实验二：类与对象定义初始化（实验报告）

1. 程序清单、

#include <iostream>

using namespace std;

namespace golitter {

namespace lab2 {

class CostOfAround {

public:

CostOfAround() {}

void calculate\_aislePrice();

void calculate\_fencePrice();

void info();

void input\_r();

const double PI = 3.1415962;

private:

double m\_r;

double aislePrice;

double fencePrice;

void read\_r(double r);

};

static double PI = 3.1415962;

void CostOfAround::read\_r(double r) {

m\_r = r;

}

// 读入游泳池半径

void CostOfAround::input\_r() {

double tmp; cin>>tmp;

read\_r(tmp);

}

// 计算过道的造价

void CostOfAround::calculate\_aislePrice() {

aislePrice = m\_r\*2\*PI + (m\_r + 3)\*2\*PI;

}

// 计算栅栏的造价

void CostOfAround::calculate\_fencePrice() {

fencePrice = PI\*( (m\_r + 3) \* (m\_r + 3) - (m\_r)\*(m\_r));

}

// 打印信息

void CostOfAround::info() {

input\_r();

calculate\_aislePrice();

calculate\_fencePrice();

cout<<"游泳池半径为："<<m\_r<<'\n';

cout<<"过道造价为："<<aislePrice<<'\n';

cout<<"栅栏造价为："<<fencePrice<<'\n';

}

}}

using golitter::lab2::CostOfAround;

void solve() {

CostOfAround a;

a.info();

}

int main()

{

solve();

return 0;

}}

#include <iostream>

#include <string>

using namespace std;

namespace golitter {

namespace lab2 {

class Student {

friend istream& operator>>(istream& in, Student &rhs);

public:

Student():m\_hadSignReg(false) {}

void print\_prompt\_message();

void set\_all\_info();

void set\_score(double score);

void set\_name(string& name);

void print\_all\_info();

private:

string m\_name;

double m\_score;

bool m\_hadSignReg;

string m\_examReg\_number; // 准考证号 examRegistrationNumber

};

istream& operator>>(istream& in, Student &rhs) {

if(rhs.m\_hadSignReg != true) {

in>>rhs.m\_examReg\_number;

rhs.m\_hadSignReg = true;

}

in>>rhs.m\_name>>rhs.m\_score;

return in;

}

// 打印输入提示信息

void Student::print\_prompt\_message() {

cout<<"请依次输入：";

if(this->m\_hadSignReg != true) {

cout<<"准考证号 ";

// this->hadSignReg = true;

}

cout<<"姓名 个人成绩\n";

}

// 为准考证、姓名、成绩 赋值

void Student::set\_all\_info() {

print\_prompt\_message();

cin>>(\*this);

}

// 为成绩赋值

void Student::set\_score(double score) {

this->m\_score = score;

}

// 为名字赋值

void Student::set\_name(string& name) {

this->m\_name = name;

}

// 打印所有信息

void Student::print\_all\_info() {

cout<<"准考证号："<<this->m\_name<<" "<<"姓名："<<this->m\_name<<" 团队成绩："<<this->m\_score<<'\n';

}

}}

using golitter::lab2::Student;

void solve() {

// Student a; a.set\_all\_info();

// a.set\_all\_info();

}

int main()

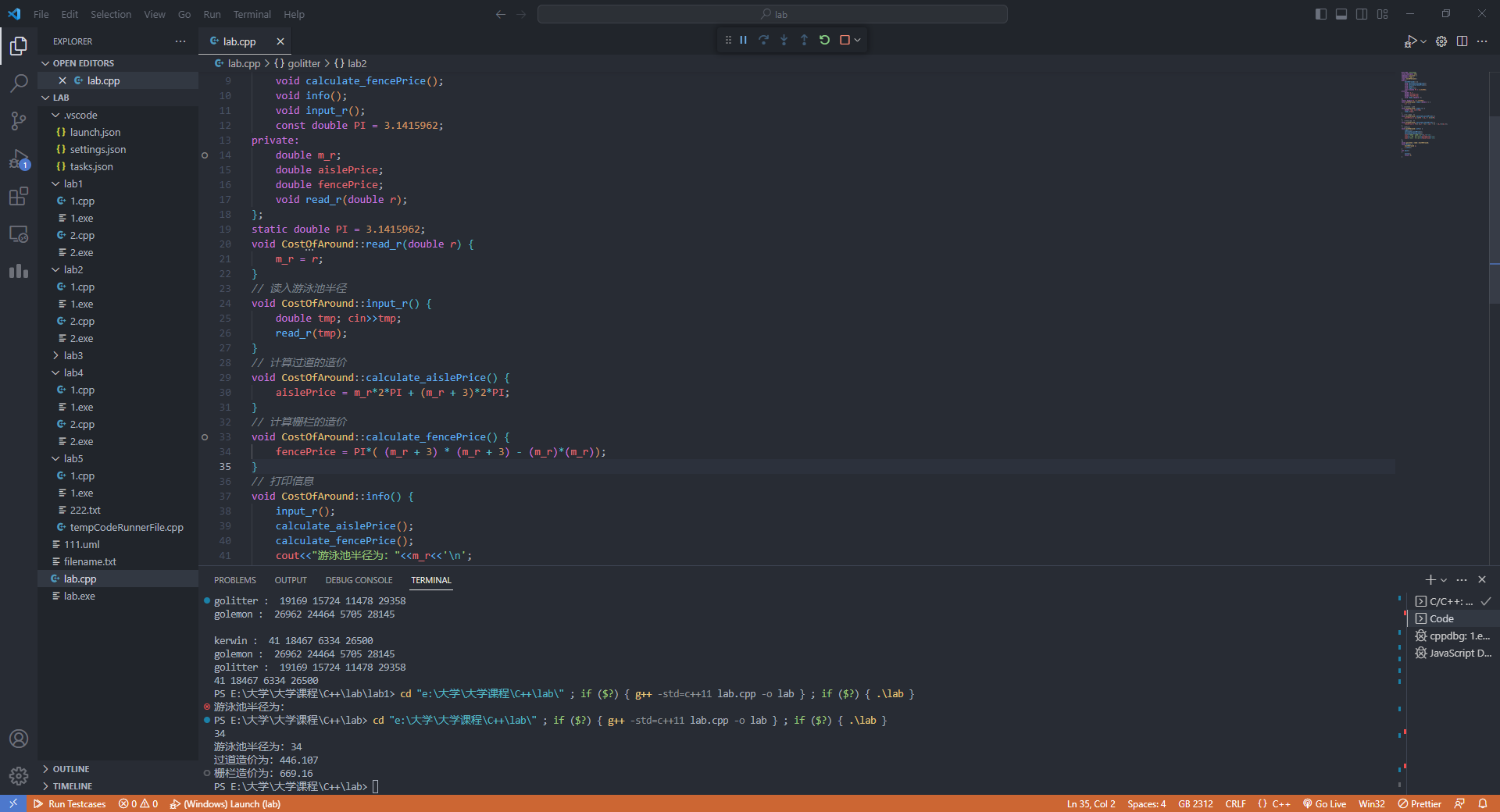
{

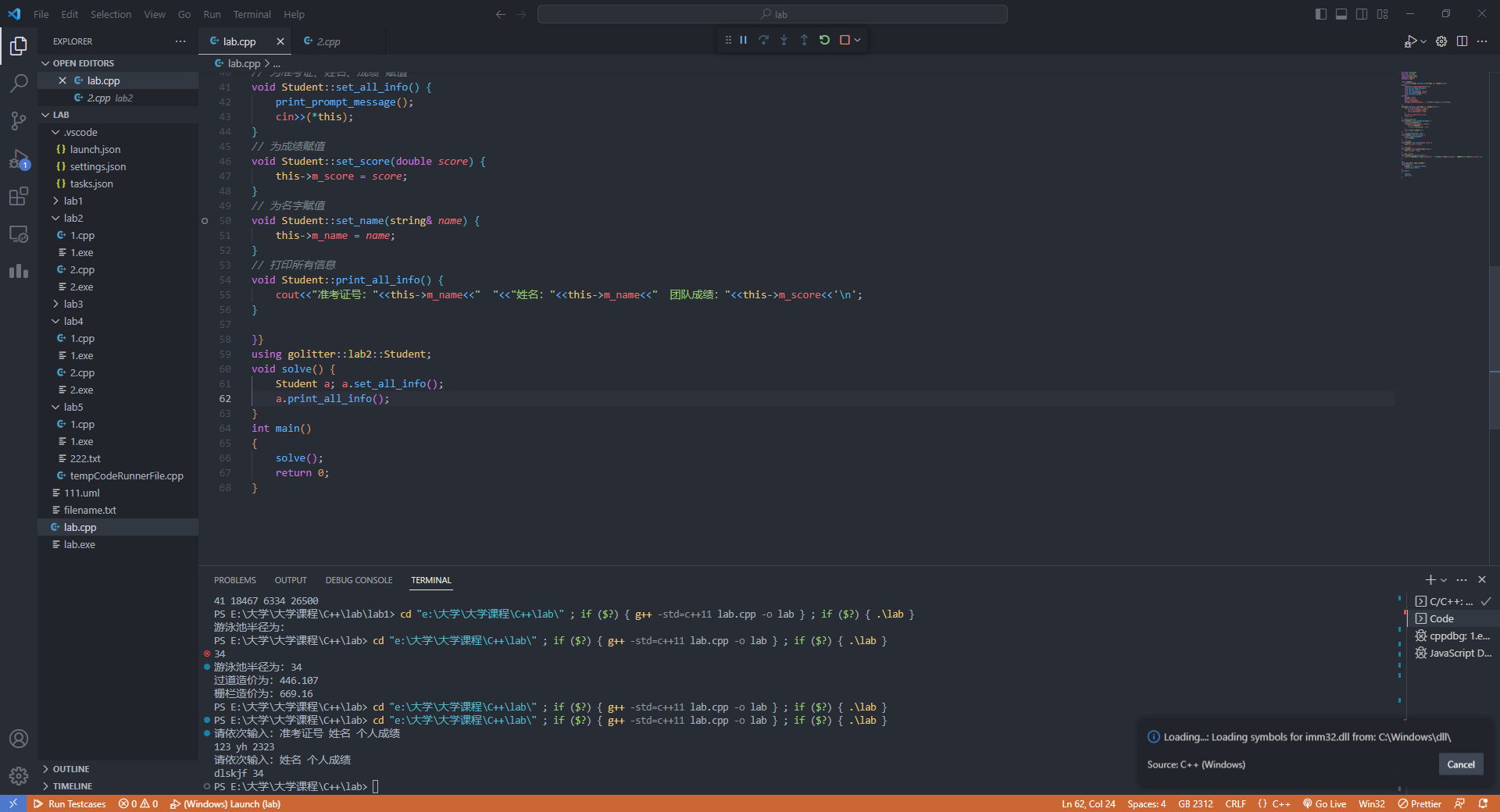
solve();

return 0;

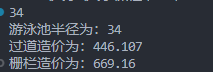
}

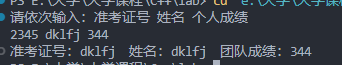
1. 调试过程





1. 实验结果





1. 小结

命名空间是 C++ 中用于避免命名冲突的一个重要机制，可以将一些相关的符号（如变量、函数、类等）放入同一个命名空间中，减少命名冲突的概率。这里的 golitter::lab2 就是一个命名空间。

类是具有相似属性和行为的对象的抽象。类中的成员函数用于定义类的行为，而成员变量则用于保存类的状态。这里的 CostOfAround 和 Student 就是两个具有不同属性和行为的类。

友元函数是可以访问类中私有成员的非成员函数。在本代码中，我使用了一个友元函数 istream& operator>>(istream& in, Student &rhs) 来实现对 Student 对象进行输入操作。

流输入输出则是 C++ 中常用的对数据进行输入输出的方式，包括输入流和输出流。本代码使用了 cin 对数据进行输入，并使用 cout 对数据进行输出。