生类会继承多个基类的成员函数和成员变量。

多重继承的语法与单一继承类似，只是在类名后面可以通过逗号分隔添加多个基类名。多继承的类具有多个父类，因此要考虑其访问控制，以确保派生类可以访问正确的基类成员。

需要注意的是，多个基类中如果出现成员函数或成员变量名称相同，就会出现命名冲突的问题。 解决方法是使用作用域解析运算符(::)或者在派生类中覆盖基类中的成员函数（也可以使用using关键字）。在使用多重继承时，需要慎重考虑继承关系的合理性，并合理控制访问权限以避免潜在的问题

类模板

在 C++ 中，类模板可以让我们写出可以处理多种数据类型的通用类。类模板定义了一个通用的类模板，其中涉及的数据类型可以用作参数进行推导，以最终生成特定的类或函数。

类模板的定义方式与函数模板类似，使用 template 关键字加上类型参数列表来定义。类型参数列表中可以有多个类型参数，每个参数可以用于指定可以用在类中的不同类型。类模板除了可以定义成员变量和成员函数外，还可以定义静态成员变量和静态成员函数。可以使用模板参数作为静态成员变量的类型或返回值类型。

c++编程思想第九章：

在多继承中，一个派生类可以同时继承多个基类，从而获得多个基类的特性。虚基类是多继承中的一个重要概念，它可以解决由多继承带来的二义性问题。虚基类背后的实现机制是通过在派生类对象中存储虚基类指针来实现的。当一个派生类同时继承多个基类时，如果这些基类中有一个或多个是虚基类，那么在派生类对象中就会存储一个指向虚基类的指针。这个指针指向的是派生类对象中的虚基类子对象，而不是基类的子对象。这样就可以保证虚基类在派生类对象中只有一份拷贝，从而避免了二义性问题。虚基类的构造顺序和普通基类的构造顺序不同，虚基类的构造是在最派生类的构造函数中完成的。当最派生类的构造函数被调用时，它会先调用虚基类的构造函数，然后再调用非虚基类的构造函数。析构的顺序与构造的顺序相反，先调用非虚基类的析构函数，再调用虚基类的析构函数。