第五次上机实验报告

电自2003 宿颢宝

实验一：

（1）原函数

代码：

#include<iostream>

#include<cmath>

#include<string>

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; // 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

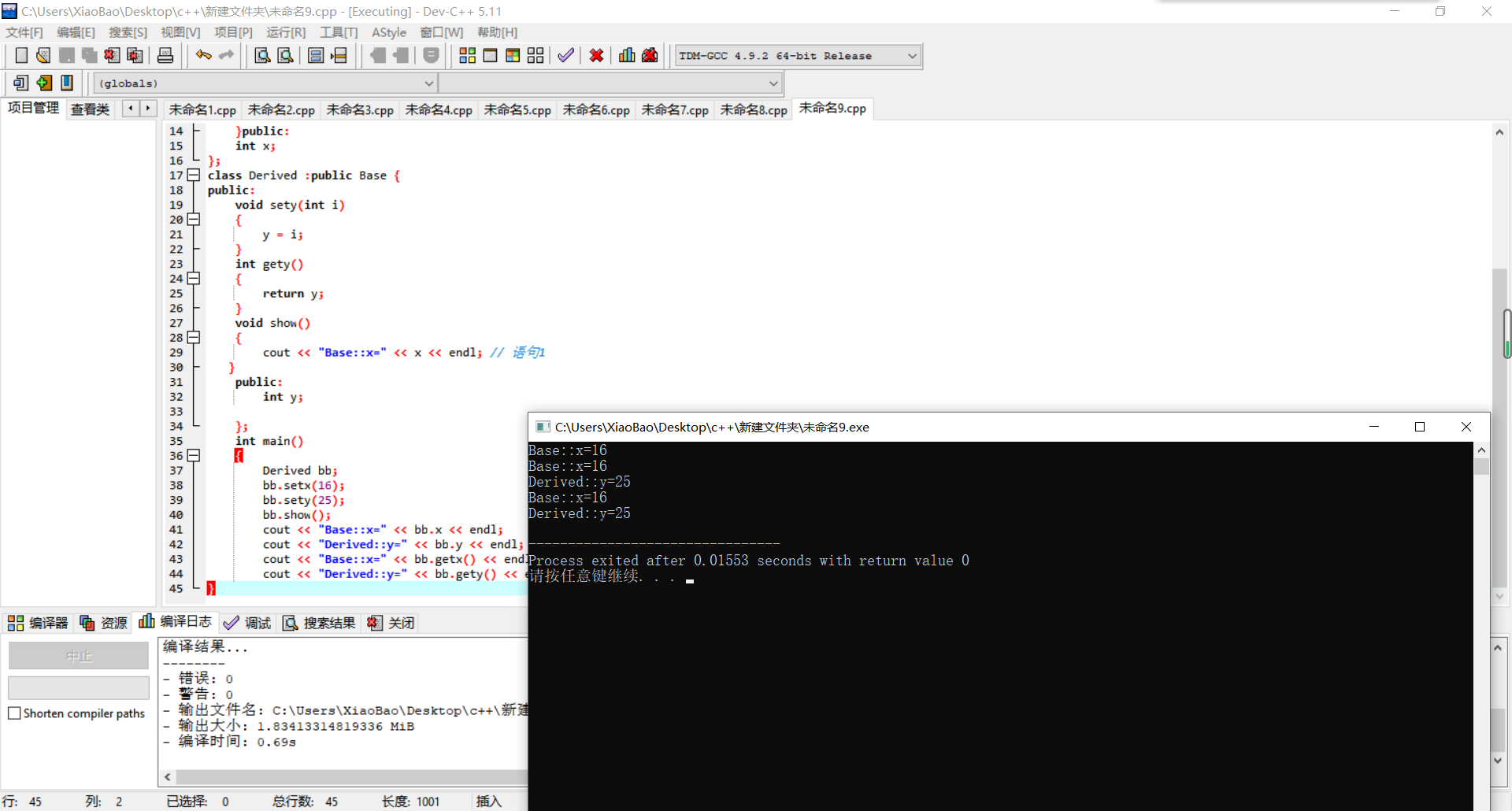
cout << "Derived::y=" << bb.y << endl; // 语句7

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

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

}

图片：



（2）当将Base中数据成员x的访问权限改为private时，将会出现编译错误的显示

这是因为，公有继承之后，对象只能对基类中的公有成员进行直接访问，但无法直接访问私有成员，甚至对象的成员函数也无法访问。对此，如果删除bb.show()函数的调用以及定义，删除cout << "Base::x=" << bb.x << endl;这一句，程序便能正常运行。

（3）当基类Base中的数据成员x的访问权限改为protected时，将会出现编译错误的显示，这是因为公有继承之后，虽然对象无法直接访问基类中的保护成员，但是与上一题不同，此时对象可以通过其成员函数来间接访问基类中的保护成员，因此这时无需删除bb.show()的调用以及其定义。

（4）当将派生类的继承方式改为private时，程序编译错误，这是因为派生类私有继承之后，其对象无法直接访问基类当中的任意成员，但可以通过成员函数对基类的公有成员进行间接访问，因此这时只需删除cout << "Base::x=" << bb.x << endl;这一句。

（5）当将派生类的继承方式改为protected时，程序编译错误，这是因为派生类保护继承之后，其对象无法直接访问基类当中的任意成员，但可以通过成员函数对基类的公有成员进行间接访问，因此这时只需删除cout << "Base::x=" << bb.x << endl;这一句。（这一点几乎与私有继承的特性一摸一样）

总结：

本次实验让我深刻地了解到三种继承方式（公有继承、私有继承、保护继承）的共性与不同之处：三种继承方式都无法访问基类的私有成员，（内部和对象两种方式都不行），但都能间接访问基类的公有成员以及保护成员；公有继承还能通过对象直接访问基类中的公有成员，这是其他两种方式无法实现的；私有继承和保护继承都无法通过对象直接访问基类的任意成员。