C++ Inheritance

继承和派生

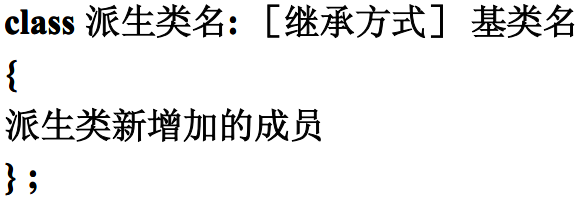
继承：一个新类从已有的类那里获得其已有特性

已存在的类称为： 

新建立的类称为-  

派生：从已有的类产生一个新的子类

声明派生类



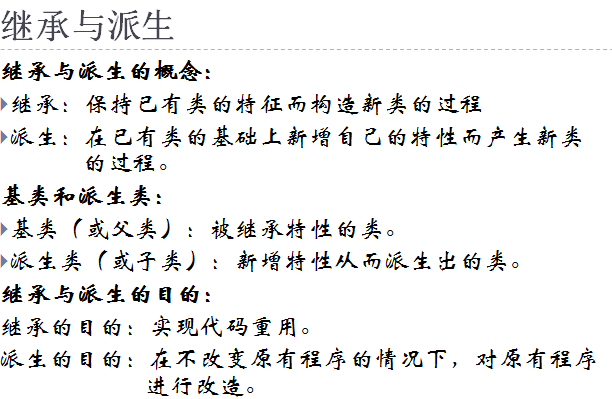
【继承就是继承已有特性，派生就是在继承的基础上加以修改，这样理解吧……】

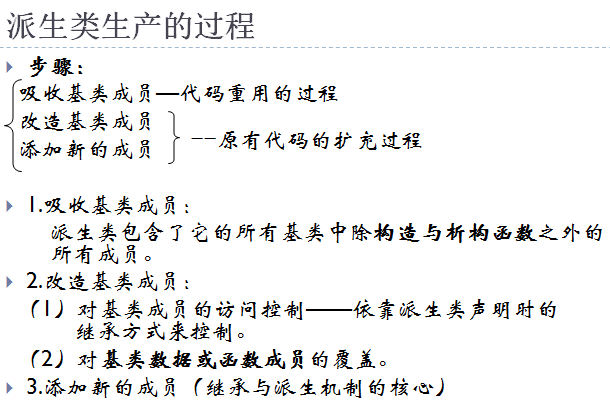
继承是从子类角度来讲的，是说子类从父类中继承得到某些属性…

派生是从父类角度来讲的，是说父类派生出子类



子类的对象同时也可以是父类的对象。





父类的公有函数继承到子类是否还是公有要看继承方式（权限约束）

父类的变量（如 int a；）如果子类有相同的名字（int a；），就会被子类覆盖掉，就是说子类里面只有一个int a；【还是可以调用父类的a，但是调用方式不一样.】

其实是有的，调用方式要用域来标志，比如A派生出B，B里面就有两个变量a，

用B：：a就是自己的，用A：：a就是父类的

派生类不能继承：

基类的析构函数、构造函数

基类的赋值运算符

基类的友元

派生类可以增加：

新的数据

新的成员函数

新的构造函数、析构函数

新的友元

C++访问控制-- **private, public, protected**

可以被以下访问

该类中的函数

其友元函数

PS。该类的对象也不能直接访问

可以被以下访问

该类中的函数

子类的函数

其友元函数

可以被以下访问

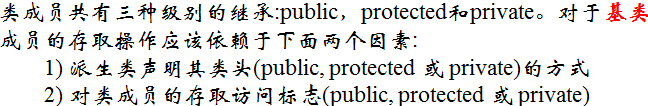
该类中的函数

子类的函数

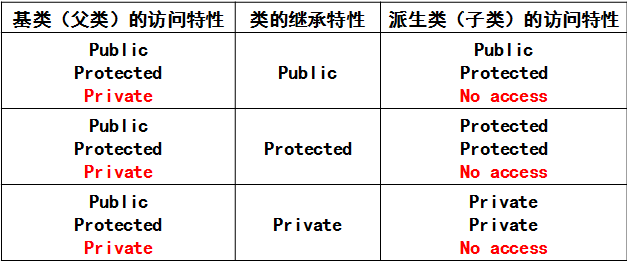
其友元函数

通过该类的对象访问

继承方式



你要注意的就是，子类有子类自己的东西可以加进去，这个跟父类是不冲突的，是子类自己定义的，而从父类继承下来的属性服从下面的表格。



基类的私有成员不被继承

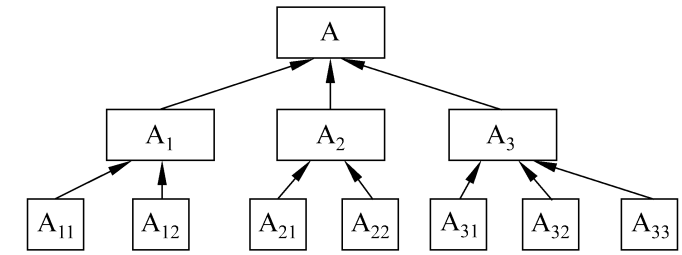
公有继承不改变属性

保护继承把基类的公有函数和保护函数改为保护

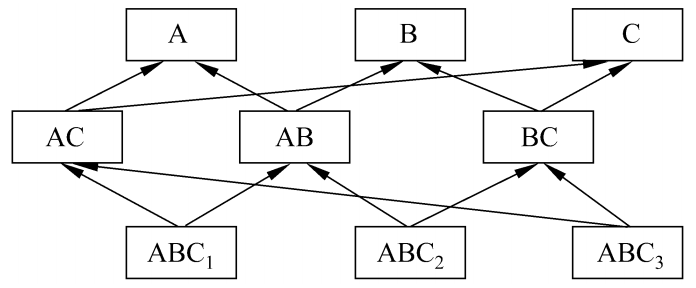
私有继承把基类的公有函数和保护函数改为私有

单继承和多继承

单继承：



多继承：



直接继承和间接继承

直接继承

Employee是HourlyWorker的直接继承

间接继承

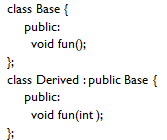
Employee是MinuteWorker的间接继承

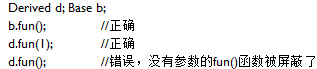
【就是类似祖孙是间接继承，父子是直接继承……】对…

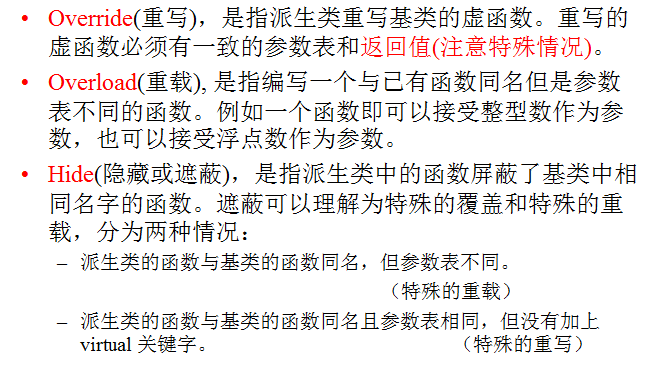
函数重写

overriding：只用函数名做标志

重写是替换原有函数，重载是允许存在参数表不同的同名函数

fun被重写了

 要访问Base中的fun()要通过



静态函数与非静态函数

【我理解的就是作用域的不同】

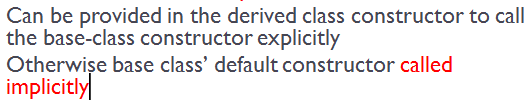
百度到的部分相关解释：

C++规定赋值运算符“=”只能重载为类的非静态成员函数，而不可以重载为类的友元函数。

不能重载为类的静态成员应该比较容易理解，因为静态成员函数是属于整个类的，不是属于某个对象的，它只能去操作类静态数据成员。而赋值运算符“=”是基于对象操作的。

派生类的构造函数和析构函数

基类的初始化可以使用初始化列表

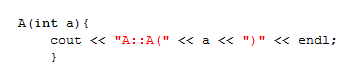


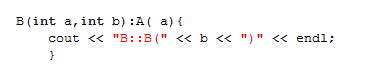
【这里好像是在说构造函数的显式调用和隐式调用。

是说如果派生类里构造函数显式地初始化了基类，否则会由隐式的缺省参数自动初始化基类？←←敲了一次简单继承的代码之后的理解。】

基类的构造函数和赋值操作符不被派生类继承，但派生类的构造函数和赋值操作符还可以调用它们

【就是说派生类的构造函数在初始化的时候依然可以调用基类的构造函数，例如：

A的构造函数：

B继承A后的构造函数：

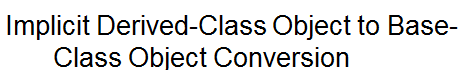
PS。如果B最后没有调用有参数的构造函数，A的有参数的构造函数也不会被调用，那初始化基类A的时候就会调用A的缺省构造函数……吧

】

对…这不就是你前面说的么…

父类一般用父类构造函数，不显式调用，默认构造函数也会被调用【理解同上】

调用顺序：父类的构造函数先调用 子类的析构函数先调用

隐式派生类到基类对象的转换

父类的私有不会被子类直接继承。

私有：继承但不直接访问

父类对象和子类对象的互相赋值

子类对象赋值给父类对象：



可以做到，因为父类对象中除私有外的数据成员都能在子类对象中找到对应

派生类的的新数据不会赋值给父类（数据会丢失）

基类的私有数据成员也不会被赋值

父类对象直接赋值给子类对象：



不一定可以，除非在子类重载赋值运算符

原因是那些派生类对象中新添加的数据成员在父类对象中找不到对应的数据成员

当子类对象访问新的数据成员时可能会出错（下标越界？？？）

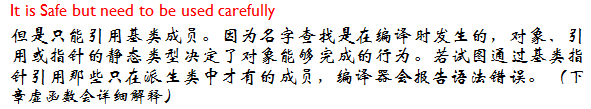
这跟下标越界什么关系…的确是有点类似，下标越界说的是访问未知的内存，这里也一样，子类对象中的那些在父类里面不存在的成员是未知的，访问的时候就可能出错…

四种混合着用基类和派生类对象和指针：

父类指针指向父类对象：可以

子类指针指向子类对象：可以

父类指针指向子类对象：



父类指针尝试去调用子类新的成员函数：目前不行（虚函数可以，多态问题）

子类的成员函数是新增加的

子类指针指向父类对象：不行，语法错误

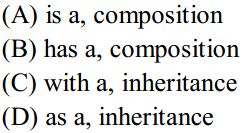
组合与继承

“is a”：继承，子类对象一定是父类对象

“has a”：包含关系，类有一个对象，这个对象里面有另外一个类作为数据成员

++++++++++++++++++++++++期末相关题目++++++++++++++++++++++++++++++

14. 



（09年第9题）

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

深拷贝和浅拷贝

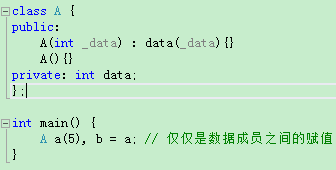
【我简单的理解是，浅拷贝其实是指向了同一个内存，深拷贝在拷贝的时候开辟了新的内存空间】

从内存空间来考虑问题…很好~

理解是对的，后面遇到具体题目的时候就知道为什么要深拷贝了…

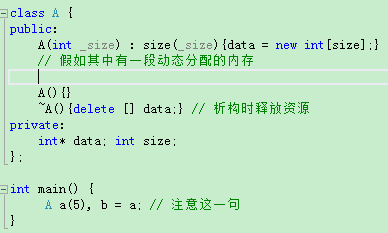
·浅拷贝就是对象的数据成员之间的简单赋值

如你设计了一个没有类而没有提供它的复制构造函数，当用该类的一个对象去给令一个对象赋值时所执行的过程就是浅拷贝，如：



这一句b = a;就是浅拷贝，执行完这句后b.data = 5;

当对象中有这些资源时，例子：



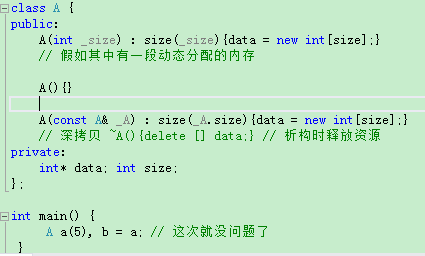
这里的b = a会造成未定义行为，因为类A中的复制构造函数是编译器生成的，所以b = a执行的是一个浅拷贝过程。我说过浅拷贝是对象数据之间的简单赋值，比如：

b.size = a.size; b.data = a.data; // Oops!

这里b的指针data和a的指针指向了堆上的同一块内存，a和b析构时，b先把其data指向的动态分配的内存释放了一次，而后a析构时又将这块已经被释放过的内存再释放一次。

对同一块动态内存执行2次以上释放的结果是未定义的，所以这将导致内存泄露或程序崩溃

·深拷贝指的就是当拷贝对象中有对其他资源（如堆、文件、系统等）的引用时（引用可以是指针或引用）时，对象的另开辟一块新的资源，而不再对拷贝对象中有对其他资源的引用的指针或引用进行单纯的赋值。如：



【深拷贝和浅拷贝的区别是在对象状态中包含其它对象的引用的时候，当拷贝一个对象时，如果需要拷贝这个对象引用的对象，则是深拷贝，否则是浅拷贝】