实验一：继承性和派生类（实验报告）

1. 程序清单、

#include <iostream>

#include <string>

using namespace std;

namespace golitter{

namespace lab3 {

class Building {

public:

Building() {}

Building(string address, signed short number):m\_address(address), m\_buildingNumber(number) {}

string get\_address();

signed short get\_buildingNumber();

void info();

private:

string m\_address;

signed short m\_buildingNumber;

};

// 获得Build地址

string Building::get\_address() {

return this->m\_address;

}

signed short Building::get\_buildingNumber() {

return this->m\_buildingNumber;

}

// 打印信息

void Building::info() {

cout<<this->m\_address<<" "<<this->m\_buildingNumber<<'\n';

}

class House: public Building {

public:

House() {}

House(string address, signed short buildNumber, double area, signed short houseNumber):

Building(address, buildNumber),

m\_area(area), m\_houseNumber(houseNumber) {}

void info();

private:

double m\_area;

signed short m\_houseNumber;

};

// 打印信息

void House::info() {

cout<<get\_address()<< " "<<get\_buildingNumber()<< " "

<<m\_houseNumber<< " "<<m\_area<<'\n';

}

class Office: public Building {

public:

Office() {}

Office(string address, signed short buildNumber, string officeName, string officePhoneNumber):

Building(address, buildNumber),

m\_officeName(officeName), m\_officePhoneNumber(officePhoneNumber) {}

void info();

private:

string m\_officeName;

string m\_officePhoneNumber;

};

// 打印信息

void Office::info() {

cout<<get\_address()<<" "<<get\_buildingNumber()<<" "

<<m\_officeName<<" "<<m\_officePhoneNumber<<'\n';

}

}}

using namespace golitter::lab3;

void solve() {

// Office o("add", 34,"dkjfkdfj", "2345234");

// o.info();

}

int main()

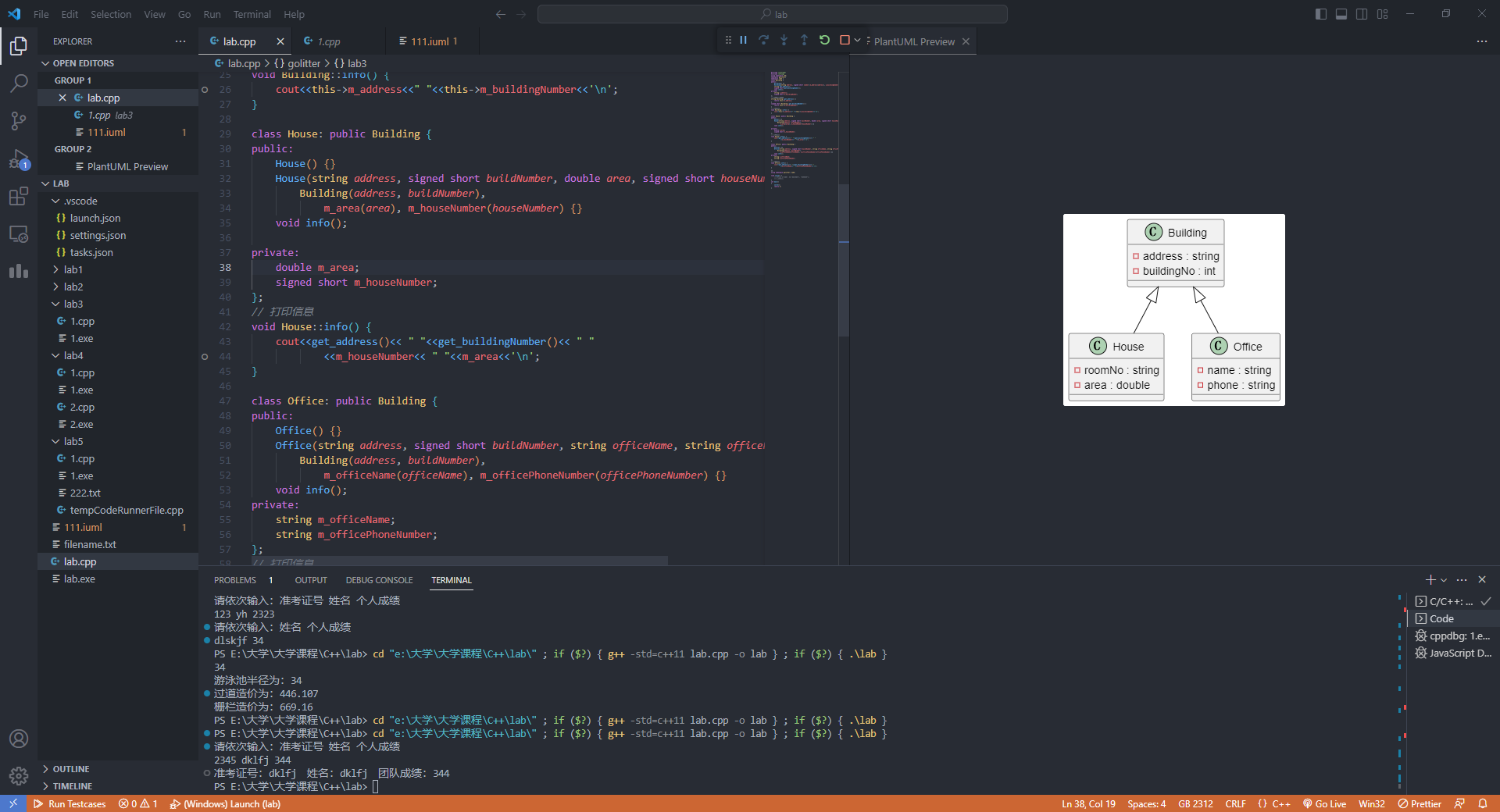
{

solve();

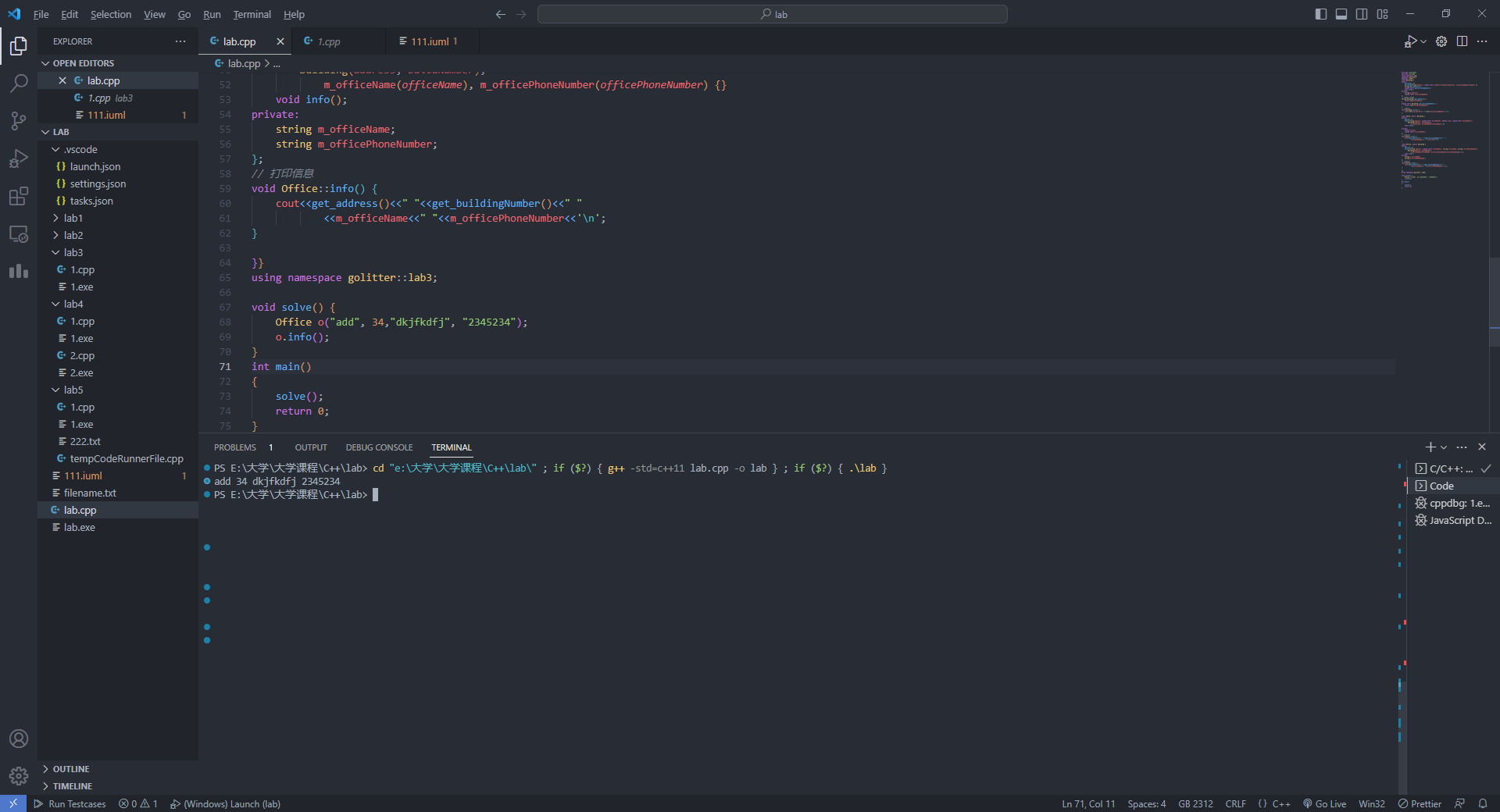
return 0;

}

1. 调试过程



1. 实验结果



1. 小结

通过本次实验，我学习到如何创建一个基类和派生类，并对它们进行操作。具体包括：

* 创建类：使用 class 关键字创建类，其中可以包含自己的成员变量和方法。
* 继承类：使用 class 派生类 : 基类 { /\*...\*/ } 的语法进行继承。在派生类中，可以访问基类的公有成员和保护成员。
* 重载方法：在派生类中定义与基类中同名的方法，从而覆盖（override）了基类中的方法。可以使用 override 关键字显式地指出该方法是对基类中方法的重载。
* 命名空间：使用 namespace 关键字定义命名空间，可以避免不同模块之间的命名冲突问题。