**Copyright ©2021-2099 liangxiankui. All rights reserved**

**5.1**

#include<iostream>

using namespace std;

class Base{

public:

void setx(int i)

{ x=i; }

int getx()

{ return x; }

public:

int x;

};

class Derived:public Base{

public:

void sety(int i)

{ y=i; }

int gety()

{ return y; }

void show()

{ cout<<"Base::x="<<x<<endl; // 语句1 }

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;

}

程序结果如图

感想：先执行基类的构造函数，再执行对象成员的构造函数，最后执行派生类的构造函数。

处于同一层次的各个基类构造函数的执行顺序，取决于声明派生类时所指定的各个基类的顺序，与派生类构造函数中所定义的成员初始化列表的各项顺序无关。

