程序代码

#include<iostream>

using namespace std;

class Base {

public:

void setx(int i)

{

x = i;

}

int getx()

{

return x;

}

public:

int x;

};

class Derived :protected Base

{

public:

int y;

void sety(int i)

{

y = i;

}

int gety()

{

return y;

}

void show()

{

cout << "Base::x=" << x << endl; // 语句1

}

};

int main()

{

Derived bb; // 语句2

bb.setx(16); // 语句3

bb.sety(25); // 语句4

bb.show(); // 语句5

cout << "Base::x=" << bb.x << endl; // 语句6

cout << "Derived::y=" << bb.y << endl; // 语句7

cout << "Base::x=" << bb.getx() << endl; // 语句8

cout << "Derived::y=" << bb.gety() << endl; // 语句9

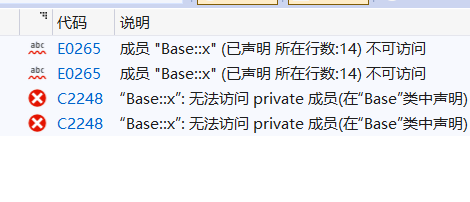
return 0;

}运行结果

**1.**

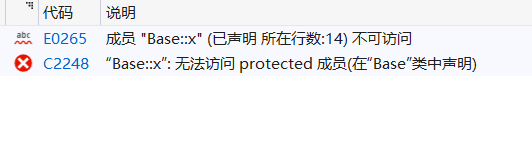


**2. 将基类Base中数据成员x的访问权限改为private时**



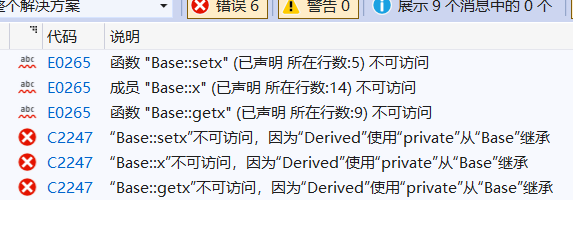
**原因：**将基类Base中数据成员x的访问权限改为private时，语句1和6错误，因为基类中的私有成员x不可被派生类Derived和派生类对象bb访问

**3. 将基类Base中数据成员x的访问权限改为protected时**



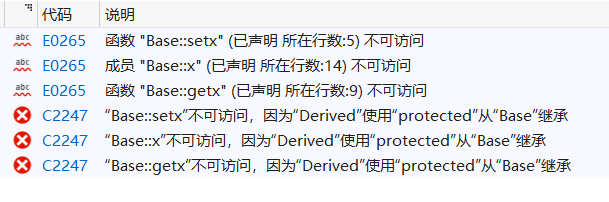
**原因：**将基类Base中数据成员x的访问权限改为protected时，语句6错误，因为基类的保护成员x不可被派生类对象bb访问，仅能被派生类Derived和基类自己的成员函数getx访问。

**4. 在原程序的基础上，将派生类Derived的继承方式改为private时**



**原因：**在原程序的基础上，将派生类Derived的继承方式改为private时，语句3、6、8错误，因为基类中的所有公有成员均变为私有访问属性，仅可内部访问，而对象bb则无法访问。

**5. 在原程序的基础上，将派生类Derived的继承方式改为protected时**



**原因：**在原程序的基础上，将派生类Derived的继承方式改为protected时，语句3、6、8错误，因为。基类中的所有公有成员均变为保护访问属性，仅可内部访问，而对象bb则无法访问。

感想心得

在本次实验中我掌握了派生类的定义方法的和派生类构造函数定义方法，掌握了在不同继承方式的情况下，基类成员在派生类中的访问权限，掌握了在多继承方式的情况下，构造函数与析构函数的调用时机与顺序。

了解到了构造函数的调用顺序：先调用所有基类的构造函数，再调用派生类中子对象类的构造函数（如果派生类中没有子对象），最后调用派生类的构造函数。

了解到了不同继承方式下基类成员在派生类中的访问权限：

当类的继承方式为公有继承时，在派生类中，基类的公有成员和保护成员被继承后分别作为派生类的公有成员和保护成员，这样使得派生类的成员函数可以直接访问它们，而派生类成员函数无法直接访问基类的私有成员。在类的外部，派生了的对象可以访问继承下来的基类公有成员。

当类的继承方式为私有继承时，在派生类中，基类的公有成员和保护成员作为派生类的私有成员，派生类的成员函数可以直接访问它们，而派生类的成员函数无法直接访问基类的私有成员。在类外部，派生类的对象无法访问基类的所有成员。

当类的继承方式为保护继承时，在派生类中，基类的公有成员和保护成员作为派生类的保护成员，派生类的成员函数可以直接访问它们，而派生类的成员函数无法直接访问基类的私有成员。在类外部，派生类的对象无法基类的所有成员。