实验一：简单程序设计（实验报告）

1. 程序清单、

#include <iostream>

namespace golitter {

namespace lab1 {

class primPrinter {

public:

primPrinter(): m\_n(100) { sum = 0; }

primPrinter(int n): m\_n(n) { sum = 0; }

void info();

~primPrinter() {

clear\_Allocate\_Space();

}

private:

int m\_n; // 判断从1到n的质数;

bool \*ptr\_size = nullptr; // 开大小为n的bool类型数组指针

int \*ptr\_prime = nullptr; // 纯质数

int sum;

bool allocate\_Space\_For\_boolArray();

void clear\_Allocate\_Space();

void linear\_Screen();

};

// 打印信息

void primPrinter::info() {

linear\_Screen();

std::cout<<"质数个数： "<<sum<<'\n';

int cnt(0);

for(int i = 0; i < sum; ++i) {

cnt++;

if(cnt %6 == 0) std::cout<<'\n';

std::cout<<ptr\_prime[i]<<" ";

}

}

// 清除

void primPrinter::clear\_Allocate\_Space() {

if(this->ptr\_size != nullptr) {

delete [] this->ptr\_size;

delete [] this->ptr\_prime;

}

}

// 分配大小为n个的数组

bool primPrinter::allocate\_Space\_For\_boolArray() {

if(this->ptr\_size != nullptr) {

clear\_Allocate\_Space();

}

this->ptr\_size = new bool[this->m\_n + 1];

this->ptr\_prime = new int [this->m\_n + 1];

for(int i = 0; i <= this->m\_n; ++i) {

ptr\_size[i] = false;

}

return true;

}

// 线性筛求质数

void primPrinter::linear\_Screen() {

if(!allocate\_Space\_For\_boolArray()) { // err

return ;

}

int n = m\_n;

for(int i = 2; i <= n; ++i) {

if(!ptr\_size[i]) ptr\_prime[sum++] = i;

for(int j = 0; ptr\_prime[j] <= n/i; ++j) {

ptr\_size[ ptr\_prime[j] \* i] = true;

if(i % ptr\_prime[j] == 0) break;

}

}

}

}}

using golitter::lab1::primPrinter;

int main()

{

primPrinter a(34);

a.info();

std::cout<<"hello world"<<std::endl;

}

#include <iostream>

#include <string>

#include <vector>

#include <unordered\_map>

namespace golitter {

namespace lab1 {

using std::cout;

class scoreProcessor {

public:

scoreProcessor() {}

scoreProcessor(int M, int N):m\_M(M), m\_N(N) {

allocationSpace();

// test\_inputOtherInfo();

inputOtherInfo();

}

void sort\_columns(int col);

void info();

void input\_M\_and\_N(int M, int N);

void inputOtherInfo();

void findName(std::string name);

~scoreProcessor() {

(\*d2\_score).clear();

delete d2\_score;

delete [] ptr\_nameMap;

}

private:

int m\_M;

int m\_N;

std::string \*ptr\_nameMap = nullptr;

std::vector< std::vector<double> > \*d2\_score = nullptr; // 行学生，列分数

void allocationSpace();

void quick\_sort(int l, int r, int col);

void test\_inputOtherInfo();

};

// 分配二维数组空间

void scoreProcessor::allocationSpace() {

d2\_score = new std::vector< std::vector<double> >(m\_M);

ptr\_nameMap = new std::string[m\_M];

}

// 读入M和N

void scoreProcessor::input\_M\_and\_N(int M, int N) {

m\_M = M;

m\_N = N;

}

// 测试

void scoreProcessor::test\_inputOtherInfo() {

ptr\_nameMap[0] = "kerwin";

ptr\_nameMap[1] = "golitter";

ptr\_nameMap[2] = "golemon";

for(int i = 0; i < 3; ++i) {

for(int j = 0; j < 4; ++j) {

(\*d2\_score)[i].push\_back(rand());

}

}

}

// 读入二维分数表

void scoreProcessor::inputOtherInfo() {

for(int i = 0; i < m\_M; ++i) {

std::cin>>ptr\_nameMap[i]; // 建立序号与名字的映射

double sum(0), inp;

for(int j = 0; j < m\_N; ++j) {

std::cin>>inp;

(\*d2\_score)[2].push\_back(inp);

sum += inp;

}

(\*d2\_score)[2].push\_back(sum);

}

}

// 快排

void scoreProcessor::quick\_sort(int l, int r, int col) {

if(l >= r) return ;

int i = l - 1, j = r + 1;

double x = (\*d2\_score)[l + r >> 1][col];

while(i < j) {

do i++; while((\*d2\_score)[i][col] < x);

do j--; while((\*d2\_score)[j][col] > x);

if(i < j) swap(ptr\_nameMap[i], ptr\_nameMap[j]), swap( (\*d2\_score)[i], (\*d2\_score)[j]);

}

quick\_sort(l, j, col);

quick\_sort(j+1, r, col);

}

/// @brief 按列属性对学生分数进行排序

/// @param col

void scoreProcessor::sort\_columns(int col) {

quick\_sort(0, m\_M - 1, col);

}

// 打印信息

void scoreProcessor::info() {

for(int i = 0; i < m\_M; ++i) {

std::cout<<ptr\_nameMap[i]<<" : ";

for(int j = 0; j < m\_N; ++j) {

std::cout<<(\*d2\_score)[i][j]<<" ";

}

std::cout<<'\n';

}

}

// 查找名字 & 打印分数

void scoreProcessor::findName(std::string name) {

int order(0);

for(order; order < m\_M; ++order) {

if(ptr\_nameMap[order] == name) {

break;

}

}

if(order == m\_M) {

std::cout<<"查无此人"<<std::endl;

return ;

}

for(int i = 0; i < m\_N; ++i) {

std::cout<<(\*d2\_score)[order][i]<<" ";

}

puts("");

}

}}

using golitter::lab1::scoreProcessor;

void solve() {

scoreProcessor s(3,4);

// s.info();

// s.sort\_columns(3);

// puts("");

// s.info();

// s.findName("kerwin");

}

int main()

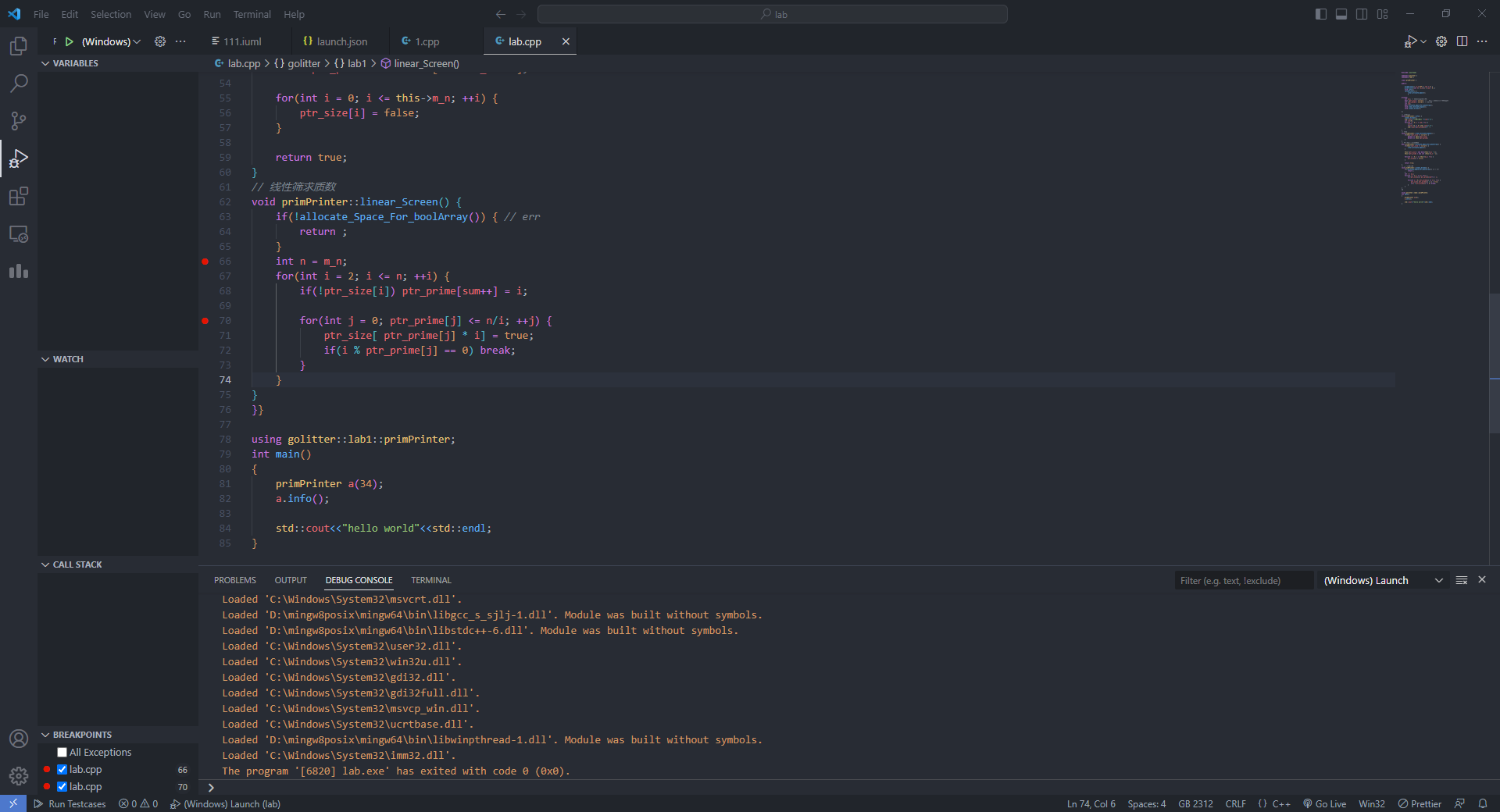
{

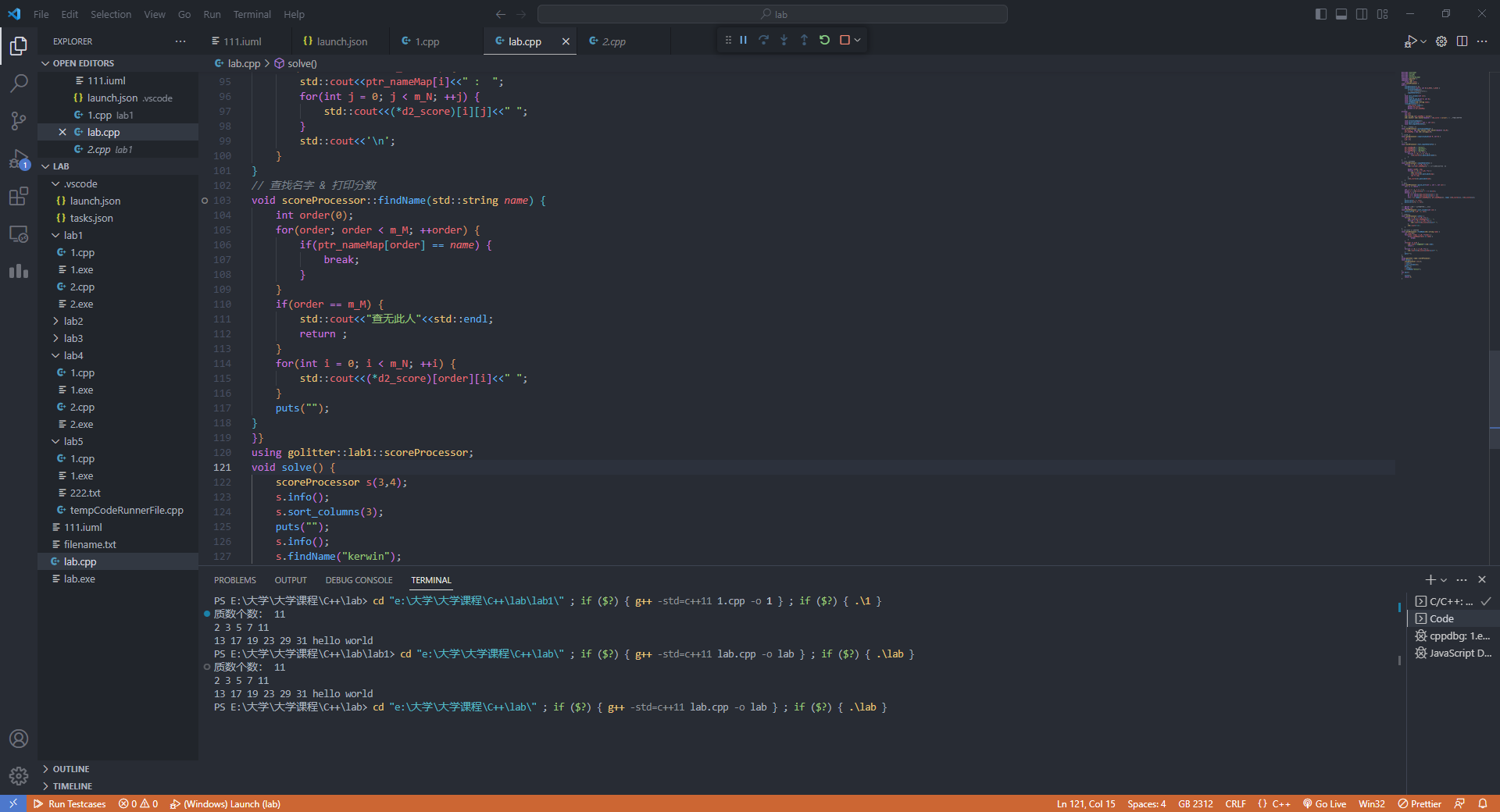
solve();

return 0;

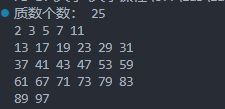
}

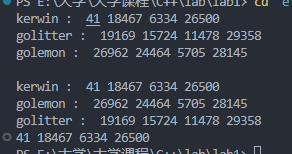
1. 调试过程





1. 实验结果





1. 小结

通过本次实验，我学到了以下知识点：

通过类的定义，实现了一个具有线性筛选功能的算法和一个学生成绩信息处理系统。

学习了动态内存分配的相关知识，并且熟悉了对指针的操作方法。

实现了快速排序算法，并熟悉了快排的基本思想。

学会了在程序中使用 STL 类库中的 vector 和 string 等容器，并掌握了相应的操作方法。

学习了如何在程序中定义并使用命名空间，避免了命名冲突。

提高了编程能力和解决问题的技巧，通过做题加深了对 C++ 语言的理解和掌握。