实验五：文件操作（实验报告）

1. 程序清单、

#include <iostream>

#include <fstream>

#include <string>

#include <cstring>

using namespace std;

namespace golitter {

namespace lab5 {

class FileObject {

public:

    FileObject(const char\* *filename*) {

        fileptr = new char[strlen(*filename*) + 1];

        strcpy(fileptr, *filename*);

        open();

    }

    void open() {

        fo.open(fileptr, ios::out);

    }

    void write(const char\* *content*) {

        fo<<*content*<<endl;

    }

    void append(const char\* *content*) {

        close();

        fo.open(fileptr, ios::out | ios::app);

        write(*content*);

    }

    void close() {

        if(!fo.good()) return ;

        fo.close();

    }

    ~FileObject() {fo.close(); delete [] fileptr; }

private:

    ofstream fo;

    char \* fileptr;

    FileObject() {}

};

}}

using golitter::lab5::FileObject;

void solve() {

    FileObject a("222.txt");

    a.write("dskljfkldjf");

    a.append("dkjfdlfj");

}

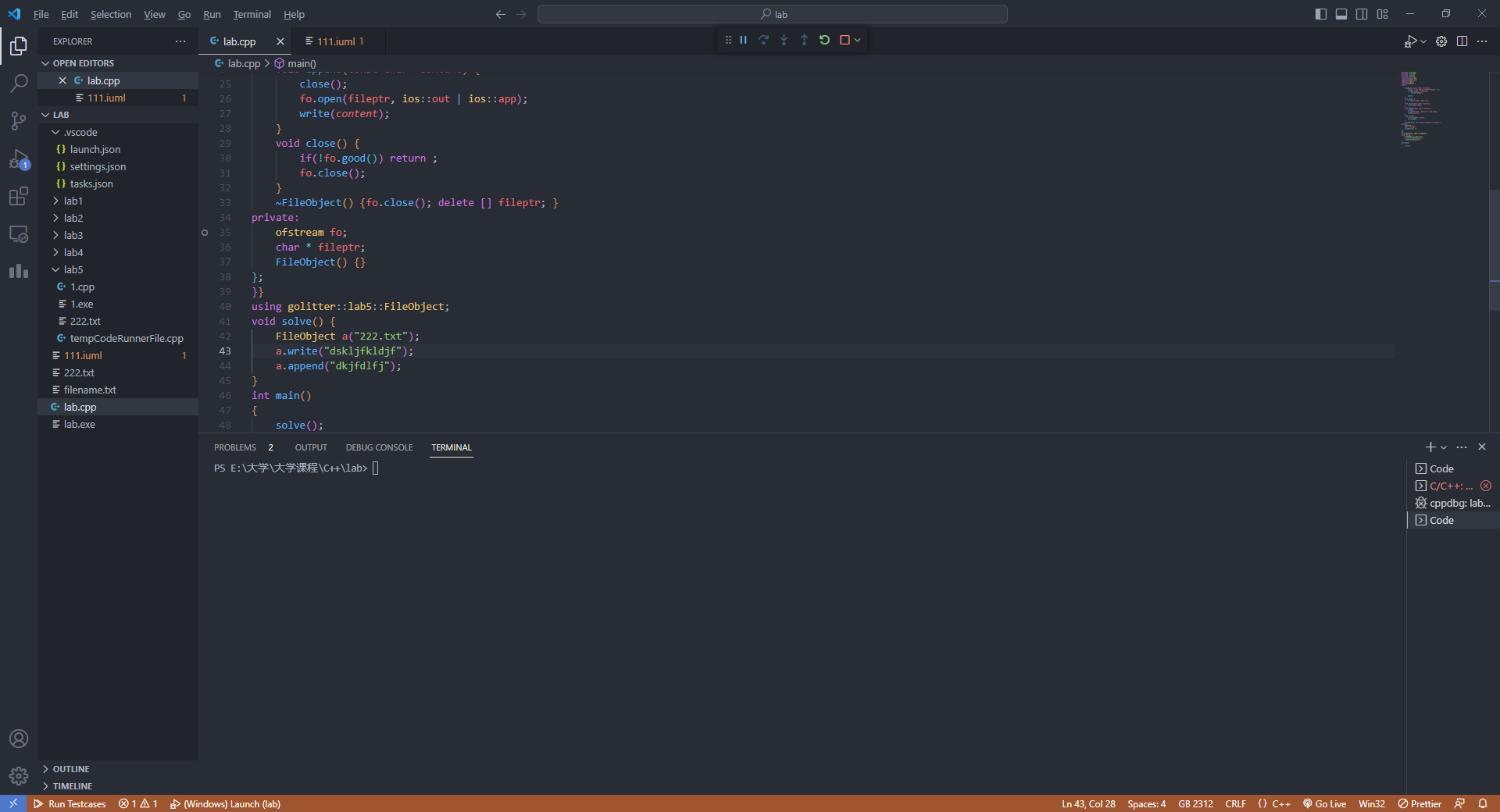
int main()

{

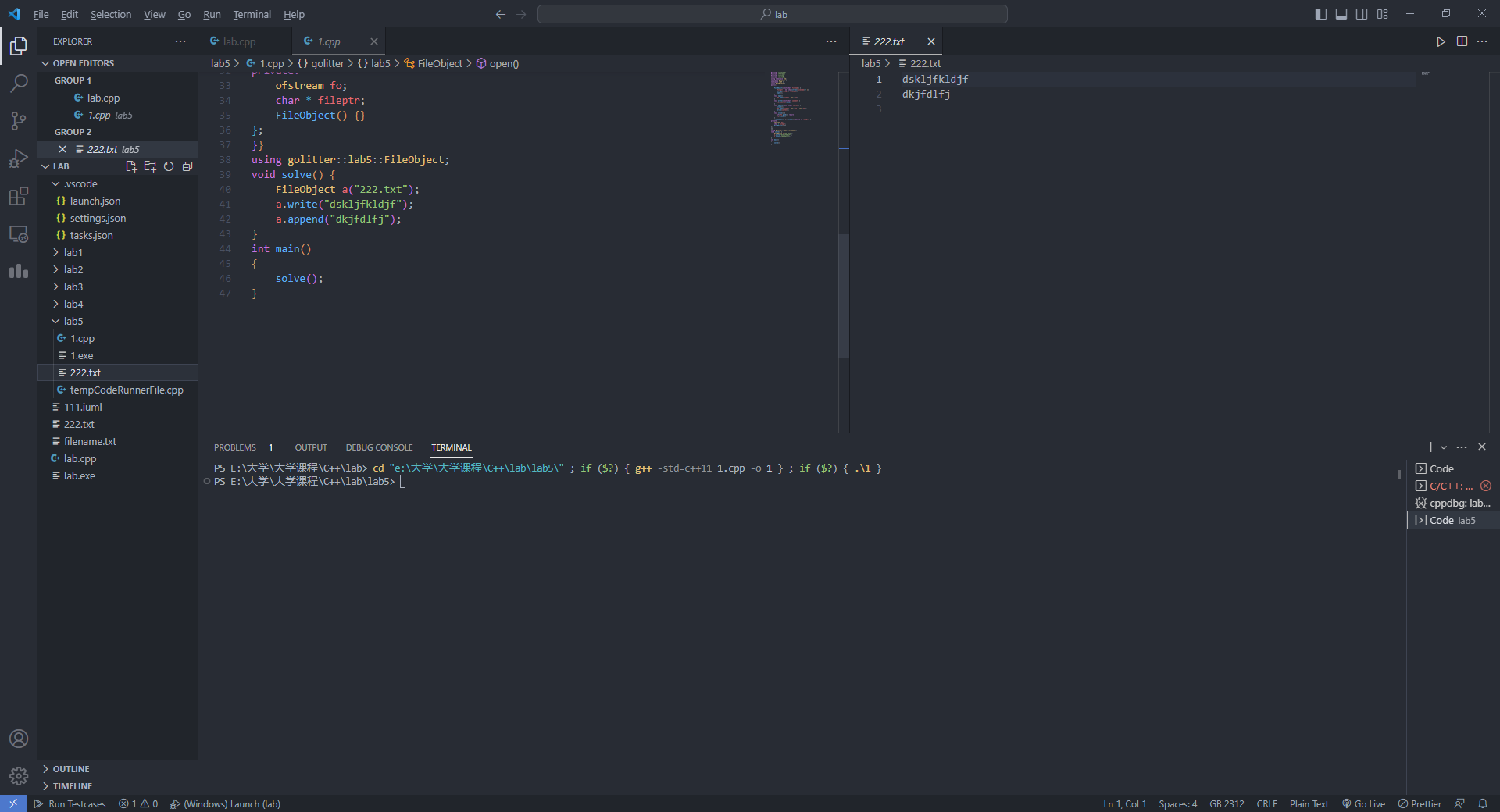
    solve();

}

1. 调试过程



1. 实验结果



1. 小结

* 文件流对象的创建：利用std::ofstream类来创建文件输出流对象，需要指定文件名和打开方式。
* 文件打开模式的选择：通过指定不同的打开方式，可以选择覆盖原有文件、追加内容或者读取文件等不同的操作模式。
* 文件的写入和追加：使用<<运算符将内容写入到文件输出流即可完成文件的写入操作，利用ios::out | ios::app打开方式可以实现文件的追加功能。
* 文件的关闭：在文件操作完成后，通过调用std::ofstream对象的close()函数来关闭文件输出流对象。
* 动态内存分配的使用：在构造函数中，通过动态分配内存来保存文件名，并在析构函数中释放内存空间，从而避免内存泄漏。