1. 继承

C++的重要特征就是代码重用，通过继承机制可以利用已有的数据类型来定义新的数据类型，新的类不仅拥有旧类的成员，还拥有新定义的成员。

二 .继承方式

Public公有继承

protected保护继承

private私有继承

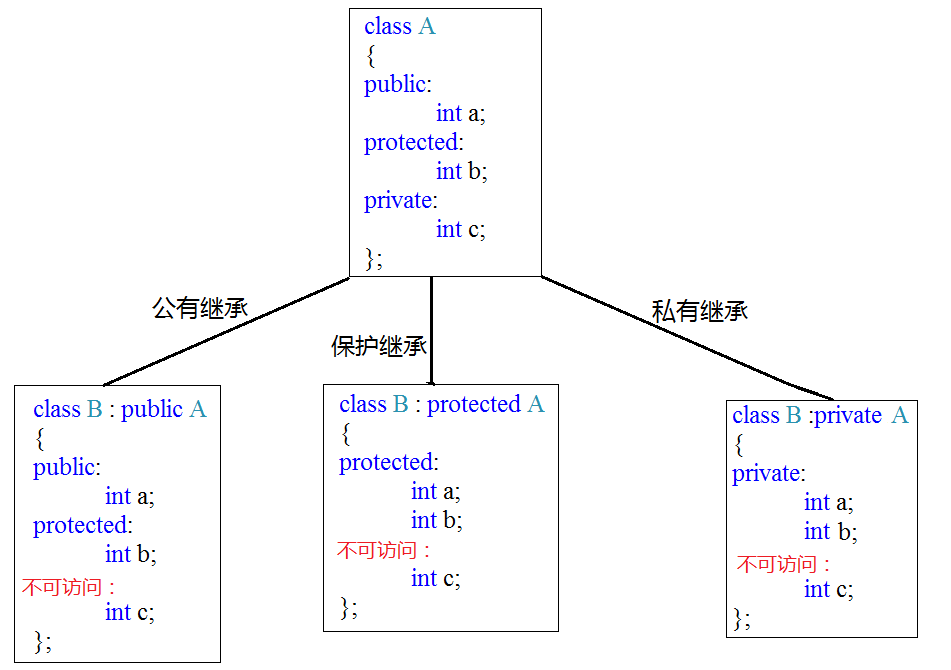
所有继承方式可以通过下图清晰，完整地记住。

基类中的private属性成员不论继承方式是什么，都是无法访问的。

基类中的public属性和protected属性成员，在public继承方式下属性不变。

基类中的public属性和protected属性成员，在protected继承方式下属性都变为protected

基类中的public属性和protected属性成员，在private继承方式下属性都变为private:



子类public继承了基类的protected属性成员，在子类中依然是protected，在类外依然是不能访问的。

1. 子类继承父类的成员变量和函数

子类会继承父类中的所有成员变量和成员函数，包括不能访问的private属性成员。子类的成员是由父类成员叠加子类的新成员而形成的。即子类的大小是由父类的所有成员加上子类的新成员构成的。

子类会继承父类的成员属性，成员函数，但是不会继承父类的构造函数和析构函数。

只有父类自己知道构造函数和析构函数是如何构成的。每一个子类都需要自己创造构造函数和析构函数。

1. 继承中的构造和析构顺序

先构造基类，再构造子类

理解：子类是由基类继承的，当然是先把基类构造好了才能构造子类 。

先析构子类，再析构基类

理解：子类是基于基类的，若先析构基类，则子类就无法析构了。

1. 继承中的同名成员如何区分？

设Base为基类，Son为子类

Son s1;

S1.m\_A;

S1.Base::m\_A;

成员函数同理

理解：子类和父类拥有同名的成员变量和函数，子类会把父类的所有同名版本（包括重载版本）都隐藏掉，要想调用父类的成员变量和函数，必须加作用域才能调用到。

1. 多继承

同时从多个类继承，但是由于多个类继承会导致函数，变量等同名的歧义。不建议使用。

实例：

Class Son : public Basic1, public Basic2

若有多个基类有重名成员变量或函数，依然是用作用域来调用各自基类的成员变量或函数。

1. 菱形继承

两个派生类A1，A2继承同一个基类A，同时又有一个类B同时多继承A1，A2，这种继承叫做菱形继承或者钻石型继承。

产生的问题：

1. A1，A2都继承了A的成员变量和函数，当B调用成员变量和函数时，会产生二义性的问题。
2. B继承了A1，A2的成员变量和函数，即继承了两份A的成员变量和函数，是重复的，

造成资源浪费。

解决菱形继承的方法：虚继承

虚继承写法：

class Sheep : virtual public Animal

虚继承的字面意思是：虚假的继承，即只是表面上是继承的，真实情况是我们对虚基类Animal的成员和函数用虚继承不是直接继承，子类Sheep只是单纯地存储着（vbptr）（virtual basic pointer）虚基类指针，指向vbtable(虚基类表)，虚基类表里存储着从vbptr（虚基类指针）开始到虚基类Animal的首地址的偏移量，通过这个偏移量子类就可以找到相应的虚基类变量。

举例：

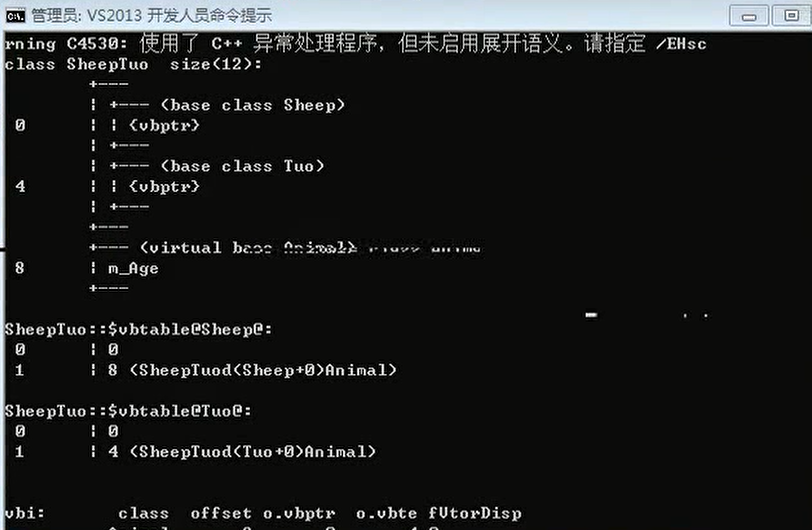
比如说

class Sheep : virtual public Aniaml{}

class Tuo : virtual public Animal{}

class SheepTuo : public Sheep, public Tuo{}

可以看到，子类Sheep，Tuo是虚继承于虚继类Animal的，子类Sheep中就不会再存储虚基类Animal的成员变量和成员函数，而存储的是vbptr（虚基类指针），指向子类Sheep的vbtable，在vbtable中存储的是子类Sheep要继承的虚基类Animal的成员变量相对于子类sheep的vbptr的偏移量8，即偏移8个值就可以得到虚基类中的变量。



上图是多继承于子类Sheep, Tuo的类SheepTuo的结构组成，可以看出，子类Sheep，Tuo用虚继承以后，SheepTuo中就只有一份虚基类Animal的成员变量m\_Age了。这就是解决菱形继承的虚继承方法。