Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного обеспечения и администрирования информационных систем

Направление подготовки математическое обеспечение и администрирование информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе №5**

«Стандартная библиотека шаблонов»

Выполнил:

студент группы 213.1 Козявин М. С.

Проверил:

старший преподаватель кафедры ПОиАИС Ураева Е. Е.

Курск, 2022

***Цель работы:*** Изучить особенности написания программ на языке С++

с использованием стандартной библиотеки шаблонов.

***Задание***

*Задача 1.* Дан массив целых чисел размера *n.* Перед каждым значением, которое является числом Люка вставить ноль или сообщить, что таких элементов нет.  29. Дан массив целых чисел размера *n.* Перед первым минимальным элементом массива вставить все его делители.

*Задача 2.* Дан текст на русском языке. Напечатать в алфавитном порядке все звонкие согласные буквы, которые входят более чем в одно слово.

*Задача 3.* Разработать и согласовать с преподавателем набор функций (методов) для базового (или более высокого) уровня задания курсового проекта, использующих контейнеры map (или multimap).

*Задача 4.* Дан вектор *v*. Найти сумму отрицательных и сумму положительных элементов вектора. Использовать два вызова алгоритма *accumulate* с параметрами — функциональными объектами.

***Разработка алгоритма***

*Задача 1*

Входные данные: *arr – массив целых чисел*.

Выходные данные: *arr – массив целых чисел*.

*Задача 2*

Входные данные: *text – исходная строка*.

Выходные данные: *text – строка из отсортированных звонких согласных букв*.

*Задача 3*

Входные данные: *файл с данными рекордов игроков*.

Выходные данные: *таблица рекордов игроков*.

*Entry1* – класс с данными для заполнения таблицы для одного игрока. Реализован следующим набором полей и методов:

*map* – название карты

*pl1name* – имя первого игрока

difficulty – сложность

*lives* – количество жизней игроков

*pl1score* – набранные очки первого игрока

*time* – время игры

*addTo()*

Метод добавления данных в таблицу.

Входные данные: указатель на таблицу

Выходные данные: отсутствуют

*Entry2* – класс с данными для заполнения таблицы для двух игроков. Наследуется от *Entry1.* Реализован следующим набором полей и методов:

*pl2name* – имя первого игрока

*pl2score* – набранные очки первого игрока

*Задача 4*

Входные данные: *arr – массив целых чисел*.

Выходные данные: *два целых числа – сумма положительных и отрицательных чисел массива*.

***Текст программы***

*Текст программы для решения задач 1, 4*

#include <iostream>

#include <vector>

#include <set>

#include <cmath>

#include <numeric>

using namespace std;

void taskOne();

void fillArr(vector<int> &arr);

bool isL(int num);

void printVector(vector<int> arr);

void taskFour();

int sumPositive(int a, int b);

int sumNegative(int a, int b);

int main()

{

int taskNumber;

cout << "task: ";

cin >> taskNumber;

while (!cin.eof())

{

switch (taskNumber)

{

case 1:

taskOne();

cout << "task: ";

cin >> taskNumber;

break;

case 4:

taskFour();

cout << "task: ";

cin >> taskNumber;

break;

default:

cout << "input error\n";

cout << "ctrl + z for exit.\n";

cin.clear();

cin.ignore(1000, '\n');

cout << "task: ";

cin >> taskNumber;

break;

}

}

}

void taskOne () {

vector <int> arr;

fillArr(arr);

int i = 0;

while (i < arr.size()) {

if ( isL(arr.at(i)) ) {

arr.insert(arr.begin()+i, 0);

i += 2;

} else {

i++;

}

}

printVector(arr);

}

void fillArr(vector<int> &arr) {

int input;

cout << "numbers: ";

while (cin >> input)

arr.push\_back(input);

cin.clear();

cin.ignore(1000, '\n');

}

bool isL(int num) {

for (int i = 1; i < 100; i++) {

if (pow((1+sqrt(5))/2, i)+pow((1-sqrt(5))/2, i) == num) {

return true;

}

}

return false;

}

void printVector(vector<int> arr) {

for (int i : arr) {

cout << i << " ";

}

cout << endl;

}

void taskFour() {

vector <int> arr;

fillArr(arr);

cout << "Positive: " << accumulate(arr.begin(), arr.end(), 0, sumPositive) << endl;

cout << "Negative: " << accumulate(arr.begin(), arr.end(), 0, sumNegative) << endl;

}

int sumPositive(int a, int b) {

if (b > 0) {

return a+b;

} else {

return a;

}

}

int sumNegative(int a, int b) {

if (b < 0) {

return a+b;

} else {

return a;

}

}

***Текст программы для решения задачи 2***

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <vector>

#include <set>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

QString text = ui->textEdit->toPlainText();

std::vector<QString> targetArr = {"б", "в", "г", "д", "ж", "з", "й", "л", "м", "н", "р"};

std::vector<QString> arr;

std::multiset<QString> set;

QStringList list = text.split(' ');

for (QString str : list) {

for (QString target : targetArr) {

if (str.contains(target)) {

set.insert(target);

}

}

}

ui->textEdit\_2->clear();

for (QString target : targetArr) {

if (set.count(target) > 1) {

ui->textEdit\_2->append(" " + target);

}

}

}

***Текст программы для решения задачи 3***

#ifndef RECORDS\_H

#define RECORDS\_H

#include <QWidget>

#include <QTableWidget>

#include <QFile>

#include <QTextStream>

#include <QStandardItemModel>

#include <widget.h>

#include <map>

class Entry1 {

public:

QString map;

QString pl1name;

int difficulty;

int lives;

int pl1score;

float time;

Entry1(QStringList arr) {

map = arr.at(0);

difficulty = arr.at(1).toInt();

lives = arr.at(2).toInt();

pl1name = arr.at(3);

pl1score = arr.at(4).toInt();

time = arr.at(5).toFloat();

}

Entry1 () {pl1score = -1;};

virtual void addTo(QTableWidget\* table) {

int count = table->rowCount();

table->setRowCount(count+1);

table->setItem(count, 0, new QTableWidgetItem(map));

switch (difficulty) {

case 1:

table->setItem(count, 1, new QTableWidgetItem("Easy"));

break;

case 2:

table->setItem(count, 1, new QTableWidgetItem("Medium"));

break;

case 3:

table->setItem(count, 1, new QTableWidgetItem("Hard"));

break;

default:

break;

}

QTableWidgetItem\* item = new QTableWidgetItem();

item->setData(Qt::DisplayRole, lives);

table->setItem(count, 2, item);

item = new QTableWidgetItem();

table->setItem(count, 3, new QTableWidgetItem(pl1name));

item->setData(Qt::DisplayRole, pl1score);

table->setItem(count, 4, item);

item = new QTableWidgetItem();

item->setData(Qt::DisplayRole, time);

table->setItem(count, 5, item);

}

};

class Entry2: public Entry1 {

public:

QString pl2name;

int pl2score;

Entry2 (QStringList arr) {

map = arr.at(0);

difficulty = arr.at(1).toInt();

lives = arr.at(2).toInt();

pl1name = arr.at(3);

pl2name = arr.at(4);

pl1score = arr.at(5).toInt();

pl2score = arr.at(6).toInt();

time = arr.at(7).toFloat();

}

Entry2 () {pl1score = -1;};

void addTo(QTableWidget\* table) {

int count = table->rowCount();

table->setRowCount(count+1);

table->setItem(count, 0, new QTableWidgetItem(map));

switch (difficulty) {

case 1:

table->setItem(count, 1, new QTableWidgetItem("Easy"));

break;

case 2:

table->setItem(count, 1, new QTableWidgetItem("Medium"));

break;

case 3:

table->setItem(count, 1, new QTableWidgetItem("Hard"));

break;

default:

break;

}

QTableWidgetItem\* item = new QTableWidgetItem();

item->setData(Qt::DisplayRole, lives);

table->setItem(count, 2, item);

table->setItem(count, 3, new QTableWidgetItem(pl1name));

table->setItem(count, 4, new QTableWidgetItem(pl2name));

item = new QTableWidgetItem();

item->setData(Qt::DisplayRole, pl1score);

table->setItem(count, 5, item);

item = new QTableWidgetItem();

item->setData(Qt::DisplayRole, pl2score);

table->setItem(count, 6, item);

item = new QTableWidgetItem();

item->setData(Qt::DisplayRole, time);

table->setItem(count, 7, item);

}

};

namespace Ui {

class Records;

}

class Records : public QWidget

{

Q\_OBJECT

public:

explicit Records(QWidget \*parent = nullptr);

void fill1player();

void fill2players();

void load();

bool twoPlayersMode = false;

std::map<QString,Entry1> map1pl;

std::map<QString,Entry2> map2pl;

QStandardItemModel\* model;

QTableWidget\* table;

~Records();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

private:

Ui::Records \*ui;

};

#endif // RECORDS\_H

// records.cpp

void Records::load() {

QString endPath = QCoreApplication::applicationDirPath() + "/records/records1player.txt";

QFile file(endPath);

file.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in(&file);

while (!in.atEnd()) {

QString data = in.readLine();

QStringList splited = data.split('|');

if (splited.size() < 6) {continue;};

Entry1 entry1 = Entry1(splited);

auto id = entry1.pl1name + entry1.difficulty + entry1.lives + entry1.map;

int count = map1pl.count(id);

if (count == 0 || (count > 0 && map1pl.at(id).pl1score\*(1/map1pl.at(id).time) < entry1.pl1score\*(1/entry1.time))) {

map1pl[id] = entry1;

}

}

file.close();

endPath = QCoreApplication::applicationDirPath() + "/records/records2players.txt";

QFile file2(endPath);

file2.open(QIODevice::ReadOnly | QIODevice::Text);

QTextStream in2(&file2);

while (!in2.atEnd()) {

QString data = in2.readLine();

QStringList splited = data.split('|');

if (splited.size() < 8) {continue;};

Entry2 entry2 = Entry2(splited);

map2pl[entry2.pl1name + entry2.pl2name] = entry2;

}

file2.close();

}

***Тестирование программы***

Тестирование задачи 1 представлено на рисунке 1.

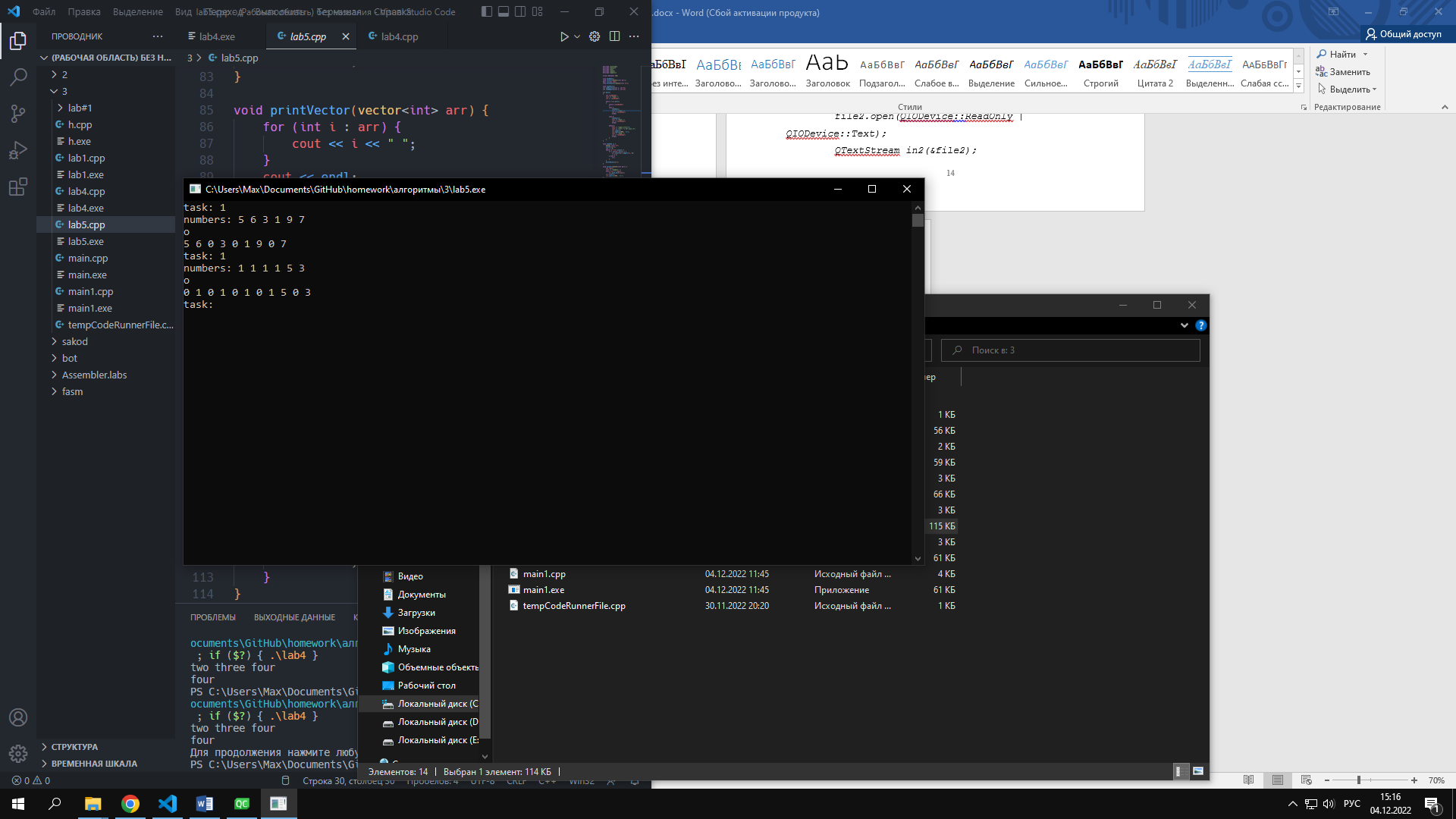


Рисунок 1 - Тест 1-2 задачи 1

Тестирование задачи 2 представлено на рисунках 2-3.

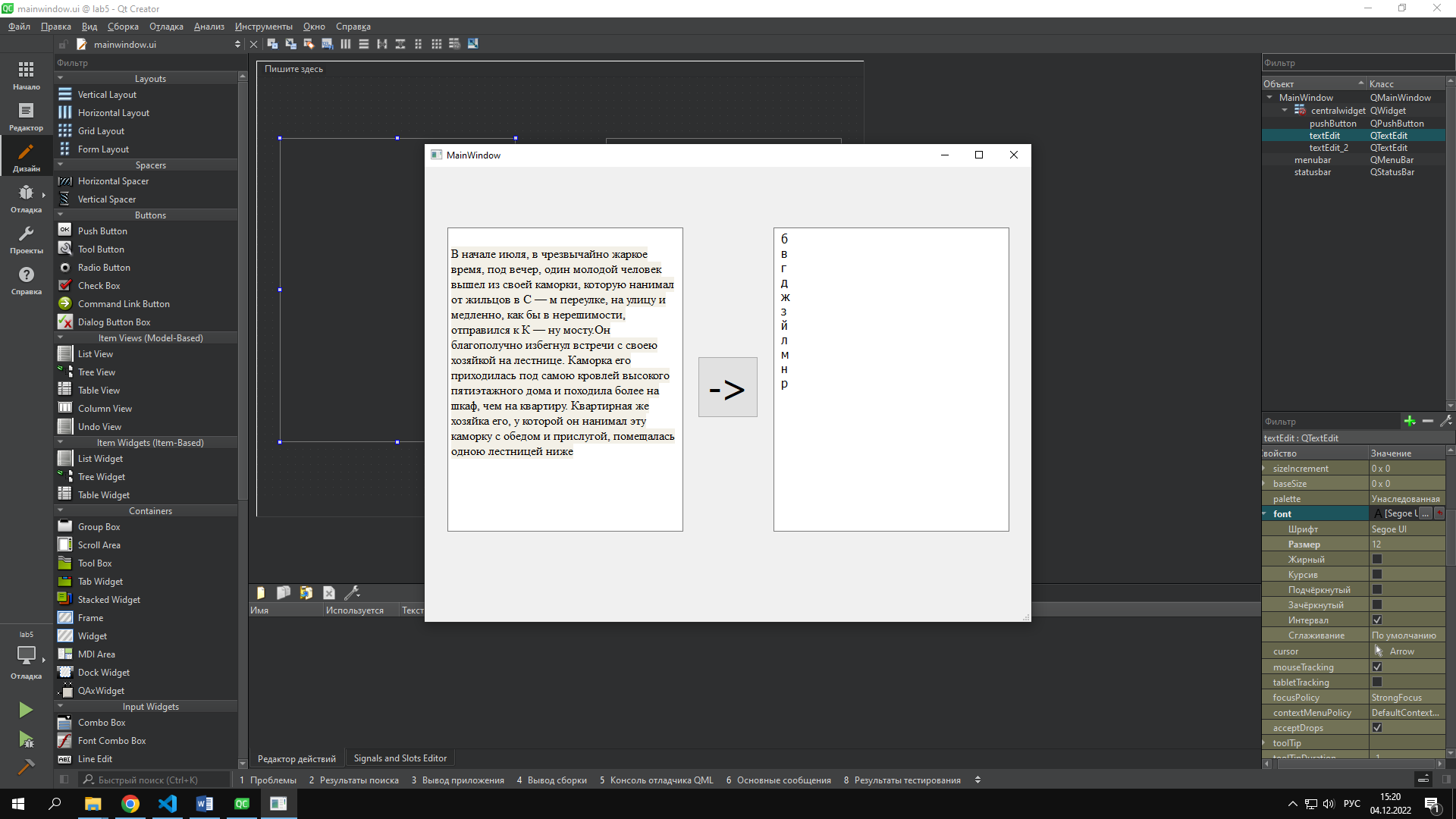


Рисунок 2 - Тест 1 задачи 2

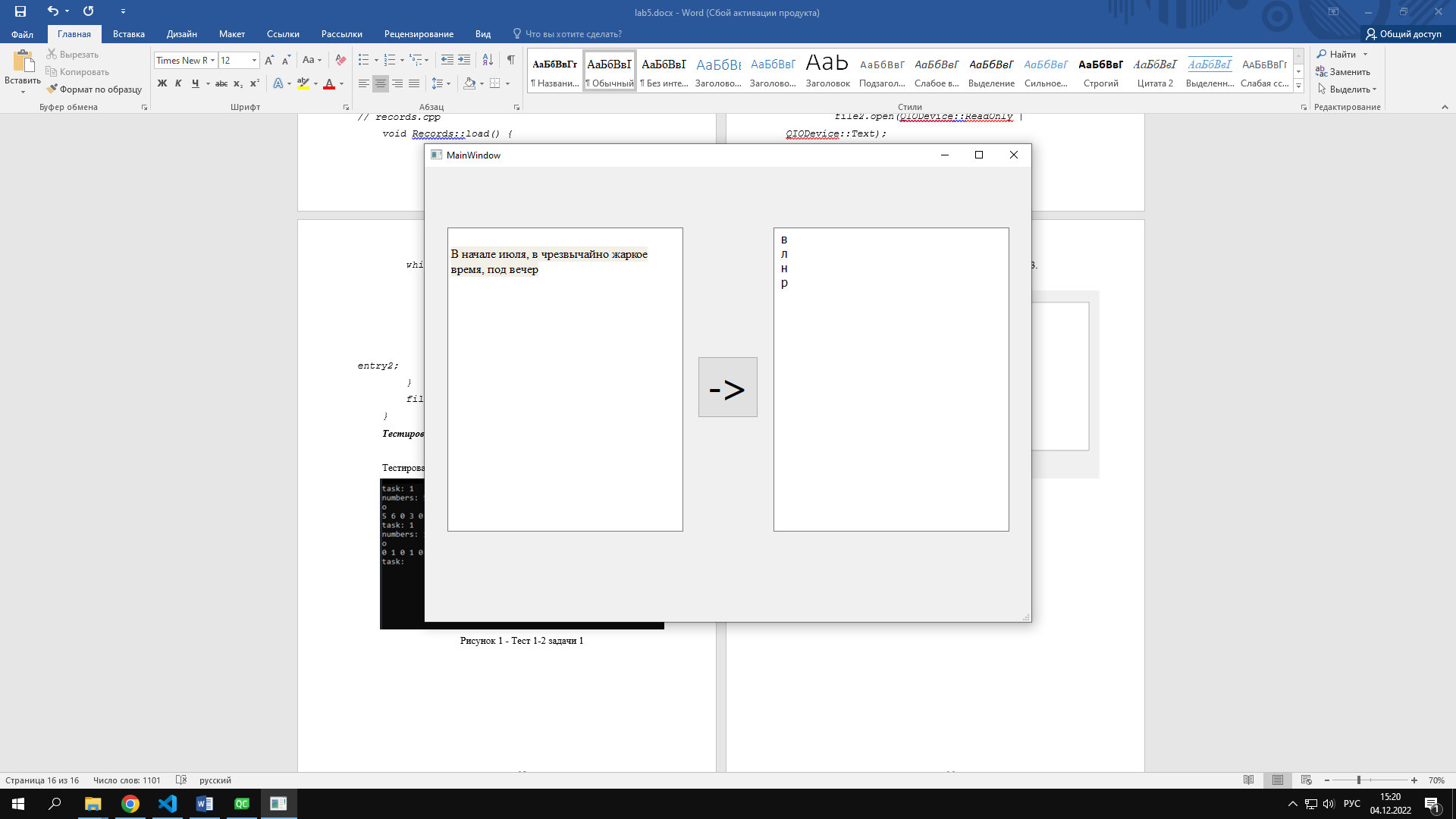


Рисунок 3 - Тест 2 задачи 2

Тестирование задачи 3 представлено на рисунках 4-5.

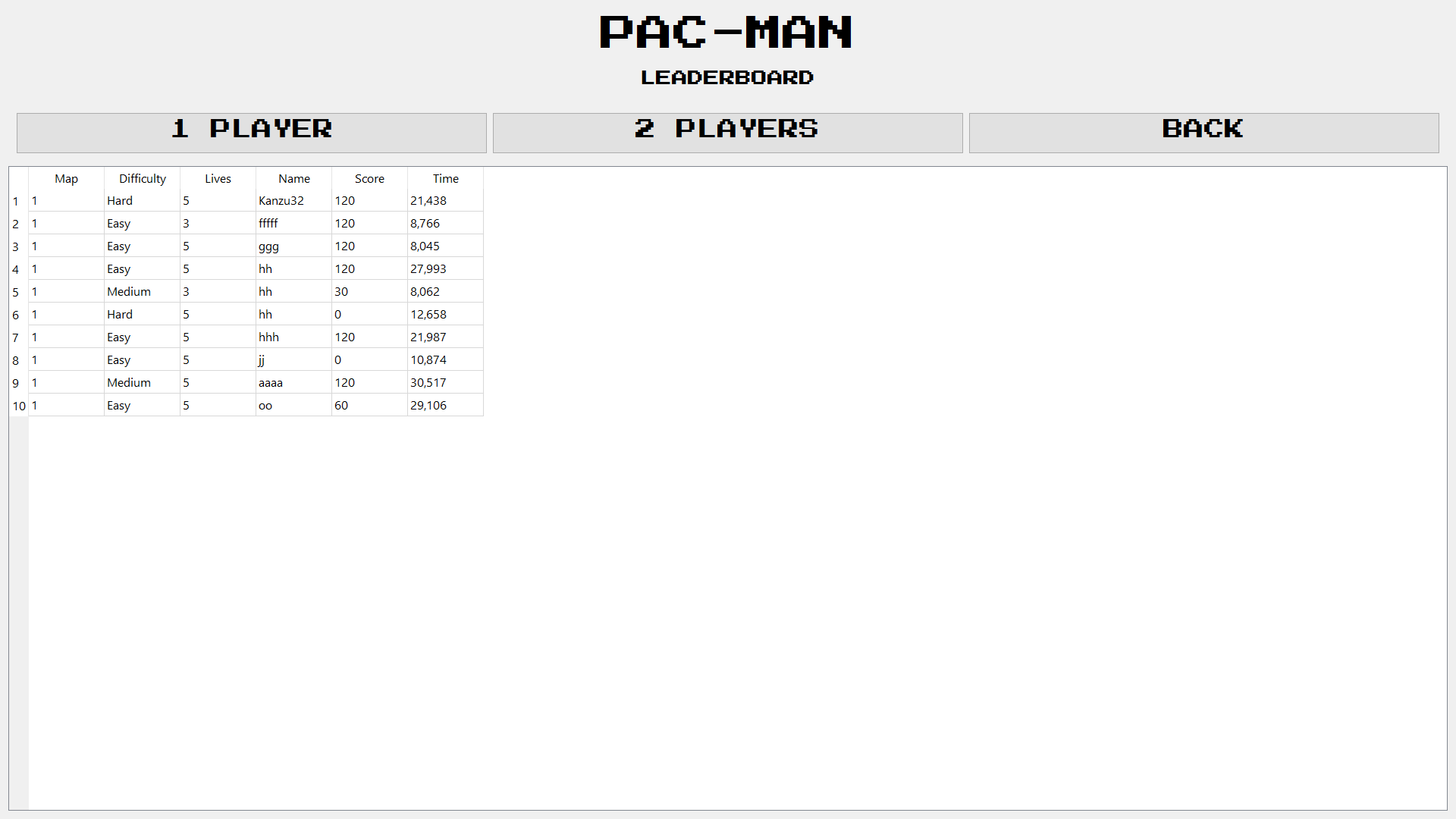


Рисунок 4 - Тест 1 задачи 3

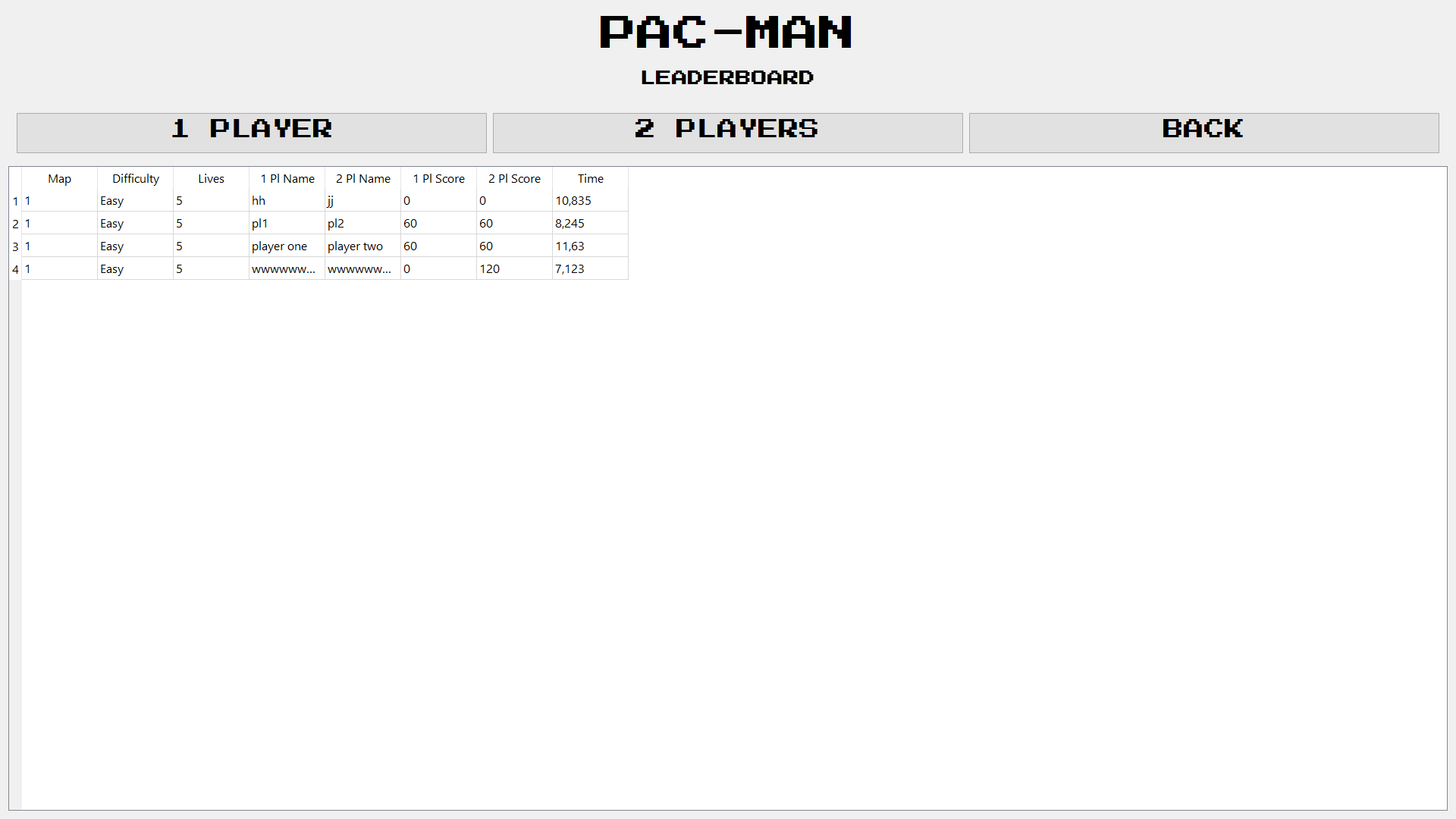


Рисунок 5 - Тест 2 задачи 3

Тестирование задачи 4 представлено на рисунке 6.

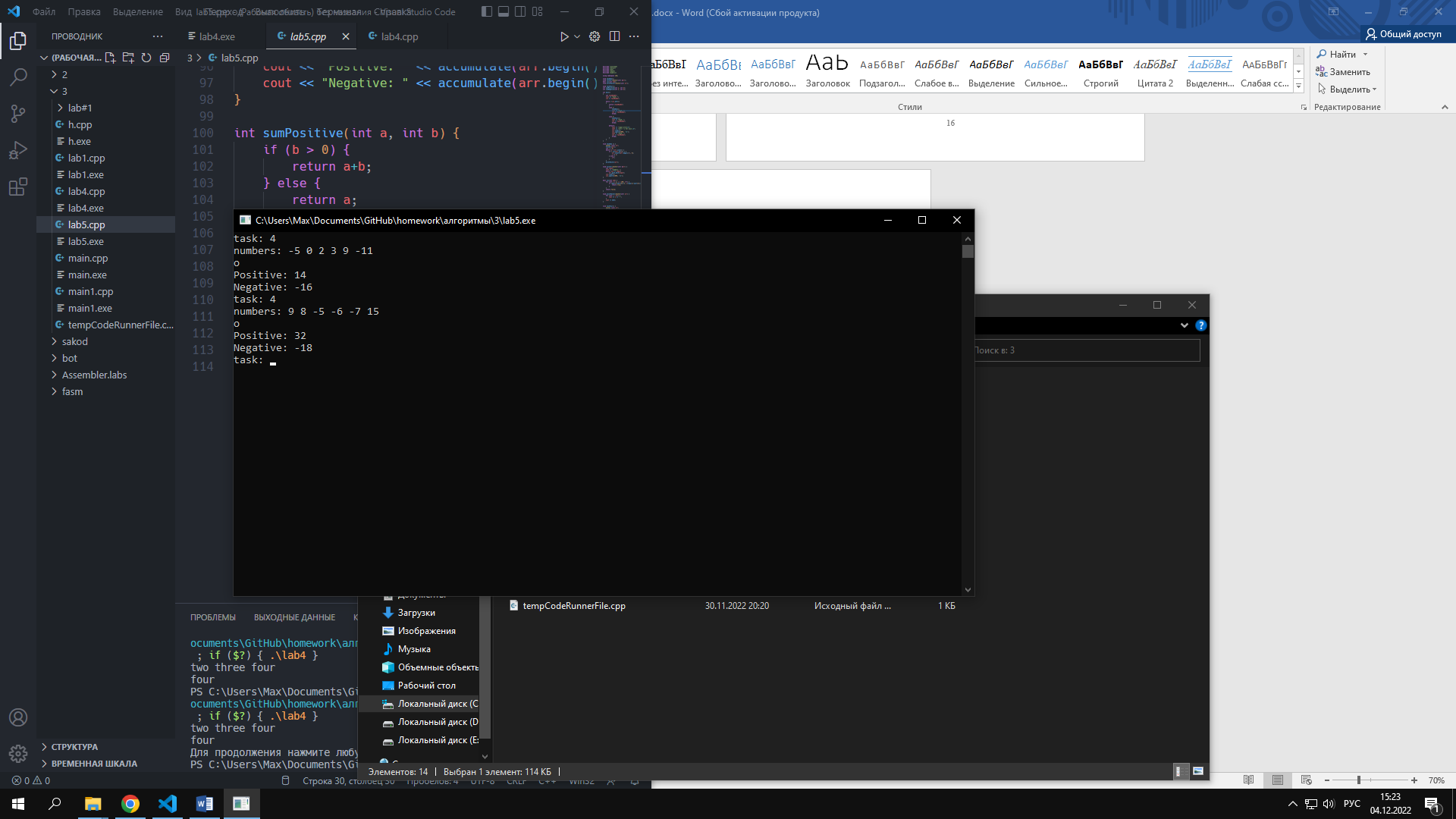


Рисунок 6 - Тест 1-2 задачи 4