ГУАП

КАФЕДРА № 53

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ассистент |  |  |  | С. Ю. Гуков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| №3.2. АЛГОРИТМ С ИСПОЛЬЗОВАНИЕМ РЕКУРСИИ И ЦИКЛОВ |
| Вариант 3 (15) |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 5136 |  | , |  | Д. Л. Мягков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

Оглавление:

[Цель. 3](#__RefHeading___Toc11772_844291809)

[Индивидуальное задание](#__RefHeading___Toc11774_844291809) [4](#__RefHeading___Toc12665_3292820999)

[Описание алгоритма 5](#__RefHeading___Toc11776_844291809)

[Листинг программы](#__RefHeading___Toc11778_844291809) [30](#__RefHeading___Toc12667_3292820999)

[Вывод. 33](#__RefHeading___Toc11780_844291809)

# Цель.

Задание: Реализовать программную функцию на языке C/С++, выполняющую поставленную задачу с использованием рекурсии. Вариант задания, пример входных и выходных данных представлен в таблице 7.

* Написать код функции, принимающей в качестве аргументов и возвращающей все необходимые параметры, без использования глобальных переменных. Допустимо использование дополнительных функций.
* Протестировать функцию для всех возможных исключительных ситуаций, особое значение придается текстам на возникновение ошибок в ходе работы программы.
* Из наименования функции и принимаемых аргументов должно быть ясно их назначение.
* В ходе тестирования функции при каждом вызове рекурсивной функции необходимо вывести отладочную информацию: порядковый номер вызова рекурсивной функции, значения изменяющегося аргумента и возвращаемого значения, если они присутствуют. Привести глубину рекурсии для каждого тестового примера.

# Индивидуальное задание.

## 

Рисунок 1. Индивидуальное задание

# Описание алгоритма

Алгоритм является рекурсией. На каждом вызове, количество вызовов увеличивается, глубина увеличивается, в массив шагов добавляется шаг, в массив точек добавляется точка, в массив глубины добавляется глубина, в массив памяти добавляется точка.

Если вышли за границы исходного прямоугольника, то возвращаем ноль.

Если уровень глубины больше максимального то максимальному уровню присваиваем уровень глубины — 1.

Если пришли в точку, в которой уже были, возвращаем ноль.

Если ширина на высоту равно размеру массива маршрута, то возвращаем 1.

Дальше идет цикл от 0 до размера исходного размера шагов. На каждом шаге к координатам точки, где мы сейчас находимся прибавлются координаты точки куда мы придем. Потом вызывается эта же функция.

Если вызов функции вернул 1, то в массив шагов добавляем шаг, в массив точек добавляем точку, добавляем маршрут в массив маршрутов, удаляем последние элементы из массива шагов и массива точек. Если счетчик i != T.size() - 1, печатаем "Идем на предыдущий шаг:", строку маршрута и "Идем дальше:". Иначе, если счетчик i != T.size() - 1, печатаем "Весь маршрут:", маршрут, исходную область с нарисованным маршрутом и "Идем дальше:".

После проверки вызова функции, отнимает от координат А, координаты точки T[i].

Таким образом находим все маршруты.

# Листинг программы.

Листинг lab\_3\_2.cpp:

#include <iostream>// библиотека

#include <iomanip>

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include "main\_header.h"

#include "string"

#include "vector"

#include <set>

#include <iterator>

#include "Point.h"

using namespace std; //подключение пространства имен std

struct Values

{

bool p\_o\_n\_p\_intermediate\_values = 1,

print\_or\_not\_print\_find\_array = 1,

p\_o\_n\_p\_final\_values = 1;

};

struct Routes

{

vector <Point> route;

vector <Point> step;

};

void print\_vector\_of\_vectors(int height, int width, vector <Routes> array, bool print\_table\_route, bool print\_on\_place, int speed);

string Point\_to\_string(Point A);

void print\_string(int call\_count, int deep\_level, vector<Point> memory);

void print\_vector\_of\_point(vector<Point> T);

void print\_string(int deep\_level, Point A);

bool in\_route(vector<Point> memory, Point A);

void print\_area(int height, int width, vector <Point> route, int size);

void print\_route(vector<Point> memory);

vector<vector<Point>> fill\_variants();

void rec\_fill\_variants(int last, int m, vector<int> indexes, vector<vector<Point>>& variants, vector<Point> T);

bool find\_route(Point step, vector <Routes>& routes, int width, int height, vector<Point> T, Routes route, Point A, int count, int& call\_count, vector<Point>& memory, int deep\_level, vector<int>& levels, int& max\_deep\_level);

void lab\_3\_2()

{

setlocale(0, ""); // поддержка кириллицы в консоли (вывод)

SetConsoleCP(1251); // поддержка кириллицы в консоли (ввод)

SetConsoleOutputCP(1251); // поддержка кириллицы в консоли (ввод)

Values values;

int call\_count = 0, deep\_level = 0, max\_deep\_level = 0, height = 3, width = 2, count = 0;

/\*

cout << "Введите размеры поля \nВведите height: ";

int height = get\_number<int>();

cout << "Введите width: ";

int width = get\_number<int>();

\*/

vector <Point> T, memory = {};

Routes route = {};

vector <int> levels = {};

vector <Routes> routes = {};

//T = { {2, 1},{-2, -1},{1, 2},{-1, -2} };

T = { {0, 1},{0, -1},{1, 0},{-1, 0} };

//T = { {0, 1}, {0, -1},{-1, 0},{2, 1} };

//values.print\_or\_not\_print = 0;

/\*cout << "Введите (Ax,Ay)\nВведите A.x: ";

A.x = get\_number<int>();

cout << "Введите A.x: ";

A.x = get\_number<int>();

\*/

Point A = { 2,2 };

/\*for (int y = 0; y < 4; y++)

{

vector<int> tmp;

for (int y = 0; y < 2; y++)

{

cout << "Введите " << y << " элемент " << y << "элемента массива T: ";

tmp.push\_back(get\_number<int>());

}

T.push\_back(tmp);

}\*/

route.route.push\_back(A);

print\_area(height, width, route.route, route.route.size());

route.route.pop\_back();

cout << "Поиск массивов ходов (1) или проверка работы (0): ";

bool task;

task = get\_number < bool>();

//task = 1;

if (task == 0)

{

if (values.p\_o\_n\_p\_intermediate\_values == 1)

cout << "\nКоличество вызовов:| Глубина:| Значение:\n";

find\_route({}, routes, width, height, T, route, A, count, call\_count, memory, deep\_level, levels, max\_deep\_level);

if (routes.size() != 0)

{

cout << "Найдены маршруты! Количество: " << routes.size() << endl;

if (values.p\_o\_n\_p\_final\_values == 1)

{

cout << "Печатать (1) или не печатать (0) путь графически: ";

bool print\_table\_route, print\_on\_place = 0;

int speed = 36;

print\_table\_route = get\_number < bool>();

if (print\_table\_route)

{

cout << "Печатать на одном месте (1) или нет (0): ";

print\_on\_place = get\_number < bool>();

if (print\_on\_place)

{

cout << "Введите скорость печатания (от 30 до 50 или другую) или нажмите пробел: ";

speed = input\_value<int>(36);

}

}

cout << endl;

print\_vector\_of\_vectors(height, width, routes, print\_table\_route, print\_on\_place, speed);

}

}

else

cout << "Невозможно" << endl;

}

else

{

cout << "Условие: ходы в диапазоне [-2;2], размер массива [1;4]\n";

vector<vector<Point>> variants = fill\_variants();

int j = 1;

cout << "Печатать (1) или не печатать (0) путь графически: ";

bool print\_table\_route, print\_on\_place = 0;

print\_table\_route = get\_number < bool>();

if (print\_table\_route)

{

cout << "Печатать на одном месте (1) или нет (0): ";

print\_on\_place = get\_number < bool>();

}

//bool routes\_find = 0;

for (int i = 0; i < variants.size(); i++)

{

T = variants[i];

routes = {};

levels = {};

memory = {};

route = {};

find\_route({}, routes, width, height, T, route, A, count, call\_count, memory, deep\_level, levels, max\_deep\_level);

if (routes.size() != 0)

{

//routes\_find = 1;

if (values.print\_or\_not\_print\_find\_array == 1)

{

cout << "Массив ходов: " << endl;

print\_vector\_of\_point(T);

print\_vector\_of\_vectors(height, width, routes, print\_table\_route, print\_on\_place, 36);

}

}

}

//if (routes\_find == 0)

// cout << "Таких массивов нет" << endl;

}

}

bool find\_route(Point step, vector <Routes>& routes, int width, int height, vector<Point> T, Routes route, Point A, int count, int& call\_count, vector<Point>& memory, int deep\_level, vector<int>& levels, int& max\_deep\_level)

{

Values values;

//values.print\_or\_not\_print = 1;

call\_count++;

deep\_level++;

//print\_vector\_of\_vectors(route);

levels.push\_back(deep\_level - 1);//запись в массив уровня глубины

route.step.push\_back(step);

route.route.push\_back(A);//запись в массив значения числа

//запись в массив значения числа

memory.push\_back(A);

if (values.p\_o\_n\_p\_intermediate\_values == 1)

print\_string(call\_count, deep\_level - 1, route.route);

if (A.x < 1 || A.y < 1 || A.x > width || A.y > height)

{

//levels.pop\_back();

//route.pop\_back();

if (values.p\_o\_n\_p\_intermediate\_values == 1)

cout << "\nВышли за границы. \n";

return 0;

}

if (deep\_level > max\_deep\_level)

{

max\_deep\_level = deep\_level - 1;//нахождение максимальной глубины

}

if (in\_route(route.route, A))

{

//A.x -= T[y].x;

//A.x -= T[y].x;

if (values.p\_o\_n\_p\_intermediate\_values == 1)

cout << "\nВернулись в одну из предыдущих точек. \n";

return 0;

}

if (width \* height == route.route.size())

{

if (values.p\_o\_n\_p\_intermediate\_values == 1)

cout << "\nНашли путь!";

return 1;

}

for (int i = 0; i < T.size(); i++)

{

A.x += T[i].x;

A.y += T[i].y;

if (find\_route(T[i], routes, width, height, T, route, A, count, call\_count, memory, deep\_level, levels, max\_deep\_level))

{

route.route.push\_back(A);

route.step.push\_back(step);//запись в массив значения числа

if (values.p\_o\_n\_p\_intermediate\_values == 1)

{

cout << "\nПуть:\n";

print\_route(route.route);

}

routes.push\_back(route);

route.route.pop\_back();

route.step.pop\_back();

if (i != T.size() - 1)

if (values.p\_o\_n\_p\_intermediate\_values == 1)

{

cout << "Идем на предыдущий шаг: \n";

//cout << "Точка в которую вернулись:" << endl;

print\_string(call\_count, deep\_level - 1, route.route);

cout << "Идем дальше:" << endl;

}

}

else

{

if (i < T.size() - 1)

{

if (values.p\_o\_n\_p\_intermediate\_values == 1)

{

cout << "\nВесь маршрут:" << endl;

print\_route(route.route);

print\_area(height, width, route.route, route.route.size());

cout << "Идем дальше:" << endl;

}

}

}

//deep\_level--;

A.x -= T[i].x;

A.y -= T[i].y;

}

if (values.p\_o\_n\_p\_intermediate\_values == 1)

{

cout << "Закончились варианты следующего шага для точки " << Point\_to\_string(A) << ". \n\n";

if (route.route.size() > 1)

cout << "Идем в предыдущую точку " << Point\_to\_string(route.route[route.route.size() - 2]) << ". \n\n";

}

return 0;

}

void print\_area(int height,int width, vector <Point> route, int size)

{

vector <vector <char>> P(height \* 2 + 3);

P[0].push\_back(' ');

P[0].push\_back('|');

for (int j = 0; j < width; j++)

{

P[0].push\_back(number\_to\_str(j + 1)[0]);

P[0].push\_back('|');

}

for (int j = 0; j < width \* 2 + 2; j++)

{

P[1].push\_back('-');

}

for (int i = 2; i < height \* 2 + 2; i++)

{

P[i].push\_back(number\_to\_str((i)/2)[0]);

P[i].push\_back('|');

for (int j = 0; j < width; j++)

{

P[i].push\_back(' ');

P[i].push\_back('|');

}

i++;

for (int j = 0; j < width \* 2 + 2; j++)

{

P[i].push\_back('-');

}

}

for (int i = 0; i < size; i++)

{

if (route[i].x > 0 && route[i].y > 0)

{

P[route[i].y \* 2][route[i].x \* 2] = '.';

}

}

for (int i = P.size() - 1; i > -1; i--)

{

for (int j = 0; j < P[i].size(); j++)

{

cout << P[i][j];

}

cout << endl;

}

cout << endl;

}

void print\_vector\_of\_vectors(int height, int width, vector <Routes> array, bool print\_table\_route, bool print\_on\_place, int speed)

{

for (int i = 0; i < array.size(); i++)

{

for (int k = i + 1; k < array.size(); k++)

{

int s = 0;

if (array[i].route[0].x != -1 &&

array[i].route[0].y != -1)

{

for (int m = 0; m < array[i].route.size(); m++)

{

if (array[i].route[m].x == array[i].route[m].x && array[i].route[m].y == array[i].route[m].y)

s++;

else break;

}

if (s < array[i].route.size())

{

array[k].route[0].x = -1;

array[k].route[0].y = -1;

}

}

}

}

//{4,6,3,6,7}

//{4,6,3,-1,7}

for (int i = 0; i < array.size(); i++)

{

if (array[i].route[0].x != -1 &&

array[i].route[0].y != -1)

{

cout << "Маршрут " << i + 1 << endl;

if (print\_table\_route)

{

if (print\_on\_place)

{

//cout << endl << "Исходная точка:" << endl;

//cout << setw(3) << 0 << ": point: " << setw(6) << Point\_to\_string(array[i].route[0]) << "| step:" << setw(8) << Point\_to\_string(array[i].step[0]) << endl;

//print\_area(height, width, array[i].route, 1);

for (int j = 0; j < array[i].route.size(); j++)

{

Point A = cursor\_position();

cout << setw(3) << j << ": point: " << setw(6) << Point\_to\_string(array[i].route[j]) << "| step:" << setw(8) << Point\_to\_string(array[i].step[j]) << endl;

print\_area(height, width, array[i].route, j + 1);

for (int k = 0; k < 10000000; k += 3)

{

k -= 2;

for (int l = 0; l < speed; l += 3)

{

l -= 2;

}

}

erase\_past\_output(A);

}

}

else

{

cout << setw(3) << 0 << ": point: " << setw(6) << Point\_to\_string(array[i].route[0]) << "| step:" << setw(8) << Point\_to\_string(array[i].step[0]) << endl;

print\_area(height, width, array[i].route, 1);

for (int j = 1; j < array[i].route.size(); j++)

{

cout << setw(3) << j << ": point: " << setw(6) << Point\_to\_string(array[i].route[j]) << "| step:" << setw(8) << Point\_to\_string(array[i].step[j]) << endl;

print\_area(height, width, array[i].route, j + 1);

}

}

}

for (int j = 0; j < array[i].route.size(); j++)

{

cout << setw(3) << j << ": point: " << setw(6) << Point\_to\_string(array[i].route[j]) << "| step:" << setw(8) << Point\_to\_string(array[i].step[j]) << endl;

}

cout << endl;

/\*cout << "Весь маршрут: " << endl;

for (int j = 0; j < array[i].route.size(); j++)

{

cout << endl;\*/

}

}

}

void print\_vector\_of\_vectors(vector <Routes> array)

{

for (int i = 0; i < array.size(); i++)

{

int unique = 1;

for (int k = i + 1; k < array.size(); k++)

{

int m = 0;

while (array[i].route.size() != m && array[k].route.size() != m)

{

if (array[i].route[m].x == array[i].route[m].x && array[i].route[m].y == array[i].route[m].y)

unique++;

m++;

}

}

if (unique == 1)

{

cout << "Маршрут " << i + 1 << endl;

for (int j = 0; j < array[i].route.size(); j++)

{

cout << j << ": point: " << setw(6) << Point\_to\_string(array[i].route[j]) << "| step:" << setw(6) << Point\_to\_string(array[i].step[j]) << endl;

}

cout << endl;

}

}

}

string Point\_to\_string(Point A)

{

string S = number\_to\_str<int>(A.x) + ", " + number\_to\_str<int>(A.y);

return S;

}

void print\_string(int call\_count, int deep\_level, vector<Point> memory)

{

cout

<< setw(19) << call\_count << "|"

<< setw(9) << deep\_level << "|"

<< setw(9) << Point\_to\_string(memory[deep\_level]) << "|"

<< endl;

}

void print\_vector\_of\_point(vector<Point> T)

{

cout << "{";

for (int i = 0; i < T.size(); i++)

{

cout << "{" << Point\_to\_string(T[i]) << "}";

if (i != T.size() - 1)

{

cout << ',';

}

}

cout << "}" << endl;

}

void print\_string(int deep\_level, Point A)

{

cout

<< setw(20) << "|"

<< setw(9) << deep\_level << "|"

<< setw(9) << Point\_to\_string(A) << "|"

<< endl;

}

bool in\_route(vector <Point> memory, Point A)

{

bool m = 0;

for (int i = 0; i < memory.size() - 1; i++)

{

if (memory[i].x == A.x && memory[i].y == A.y)

{

m = 1;

}

}

return m;

}

bool in\_T(vector <Point> T)

{

bool m = 0;

for (int i = 0; i < T.size(); i++)

{

for (int j = i; j < T.size(); j++)

{

if (T[j].x == T[i].x && T[j].y == T[i].y)

{

m = 1;

break;

}

if (m)

break;

}

}

return m;

}

void print\_route(vector <Point> memory)

{

for (int i = 0; i < memory.size(); i++)

{

print\_string(i, memory[i]);

}

cout << endl;

}

vector<vector<Point>> fill\_variants()

{

vector<vector<Point>> variants;

vector<Point> T;

for (int x = -2; x <= 2; x++)

{

for (int y = -2; y <= 2; y++)

{

T.push\_back({ x, y });

}

}

vector<Point> P = {};

//for (int size = 1; size <= 4; size++)

//{

vector <int> indexes = {};

for (int i = 1; i <= 4; i++)

{

indexes = {};

rec\_fill\_variants(0, i, indexes, variants, T);

}

return variants;

}

void rec\_fill\_variants(int last, int m, vector<int> indexes, vector<vector<Point>>& variants, vector<Point> T)

{

/\*

/\*if (variants.size() == 649)

{

cout << endl;

}

if (P.size() == size)

{

return;

}

else

{

for (int i = 0; i < T.size(); i++)

{

P.push\_back(T[i]);

rec\_fill\_variants(i, variants, T, size, P);

/\*if (P.size() == size)

{

}

variants.push\_back(P);

P.pop\_back();

}

}\*/

if (indexes.size() == m)

{

vector<Point> P = {};

for (int index = 0; index < indexes.size(); index++)

{

//cout << indexes[index];

P.push\_back(T[indexes[index]]);

}

variants.push\_back(P);

return;

//cout << endl;

}

for (int index = last + 1; index <= T.size(); index++)

{

indexes.push\_back(index - 1);

rec\_fill\_variants(index, m, indexes, variants, T);

indexes.pop\_back();

}

}

Листинг Point.h:

#pragma once

struct Point

{

int x, y;

};

Листинг main.cpp:

#include <iostream>// библиотека

#include <Windows.h> // библиотека

#include "main\_header.h"

#include "various\_functions.h"

int main()

{

setlocale(0, ""); // поддержка кириллицы в консоли (вывод)

SetConsoleCP(1251); // поддержка кириллицы в консоли (ввод)

SetConsoleOutputCP(1251); // поддержка кириллицы в консоли (ввод)

double task\_number;

boolean debugging = true;

if (debugging == true)

{

//lab\_2\_1();

//lab\_5\_1();

//lab\_2\_2();

//lab\_4\_2();

//lab\_4\_1();

//dop\_task\_1();

//lab\_3\_1();

//lab\_3\_2();

lab\_3\_2\_copy();

}

else

{

std::cout

<< "Готовые лабораторные работы: " << std::endl

<< std::endl

<< " 1.1 Поиск макс. суммы двух элементов " << std::endl

<< " 1.2 Цикл с вычислением функции " << std::endl

<< " 1.3 Поиск неповторяющихся остатков от деления " << std::endl

<< " 2.1 Количество скобок " << std::endl

<< " 2.2 Бинарный поиск строки по хэшу " << std::endl

<< " 3.1 Рекурсия " << std::endl

<< " 4.1 Оценка алгоритма (Гаусс) " << std::endl

<< " 4.2 База данных " << std::endl

<< " 5.1 Битовая последовательность " << std::endl

<< std::endl

<< "Введите номер лабораторной работы: ";

while (true)

{

task\_number = get\_number<double>();

if (task\_number == 1.1 ||

task\_number == 1.2 ||

task\_number == 1.3 ||

task\_number == 2.1 ||

task\_number == 2.2 ||

task\_number == 3.1 ||

task\_number == 4.1 ||

task\_number == 4.2 ||

task\_number == 5.1)

{

system("cls");

std::cout << "Лабораторная работа № " << task\_number << std::endl;

break;

}

else std::cout << "Такой лабораторной работы нет, введите другой номер: ";

}

if (task\_number == 1.1)

{

lab\_1\_1();

}

else if (task\_number == 1.2)

{

lab\_1\_2();

}

else if (task\_number == 1.3)

{

lab\_1\_3();

}

else if (task\_number == 2.1)

{

lab\_2\_1();

}

else if (task\_number == 2.2)

{

lab\_2\_2();

}

else if (task\_number == 3.1)

{

lab\_3\_1();

}

else if (task\_number == 4.1)

{

lab\_4\_1();

}

else if (task\_number == 4.2)

{

lab\_4\_2();

}

else if (task\_number == 5.1)

{

lab\_5\_1();

}

}

return 0;

}

Листинг various\_functions.h:

#pragma once

#include <string>

#include <vector>

#include "Point.h"

using std::string;

using std::cout;

using std::cin;

using std::pair;

template <typename T>

T get\_number\_input\_after(string str);

template <typename T>

T input\_value(T a);

template <typename T>

T get\_number();

template <typename T>

bool check(string str);

template <typename T>

bool check\_without\_print(string str);

template <typename T>

T str\_to\_number(string str);

string char\_to\_string(char ch);

Point cursor\_position();

void set\_cursor\_position(Point A);

void erase\_past\_output(Point A);

template <typename T>

string number\_to\_str(T a)

{

string type = typeid(a).name();

string S = {}, s = {};

if (type == "int")

{

if (a < 0)

{

a = a \* -1;

s += '-';

}

while (true)

{

S += (a % 10) + '0';

a /= 10;

if (a == 0)

break;

}

for (int i = S.size() - 1; i != -1; i--)

{

s += S[i];

}

}

return s;

}

template <typename T>

T input\_value(T a)

{

string str;

getline(cin, str);

if (str[0] == '\0')

return a;

return get\_number\_input\_after<T>(str);

}

template <typename T>

T get\_number\_input\_after(string str)

{

while (true) // цикл продолжается до тех пор, пока пользователь не введет корректное значение

{

if (check<T>(str) == 1)

{

return str\_to\_number<T>(str);

}

getline(cin, str);

}

}

template <typename T>

T get\_number()

{

while (true) // цикл продолжается до тех пор, пока пользователь не введет корректное значение

{

string str;

getline(cin, str);

if (check<T>(str) == 1) //проверка на число

{

return str\_to\_number<T>(str); //преобразование строки в число

}

}

}

template <typename T>

bool check(string str)

{

bool k = 1;

if ((str.find('-') < 1 || str.find('-') == -1) && ((str.find('.') < str.size() - 1 && str.find('.') > 0) || str.find('.') == -1))

{

if (str.length() > 1)

{

int count\_minus = 0, count\_point = 0;

for (int i = 0; i < str.length(); i++)

{

if (str[i] == '-')

{

count\_minus++;

}

if (str[i] == '.')

{

count\_point++;

}

}

if ((str[0] == '0' && str[1] != '.') || count\_point > 1 ||

(str[0] == '-' && str[1] == '.') ||

(str[0] == '-' && str[1] == '0' && str[2] != '.') || count\_minus > 1)

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

}

else

if (str == "-")

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

if (k == 1)

{

string s = typeid(T).name();

if (s == "double")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '.' && str[i] != '-')

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

break;

}

}

}

if (s == "int")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '-')

{

k \*= 0;

std::cout << "!!Вы ввели не целое число!!" << std::endl << "Введитe целое число: ";

break;

}

}

}

if (s == "bool")

{

if (str[0] != '0' && str[0] != '1')

{

k \*= 0;

std::cout << "!!Вы ввели не двоичное число!!" << std::endl << "Введитe двоичное число: ";

}

}

}

}

else

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

return k;

}

template <typename T>

bool check\_without\_print(string str)

{

bool k = 1;

// 67642.5423

//проверка на возможность существования такого числа

if ((str.find('-') < 1 || str.find('-') == -1) && ((str.find('.') < str.size() - 1 && str.find('.') > 0) || str.find('.') == -1))

{

//если длина больше

if (str.length() > 1)

{

//проверка на возможность существования такого числа

//после нуля всегда точка или ничего, если ноль первый

//после - если ноль, то должна быть точка или ничего

int count\_minus = 0, count\_point = 0;

for (int i = 0; i < str.length(); i++)

{

if (str[i] == '-')

{

count\_minus++;

}

if (str[i] == '.')

{

count\_point++;

}

}

if ((str[0] == '0' && str[1] != '.') || count\_point > 1 ||

(str[0] == '-' && str[1] == '.') ||

(str[0] == '-' && str[1] == '0' && str[2] != '.') || count\_minus > 1)

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

}

else

if (str == "-")

{

k \*= 0;

}

if (k == 1)

{

string s = typeid(T).name();

if (s == "double")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '.' && str[i] != '-')

{

k \*= 0;

break;

}

}

}

if (s == "int")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '-')

{

k \*= 0;

break;

}

}

}

}

}

else

{

k \*= 0;

}

return k;

}

template <typename T>

T str\_to\_number(std::string str)

{

T a = 0; //само число

int index = 0, k = 1;

if (str[0] == '-')

{

k = -1;

str.erase(0, 1);

}

if (str.find('.') != -1)

{

index = str.find('.');

str.erase(str.find('.'), 1);

}

else

index = str.length();

for (int i = 0; i < str.length(); i++)

{

a = a \* 10 + (str[i] - '0');

}

return k \* (a / pow(10, (str.length() - index)));

}

Листинг various\_functions.cpp:

#include <iostream>// библиотека

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include <vector>

#include "Point.h"

using namespace std;

string char\_to\_string(char ch)

{

string s = {};

s += ch;

return s;

}

Point cursor\_position()

{

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

return { bi.dwCursorPosition.X, bi.dwCursorPosition.Y };

}

void set\_cursor\_position(Point A)

{

COORD coord;

coord.X = A.x;

coord.Y = A.y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

}

void erase\_past\_output(Point A)

{

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

set\_cursor\_position(A);

string s = "";

for (int i = 0; i < 80; i++)

s += ' ';

for (int i = 0; i < bi.dwCursorPosition.Y - A.y; i++)

cout << s << endl;

set\_cursor\_position(A);

}

Листинг main\_header.h:

#pragma once

void dop\_task\_1();

void lab\_1\_1();

void lab\_1\_2();

void lab\_1\_3();

void lab\_2\_1();

void lab\_2\_2();

void lab\_3\_1();

void lab\_3\_2();

void lab\_4\_1();

void lab\_4\_2();

void lab\_5\_1();

Результат.

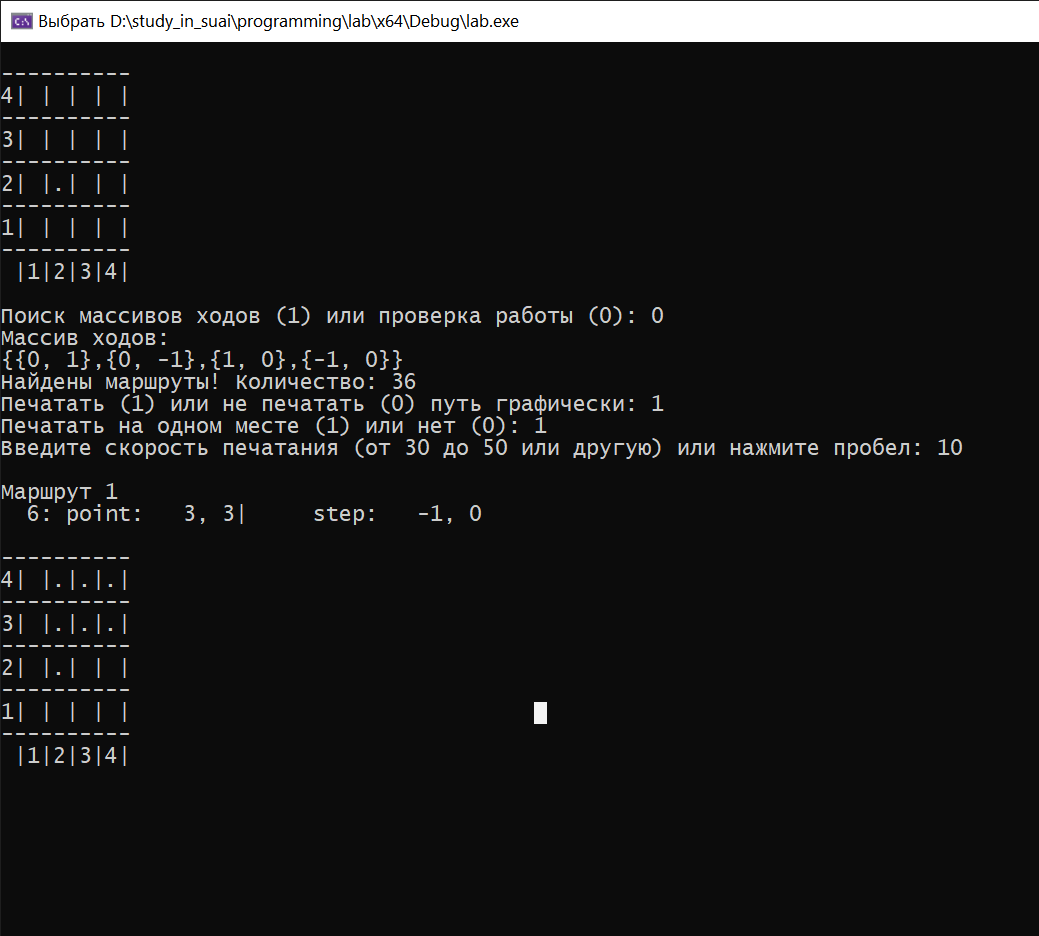


Рисунок 2. Тест 1

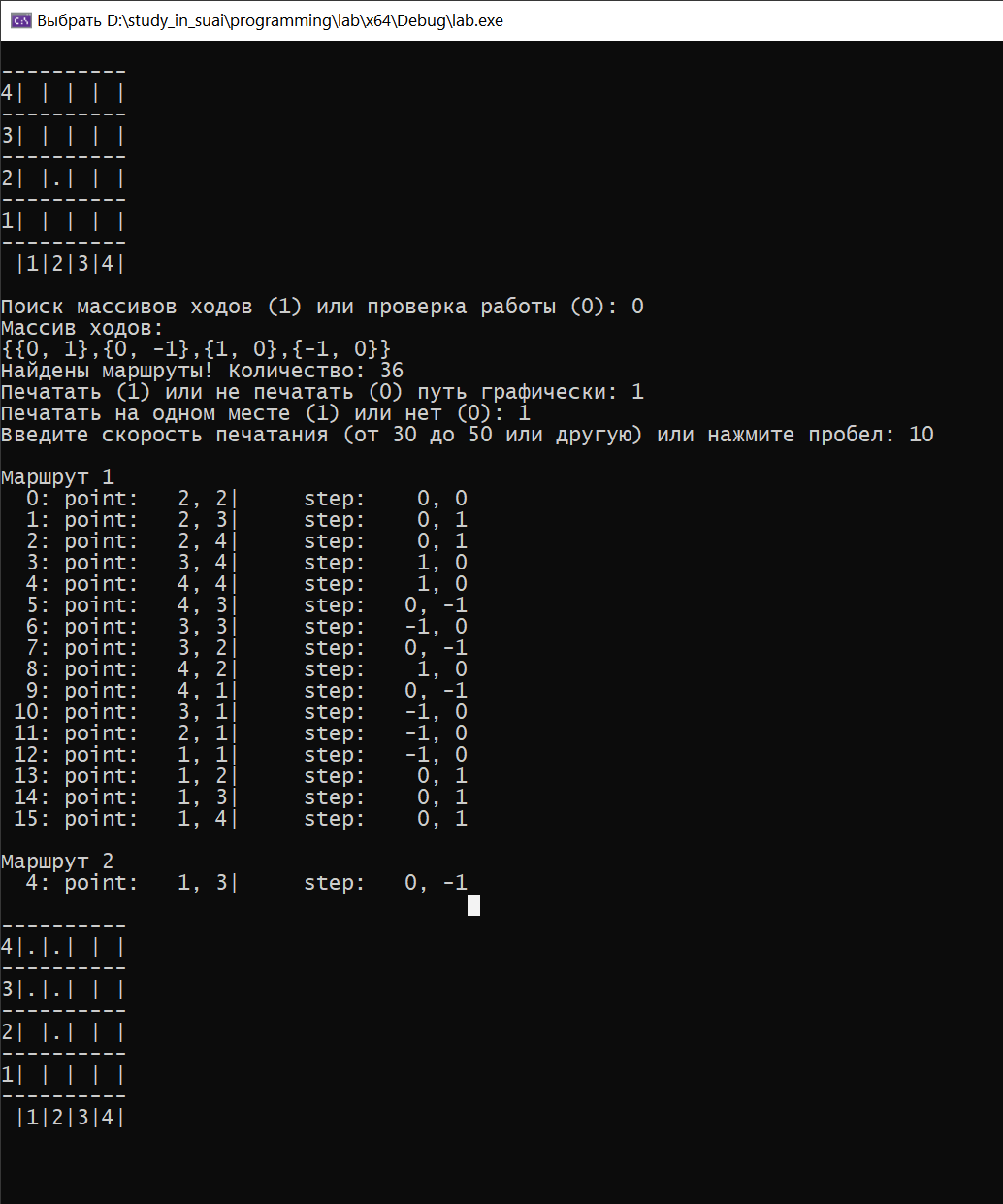


Рисунок 3. Тест 1

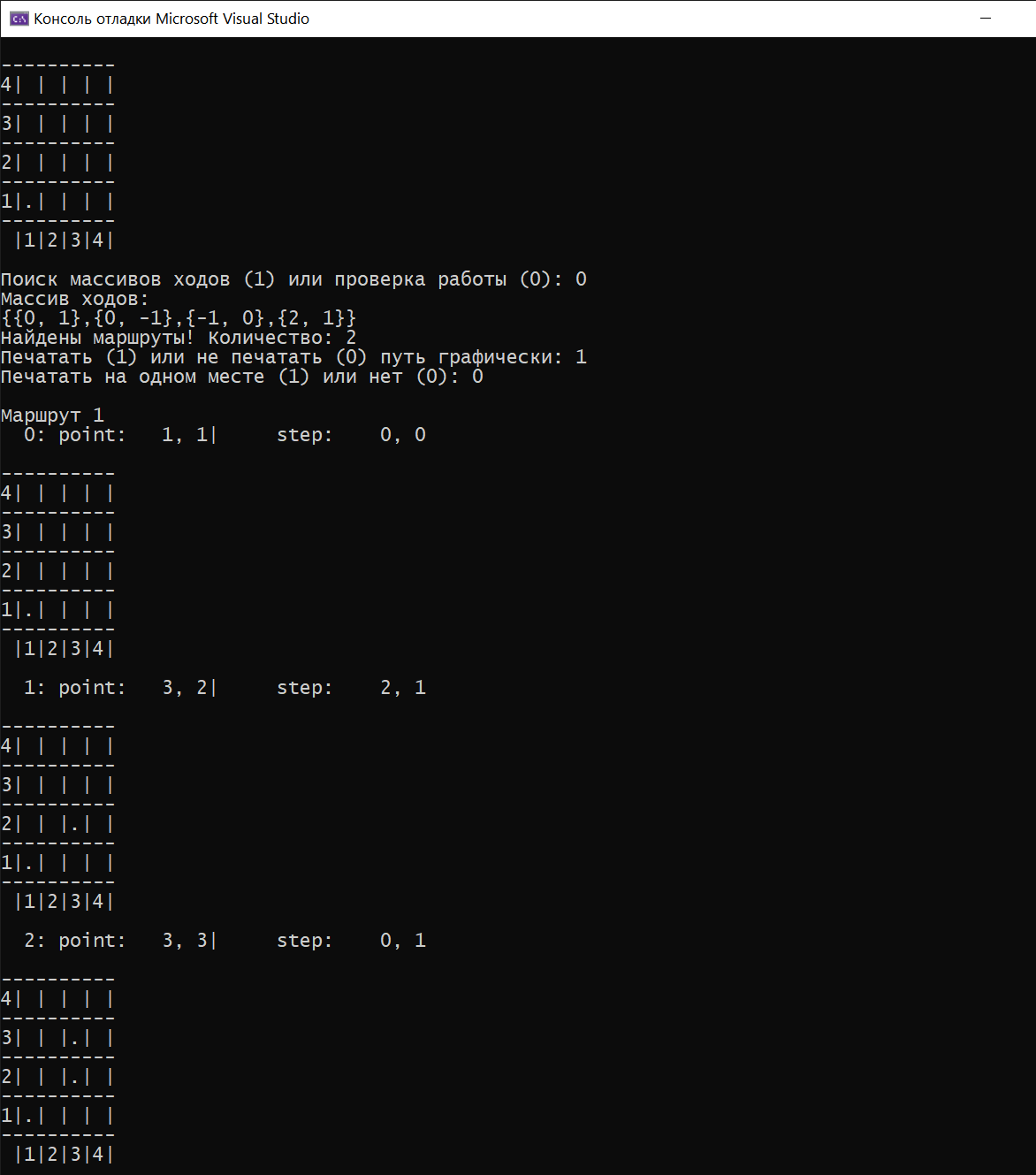


Рисунок 4. Тест 2

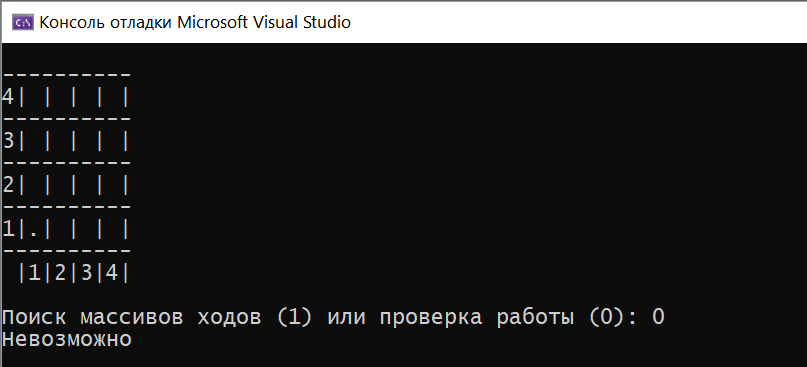


Рисунок 5. Тест 3

# 

Рисунок 6. Тест 4

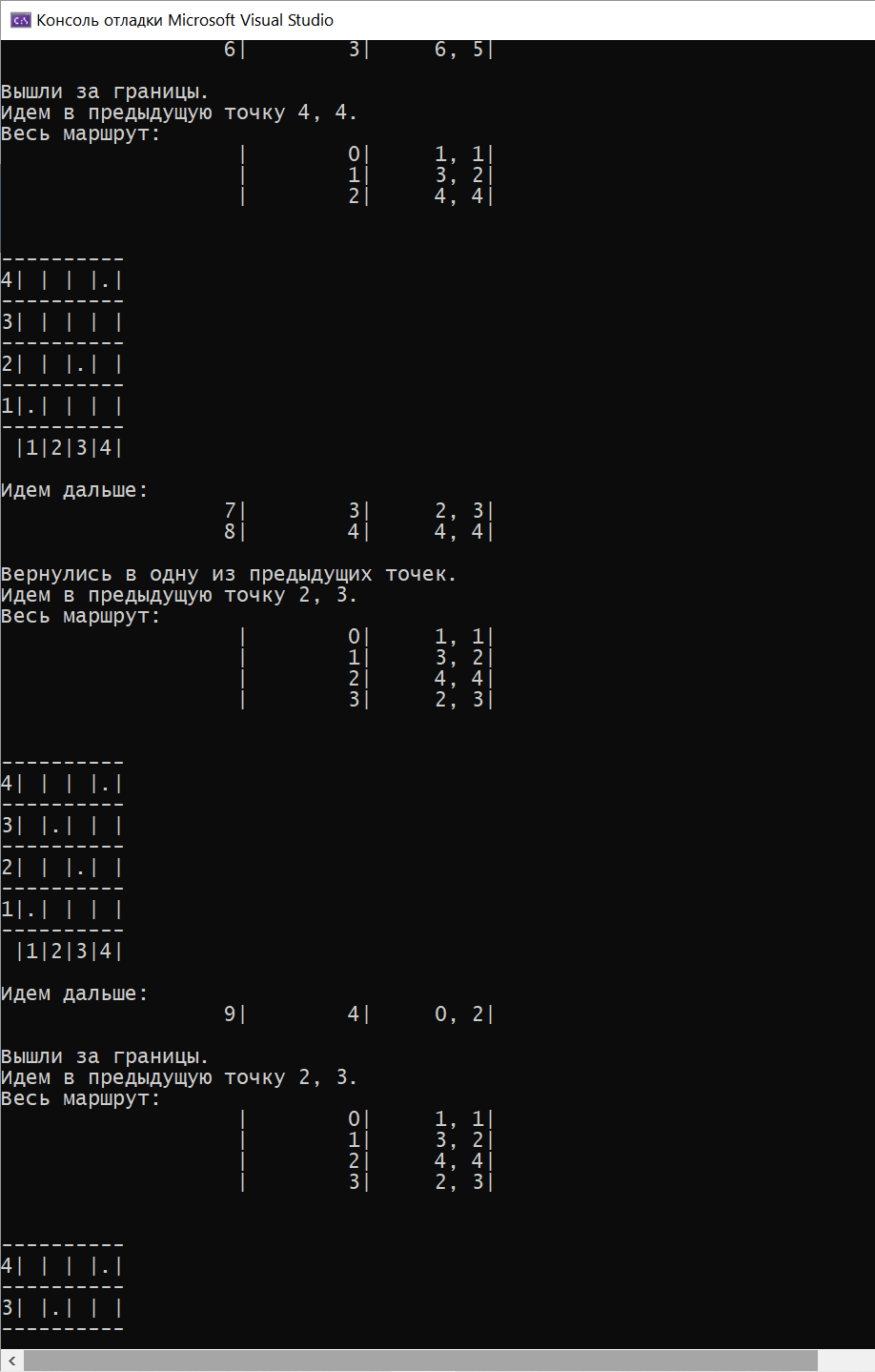


Рисунок 7. Тест 4

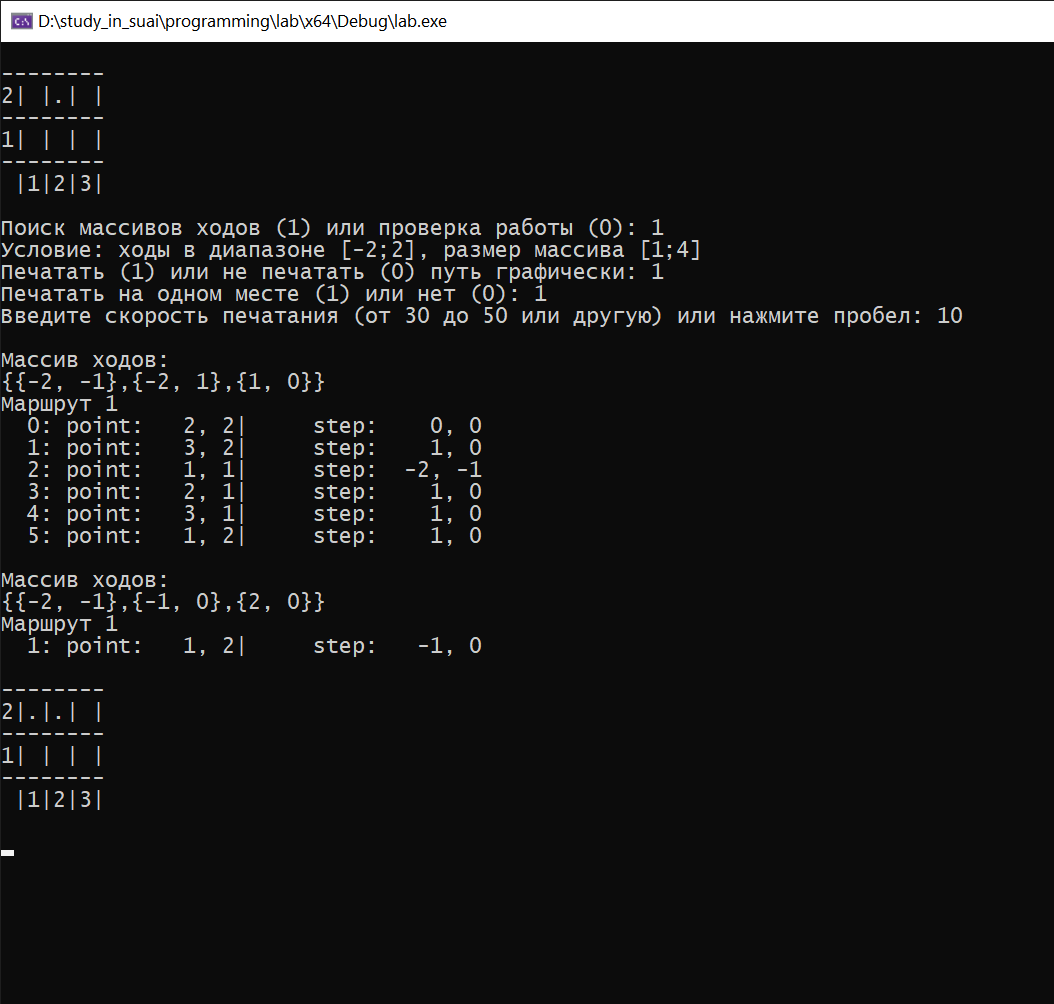


Рисунок 8. Тест 5

# Вывод.

Вводится массив ходов. Вызывается функция которая ищет возможные маршруты обхода всех клето данного поля, побывав в каждой клетке только один раз. Выводится количество маршрутов и каждый уникальный маршрут.