实验四：运算符重载和虚函数（实验报告）

1. 程序清单、

#include <iostream>

#include <cstring>

using namespace std;

namespace golitter {

namespace lab4 {

class MyString {

public:

MyString() {}

MyString(const char\* str) {

int len = strlen(str);

ptr\_str = new char [len+1];

strcpy(ptr\_str, str);

}

void info();

~MyString() {

delete [] ptr\_str;

}

MyString operator+(const MyString& rhs);

private:

char\* ptr\_str = nullptr;

};

MyString MyString::operator+(const MyString& rhs) {

char\* ptr\_tmp;

int len = strlen(this->ptr\_str) + strlen(rhs.ptr\_str) + 1;

ptr\_tmp = new char[len];

strcpy(ptr\_tmp, this->ptr\_str);

strcat(ptr\_str, rhs.ptr\_str);

return MyString(ptr\_str);

}

void MyString::info() {

cout<<this->ptr\_str<<'\n';

}

}}

using golitter::lab4::MyString;

void solve() {

MyString ms("dklfjkldjf");

MyString ms1("1234563245");

MyString c = ms + ms1;

c.info();

}

int main()

{

solve();

}

#include <iostream>

#define \_USE\_MATH\_DEFINES

#include <cmath>

using namespace std;

namespace golitter {

namespace lab4 {

class Circle {

public:

Circle() {}

Circle(double r):m\_radius(r) {}

virtual double area();

virtual double volume();

double get\_radius();

private:

double m\_radius;

};

// 求面积

double Circle::area() {

return m\_radius\*m\_radius\*M\_PI;

}

// 求体积

double Circle::volume() {

return 0;

}

// 半径

double Circle::get\_radius() {

return m\_radius;

}

class Sphere: public Circle {

public:

Sphere() {}

Sphere(double r): Circle(r) {}

virtual double area();

virtual double volume();

private:

};

// 求面积

double Sphere::area() {

return get\_radius()\*get\_radius()\*4\*M\_PI;

}

// 求体积

double Sphere::volume() {

return 4.0/3\*pow(get\_radius(),3)\*M\_PI;

}

class Column: public Circle {

public:

Column() {}

Column(double r, double h): Circle(r), m\_height(h) {}

virtual double area();

virtual double volume();

private:

double m\_height;

};

// 求面积

double Column::area() {

double ar = Circle::area()\*2;

ar += 2\*M\_PI\*get\_radius()\*m\_height;

return ar;

}

// 求体积

double Column::volume() {

return Circle::area()\*m\_height;

}

}}

using golitter::lab4::Column;

void solve() {

// Column c(3,4);

// cout<<c.area();

}

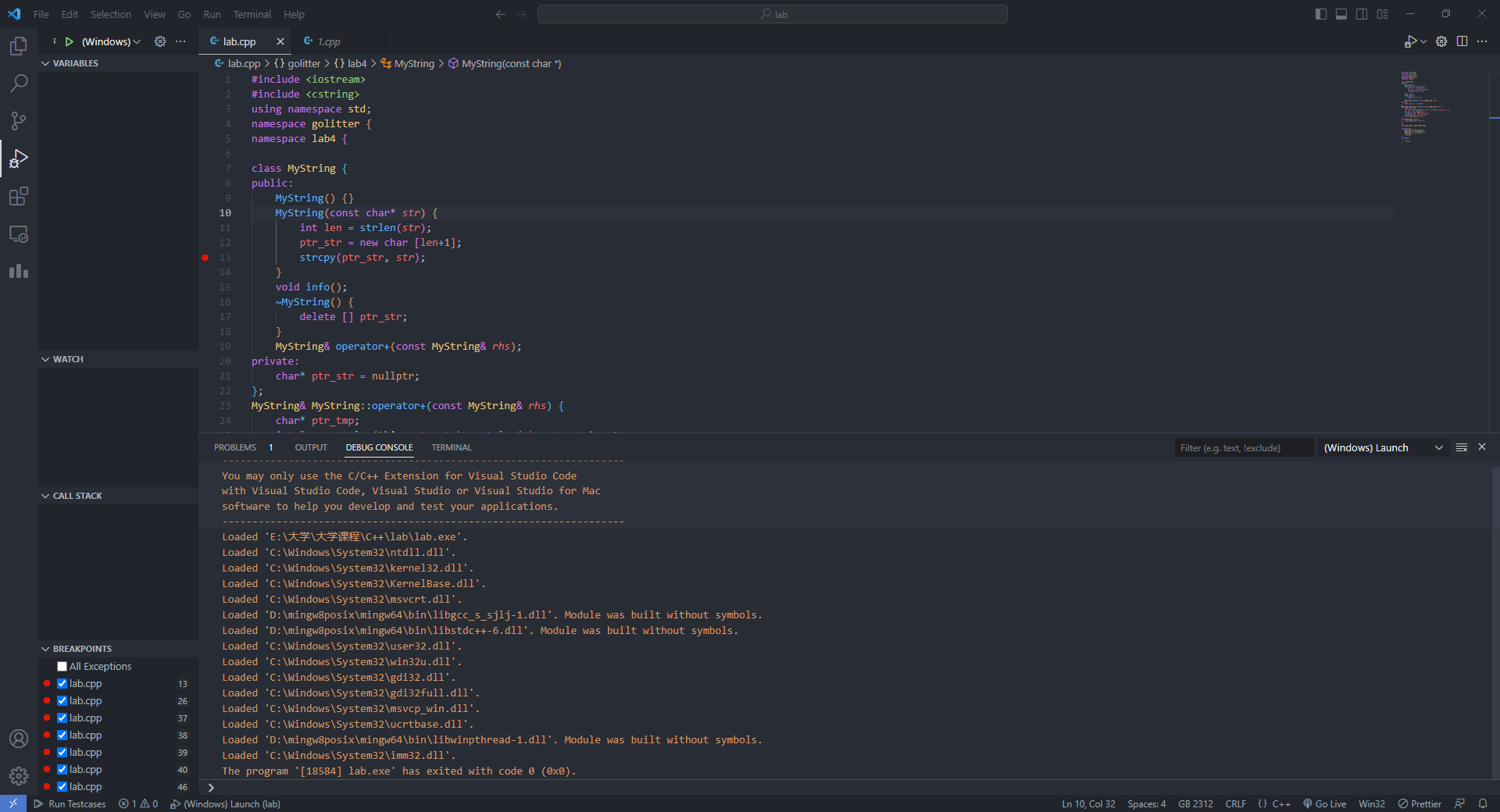
int main()

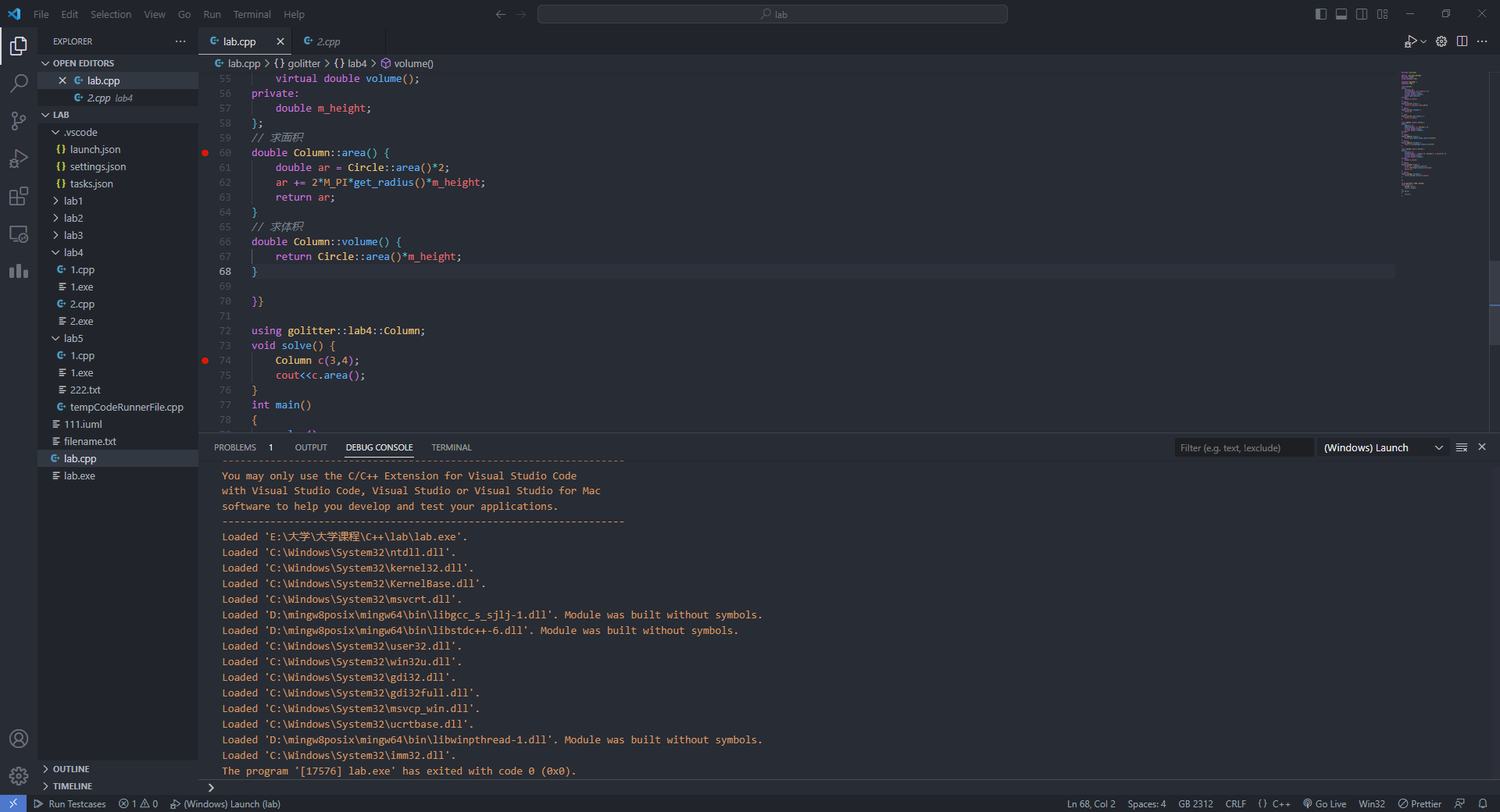
{

solve();

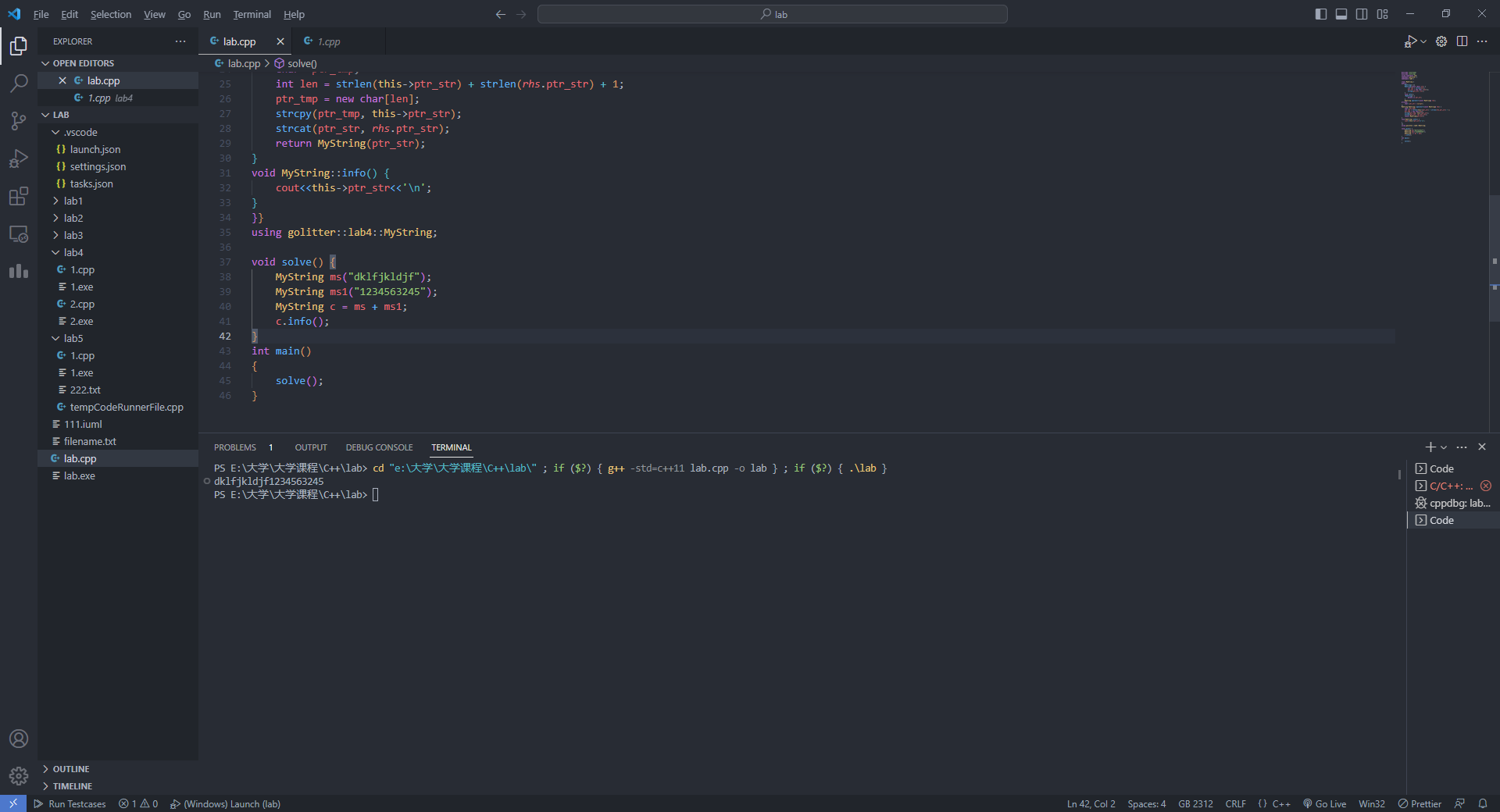
}

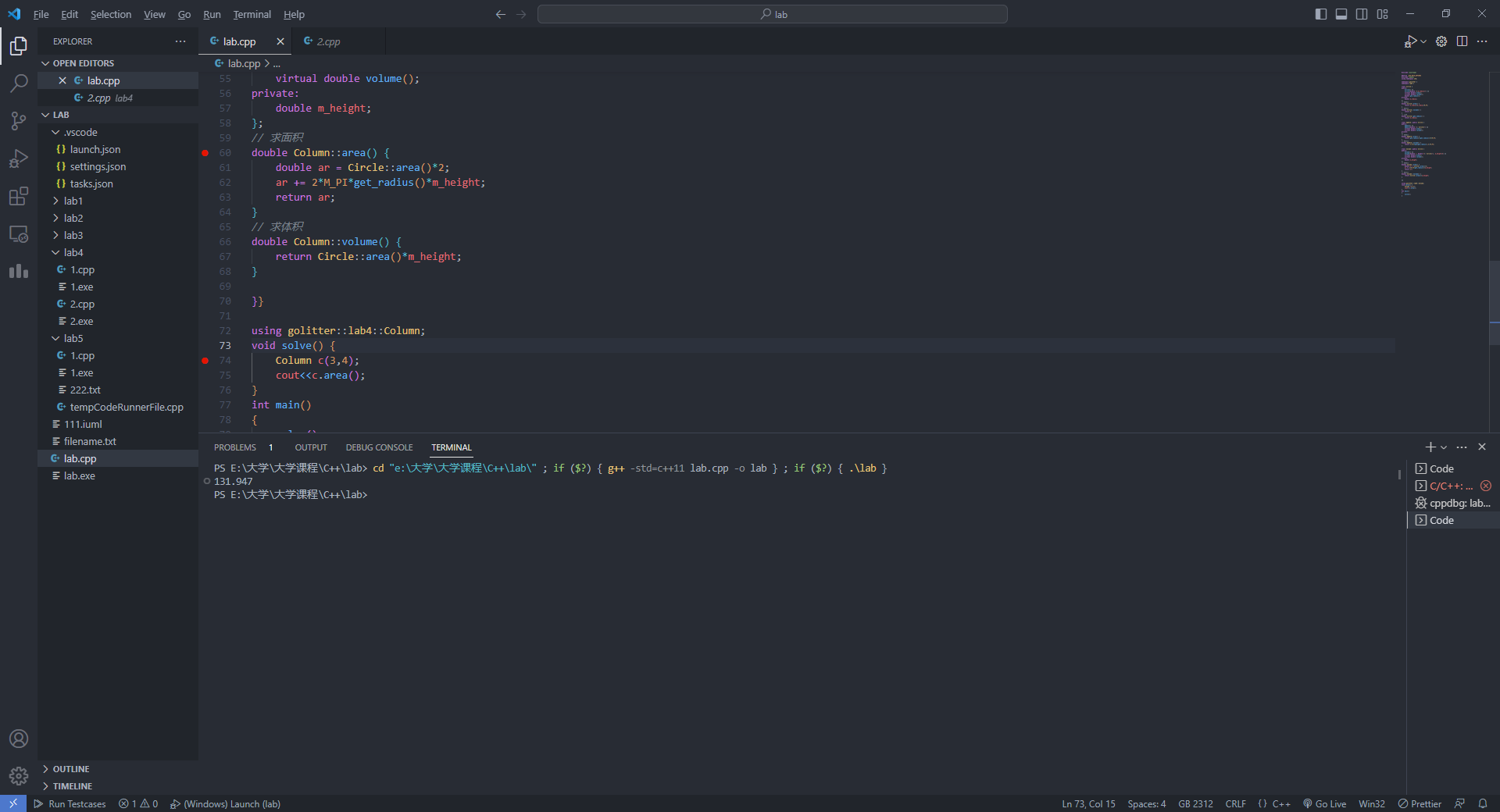
1. 调试过程





1. 实验结果





1. 小结

* 类的定义：通过定义一个类，把数据和函数组织在一起，方便进行面向对象编程。
* 构造函数和析构函数：构造函数在定义对象时被自动调用，并对对象进行实例化；析构函数在对象销毁时自动调用，负责释放对象占用的内存空间。
* 运算符重载：通过重载运算符，可以让自定义类型像内置类型一样使用常见运算符。
* 指针和动态内存分配：使用指针可以直接操作内存，动态内存分配则允许程序在运行时根据需要分配和释放内存空间。此处使用了new[]和delete[]进行动态内存分配和释放。
* 字符串操作函数：通过使用头文件<cstring>中的字符串操作函数，如strlen()、strcpy()、strcat()等，可以方便地对C风格字符串进行处理。
* 继承：通过定义一个派生类，在其基础上增加新的特性或修改已有特性，方便进行代码重用和拓展。