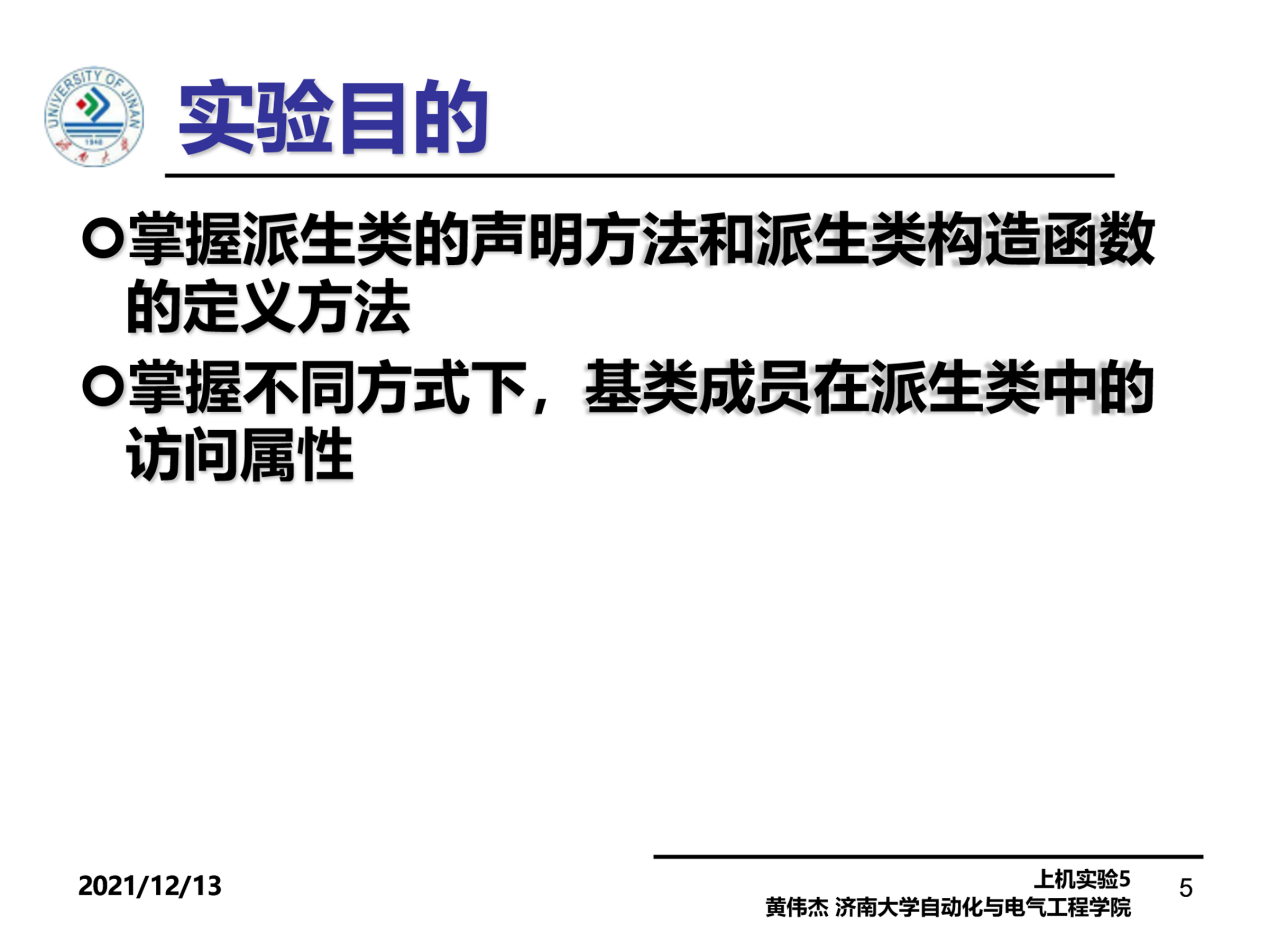
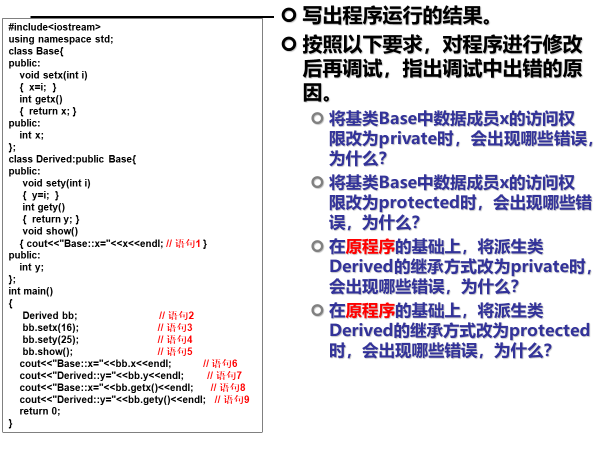
Copyright ©2021-2099 XiangleMeng. All rights reserved

**实验要求：**



****

**程序代码：**

#include <iostream>

using namespace std;

class Base {

public:

void setx(int i)

{

x = i;

}

int getx()

{

return x;

}

private:

int x;

};

class Derived :public Base {

public:

void sety(int i)

{

y = i;

}

int gety()

{

return y;

}

void show()

{

cout << "Base::x=" << x << endl;

}

public:

int y;

};

int main()

{

Derived bb;

bb.setx(16);

bb.sety(25);

bb.show();

cout << "Base::x=" << bb.x << endl;

cout << "Derived::y=" << bb.y << endl;

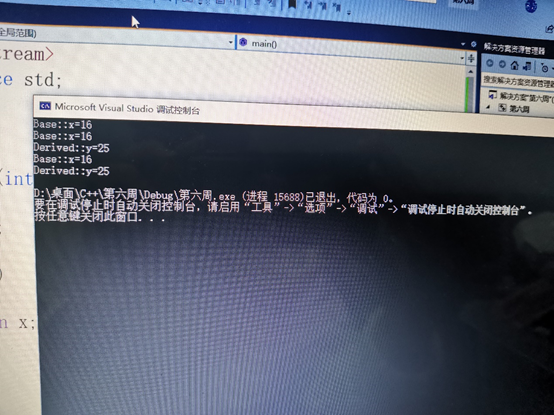
cout << "Base::x=" << bb.getx() << endl;

cout << "Derived::y=" << bb.gety() << endl;

return 0;

}

**程序运行结果如下：**

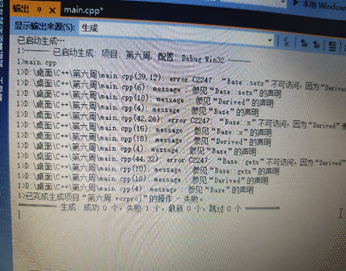


1. 出现错误为：bb.x无法访问，因为类的private成员无法在类外访问。

2. 出现错误为：bb.x无法访问，因为类的保护成员无法在类外访问。

3. 出现错误为：bb.x , bb.getx()无法访问，因为Derived的继承方式为private,类的private成员无法在类外访问。

4. 出现错误为：bb.x , bb.getx()无法访问，因为Derived的继承方式为protected,类的保护成员无法在类外访问。



实验总结：

1. 基类的构造函数和析构函数不能被继承，在派生类中，如果对派生类新增的成员进行初始化，就需要加入派生类的构造函数，同时，对所有从基类继承下来的成员的初始化工作，还是由基类的构造函数完成。
2. 通常，当创建派生类对象时，首先执行基类的构造函数，随后执行派生类的构造函数；当撤销派生类对象时，则先执行派生类的析构函数，随后再执行基类的析构函数。析构函数的调用顺序与构造函数的调用顺序正好相反。
3. 其中，基类构造函数的参数，通常来源于派生类构造函数的参数总表，也可以用常数值，因为这里是调用基类构造函数，所以这些参数是实参不是形参，它们可以是派生类构造函数总参数表中的参数，也可以是常量和全局变量。
4. 当基类的构造函数没有参数，或没有显式定义构造函数时，派生类可以不向基类传递参数，甚至可以不定义构造函数。
5. 派生类不能继承基类中的构造函数和析构函数。当基类含有带参数的构造函数时，派生类必须定义构造函数，以提供把参数传递给基类构造函数的途径。