**第四章课后习题新得和感受**

第四章就是每种继承方式以及其派生类对基类成员的继承性这一方面给我感觉很难懂，一直不是很明白，之后才了解了，公有继承时，对基类的公有成员和保护成员的访问属性不变，派生类的新增成员可以访问基类的公有成员和保护成员，但是访问不了基类的私有成员。派生类的对象只能访问派生类的公有成员（包括继承的公有成员），访问不了保护成员和私有成员。保护继承中，基类的公有成员和保护成员被派生类继承后变成保护成员，派生类的新增成员可以访问基类的公有成员和保护成员，但是访问不了基类的私有成员。派生类的对象不能访问派生类继承基类的公有成员，保护成员和私有成员。让我感受很多的就是派生类构造函数和析构函数执行的规则，先从前往后执行每一个派生类的构造函数，再从后往前执行析构函数，我觉得充分体现了C++逻辑的严谨性，更安全更可靠保护数据；而虚基类更是消除了二义性，使得功能更加完善准确，对我的感受是蛮大的，第五章运算符重载也使我感受到了C++的可创造性，这一点明显比C语言要强上很多，更简洁。

第四章知识点总结

1.当一个类通过公有方式从基类继承时，基类中的公有成员在派生类中也是公有的。不指明继承方式关键字public时，编译器会默认继承方式为private或者protected。

class <派生类名字>:[继承方式] <基类名字>{派生类新增加的成员};

//单继承派生类的声明格式

2.基类的所有私有成员仅在基类中可见，而在派生类中是不可见的，但派生类对象会为基类中的所有私有成员分配空间。尽管在派生类中不能直接访问基类的私有成员，但可以通过间接地方式(调用从基类继承来的公有成员函数)进行。

3.使用using声明可以改变成员在派生类中的访问限制 。如果基类的某个公有成员函数在继承类中不适合，则可以通过using声明将其转变为私有成员函数，从而使它在派生类中隐藏起来。

#include <iostream>

using namespace std;

class BC{

public:

void set\_x(float a){x = a;cout << x <<endl;}

private:

float x;

};

class DC:public BC {

public:

void set\_y(float b){y =b;cout << y <<endl;}

private:

float y;

using BC::set\_x;//使用using声明可以改变成员在派生类中的访问限制

};

int main(){

DC d;

d.set\_y(4.31);//ok

//d.set\_x(-8.12);//报错，set\_x在DC里是私有的

d.BC::set\_x(-8.12);//ok

}

4.如果派生类添加了一个数据成员，而该成员与基类中的某个数据成员同名，新的数据成员就隐藏了继承来的同名成员（调用时调派生类的数据成员）。同名函数亦是如此。

5.在没有继承的情况下，保护成员和私有成员类似，只在该类中可见，在公有继承方式下，保护成员和私有成员具有不同性质。

1）派生类可对从基类继承来的保护成员进行访问，亦称基类的保护成员在派生类中是可见的。（派生类不能访问一个基类对象的保护成员，因为基类对象属于基类，不属于派生类）

2）基类的私有成员在派生类中是不可见的。

6.在public派生中，基类的私有成员只能被基类的其他成员函数访问（除了friend函数）；基类的保护成员除了能被基类的其他成员函数访问外，还能被类层次结构中的所有成员函数访问；基类的公有成员可被派生类的任何函数访问。

7.一般来说，应避免将数据成员设计为保护类型，应该采用私有数据成员与相应保护型访问函数结合的模式，便于实现数据隐藏（复杂数据成员例外）。

8.三种继承的总结：

公有继承时，对基类的公有成员和保护成员的访问属性不变，派生类的新增成员可以访问基类的公有成员和保护成员，但是访问不了基类的私有成员。派生类的对象只能访问派生类的公有成员（包括继承的公有成员），访问不了保护成员和私有成员。

保护继承中，基类的公有成员和保护成员被派生类继承后变成保护成员，派生类的新增成员可以访问基类的公有成员和保护成员，但是访问不了基类的私有成员。派生类的对象不能访问派生类继承基类的公有成员，保护成员和私有成员。