**普通继承和访问权限**

当一个继承没有虚拟继承或者是多重继承时，就是一个简单的继承的时候，这个时候就是一个普通的继承。

普通继承的内存空间是：子类的对象中，包含了父类的成员变量，同时也可以调用父类的成员函数（有些情况不可以，比如访问权限和重定义）

访问权限：比较继承方式和父类的成员修饰符 -- 二者取权限低的那个。这里的访问权限，指的是子类在类内部访问父类成员的权限和子类对象在类外访问的权限。

（总结：一般继承时，父类用的是public）

**问题：没有合适的构造函数**

我们在编程的时候，经常会遇到一种情况就是，没有合适的构造函数，这个时候的可能有以下几种原因：

构造函数的参数不匹配，比如我们定义的一个构造函数是没有参数的，然后使用类实例化一个对象的时候，传递了一个参数，这个时候没有合适的构造函数；同样的道理就是，我们定义的构造函数有参数，但是我们使用类实例化一个对象 的时候，却没有传递一个参数，这个时候也是没有合适的构造函数。这里我们引入一个概念，就是default构造函数，这个意思就是，我们在使用类实例化一个对象的时候，可以不用传参数，这种情况可以是，我们定义的构造函数是不带参数的，还有一种情况是，我们的构造函数带参，但是这个参数是一个缺省值。

第二种情况就是，我们的子类继承了父类，这个时候，子类在调用构造函数的时候，需要合成一个构造函数，就是先调用父类的构造函数，然后再调用自己的构造函数，把这两个 合成了一个构造函数，这个样子就是可能存在实例化一个对象的时候，给 父类传递的参数 不正确。

#include<iostream>

using namespace std;

class A

{

public:

A(int a)

:\_a(a)

{}

public:

int \_a;

};

class B : public A

{

public:

B(int a)

:\_b(0)

, A(a) //这一步非常重要，必须要显式的构造书一个无名的父类的对象

{

}

public:

int \_b;

};

int main()

{

A a(0);

B b(0); //这个0参数实际是父类需要的参数

a = b;

return 0;

}

上面我们应该注意的一个问题就是，我们需要在子类（B）里面显式的构造出一个无名的父类对象

**is-a和has-a**

public继承时一个接口继承，是is-a原则

一个子类就是一个父类，因为父类的所有内容都可用（is-a）

protected/private继承是实现继承，父类的成员有一部分子类是无法继承的

一个子类中有一个父类，但是部分父类的成员是不可用的（has-a）

总结：这里所说的is-a和has-a仅仅限于继承的方式上面，就是如果是 public继承则是is-a，如果是其他两种继承方式则是has-a。不用考虑父类里面的访问限定符。

关于is-a的设计：我们知道公有继承是is-a的关系，但是我们不能滥用is-a，is-a代表的一种模型是，父类的每个操作，子类都能够执行，但是我们必须注意的一个问题就是，父类的有些操作子类是执行不了了，我们在设计的时候，还需要考虑现实的生活，比如我们设计了父类：鸟。鸟里面有有一个函数是fly，这里有设计了一个子类，企鹅，采用的继承方式是公有继承，但是这里显然 存在了一个问题，就是企鹅是 不会 fly的，所以上面的设计是不合理的。这里我们还应该考虑的一个问题就是，我们应该如何设计呢，两种方式，一种是在鸟类中不设计fly，然后子类继承的时候采用公有继承，可以自己定义一个fly函数，还有一种设计是，我们在鸟类中定义了fly函数，然后子类继承的时候在进行重定义或者是重写，然后企鹅中告诉我们“企鹅不会飞”。比较上面的两种方式我们发现，第一种 方式更合理一些，第二种方式显得更加的冗余了。

再来看一个例子，我们一般的认为是，一个正方形就是一个长方形，但是在设计长方形的时候，我们可以设计一个函数，改变面积，此时函数 的功能是只增大长度而不增加宽度，但是 我们 的正方形继承之后，显然这个增加面积的函数对于正方形这个类是错误的，因为我们的正方形要是定义了函数之后，正方形就不在是正方形了。

综上：我们在使用公有继承的时候，想使用is-a，就要在设计父类的时候，检查父类的所有操作是否完全符合子类的要求。

**赋值兼容规则 -- 必须是公有继承(is-a)**

我们先来看一个图，看看普通的继承内存是如何分配的 image

所谓赋值兼容规则就是，子类是一个父类，这个时候子类是可以赋值给父类的，这种赋值 包括三种行为，一种是子类的额对象赋值给父类，一种是子类的指针赋值给父类，还有一种是子类的引用赋值给父类

上面的三种行为均发生了切片行为，或者叫做是切割行为。

反过来思考一个问题就是，我们的父类可以赋值给子类吗，原则上是不可以的，比如A是父类，B是子类，下面的方式不可以的

A a;

B b;

b = a; //错误

B& rb = a; //错误

B\* pb = &a; //错误

从内存上面我们很好的理解上面的结果，但是如果我们使用了强制类型转换就可以改变其中的一部分，比如下面的行为是正确的

A a;

B b;

B& rb = (B\*)a; //错误

B\* pb = (B&)&a; //错误

原因很简单，如果是 对象赋值的话，父类的内存空间要比子类小的，无法完成赋值，但是指针和引用不一样，指针和引用的大小是没有变得，指针是四个字节，只是把类型强制 转化一个就可以了，但是有人可能会费解的一个问题就是，为什么引用可以呢，引用不是指针呀，其实解释是这个样子的，引用的底层实现其实也是一个指针。

重定义 -- 把父类成员隐藏起来

什么是隐藏

子类和父类有相同名字的成员（成员变量或者是成员函数），这个访问 子类的成员的时候，就会自动的把父类的成员隐藏起来（父类是没有虚函数的）

如何理解和记忆

父类定义了一个函数或者是成员变量，然后在子类中定义了相同的成员函数或者是变量，这个时候就相当于把父类的成员 函数或者是成员变量隐藏起来了。

重定义是方法，隐藏是结果。

(同样的道理，我们也可以理解虚函数的重写，重写是方法，覆盖是结果)