Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Тульский государственный университет»

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

**РЕКУРСИЯ С ВОЗВРАТОМ**

отчет о   
лабораторной работе №7

по дисциплине

*ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ*

***ВАРИАНТ 5***

|  |  |  |
| --- | --- | --- |
| Выполнила: | ст. гр. 230711 | Павлова В.С. |
| Проверил: | асс. каф. ИБ | Курбаков М.Ю. |

Тула, 2022 г.

# **ЦЕЛЬ И ЗАДАЧА РАБОТЫ**

**Цель:** ознакомление с понятием «рекурсия с возвратом», изучение принципов организации рекурсивно-возвратных алгоритмов.

**Задача:** в данной работе требуется написать программу, демонстрирующую использование изученных принципов.

# **ЗАДАНИЕ НА РАБОТУ**

**Задание №5. «Лабиринт».** Дан лабиринт, который задается двумерным массивом. Необходимо найти выход из начальной клетки. Элементы, по которым можно пройти, обозначаются «1», а по которым нельзя проходить – «0». Исходная позиция обозначается цифрой «2». Если проход возможен, то координаты каждого шага вывести на экран. Если пройти нельзя, то выдать соответствующее сообщение. Проход по диагонали невозможен.

# **СХЕМА ПРОГРАММЫ**

Схема алгоритма основной программы представлена на рисунке 1.

  
  
Рисунок 1 – Схема алгоритма основной программы

Схема алгоритма подпрограммы, реализующей рекурсию с возвратом и включающей в себя всю логику решения задачи №5 о поиске выхода из лабиринта, представлена на рисунке 2.



Рисунок 2 – Схема функции рекурсии с возвратом для решения задачи

# **ТЕКСТ ПРОГРАММЫ**

Текст программы на языке программирования С++ для решения задачи представлен в листинге 1.

## **Листинг 1. Текст программы**

#include <vector>

#include <iostream>

#include <fstream>

using namespace std;

bool isQuitExist(vector<vector<int>> &mazeMap, int curY, int curX, int endX, int endY)

## **Листинг 1. Текст программы (продолжение)**

{

if (mazeMap[endY][endX] == 2) //если выход достигнут

{

return true;

}

else

{

if (mazeMap[curY][curX] == 0) //если нет стены

{

mazeMap[curY][curX] = 2; //отметить клетку пройденной

if (curY < mazeMap.size() - 1 && //шаг вниз

isQuitExist(mazeMap, curY + 1, curX, endX, endY))

return true;

if (curX < mazeMap.size() - 1 && //шаг вправо

isQuitExist(mazeMap, curY, curX + 1, endX, endY))

return true;

if (curY > 0 && //шаг вверх

isQuitExist(mazeMap, curY - 1, curX, endX, endY))

return true;

if (curX > 0 && //шаг влево

isQuitExist(mazeMap, curY, curX - 1, endX, endY))

return true;

mazeMap[curY][curX] = 0; //отметить клетку непройденной

return false; //некуда идти

}

else //впереди стена

return false;

}

}

int main()

{

int n; //ввод информации о лабиринте

int startX, startY;

int endX, endY;

ifstream input("in.txt");

input >> n;

input >> startX >> startY;

input >> endX >> endY;

vector <vector<int>> maze(n, vector <int>(n, 0));

cout << "---> start is: [" << startX << "; " << startY << "]\n";

for (size\_t i = 0; i < n; i++)

{

for (size\_t j = 0; j < n; j++)

{

input >> maze[i][j];

cout << maze[i][j] << " ";

}

cout << "\n";

}

input.close();

if (isQuitExist(maze, startY, startX, endX, endY)) {

cout << "\n---> end is: [" << endX << "; " << endY << "]\n";

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

{

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

{

cout << maze[i][j] << " ";

}

cout << "\n";

## **Листинг 1. Текст программы (продолжение)**

}

}

else cout << "\n---> could not find the path :(\n";

return 0;

}

# **ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ**

Данная программа предназначена для демонстрации принципов использования рекурсии с возвратом при решении задачи №5. Пользователю предлагается ввести размерность лабиринта, координаты начальной точки, координаты точки выхода и сам лабиринт, где 0 – свободная клетка, а 1 – стена. Отсчёт координат ведётся с 0, а не с 1. После программа выполняет необходимые вычисления и выводит маршрут выхода, если такой существует, либо выводит сообщение о невозможности выхода.

# **ИНСТРУКЦИЯ ПРОГРАММИСТА**

Данная программа предназначена для решения задачи №5. Структуры данных, используемые в программе, приведены в таблице 1.

Таблица 1 – Структуры данных в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип (класс)** | **Предназначение** |
| startX, startY | int | Координаты точки входа в лабиринт |
| endX, endY | int | Координаты точки выхода из лабиринта |
| n | int | Размерность матрицы лабиринта |
| maze | vector <vector <int>> | Двумерный массив, хранящий состояние лабиринта |
| input | file | Файл для ввода информации |

В программе имеются следующие функции:

1. bool isQuitExist(vector<vector<int>> &mazeMap, int curY, int curX, int endX, int endY) – рекурсивная функция поиска выхода из лабиринта.

# **ДЕМОНСТРАЦИОННЫЙ ПРИМЕР**

Рассмотрим алгоритм работы на примере небольшого лабиринта размерностью 4. Пусть в нём существует путь из стартовой клетки с координатами [0; 2] в клетку выхода [2; 3]. Тогда он будет выглядеть следующим образом:



Рисунок 3 – Демонстрационный пример лабиринта

В каждой клетке лабиринта нам необходимо проверять, можем ли мы пойти вперёд, вверх, вниз, вправо или не можем пойти вообще. Так, например, находясь в клетке старта, обозначенной цифрой «2», мы можем сделать шаг только вперёд. Далее вперёд пойти уже нельзя, следовательно, необходимо проверять другие направления для движения, и так далее.

Программная реализация данной логики подразумевает рекурсивный вызов функции, содержащей в себе проверки возможности хода для каждого из направлений. Функция вызывается в каждой текущей точке маршрута.

Результат работы программы (рисунок 4) для данного набора входных данных будет аналогичен полученному теоретически. Цифры «2» обозначают пройденные клетки.

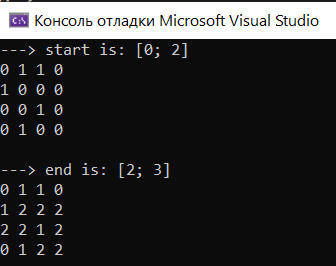


Рисунок 4 – Результат работы программы

# **ВЫВОДЫ**

В ходе данной лабораторной работы был изучен принцип работы рекурсии с возвратом. Соединение метода перебора с возвратом с рекурсией определяет специфический способ реализации рекурсивных вычислений, который называется возвратной рекурсией*.* Ценность метода возвратной рекурсии в том, что программы решения многих задач строятся по единой схеме и в том, что они компактны и тем самым просты для понимания и усвоения соответствующих идей.

Для демонстрации полученных знаний была написана программа, решающая задание №5 о поиске выхода из лабиринта. По результатам проверки можно сделать вывод о том, что программа работает корректно.