МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсия

Студент гр. 8381 _____ Гоголев Е.Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2019

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++.

Задание.

Имеется n городов, пронумерованных от 1 до n. Некоторые пары городов соединены дорогами. Определить, можно ли попасть по этим дорогам из одного заданного города в другой заданный город. Входная информация о дорогах задаётся в виде последовательности пар чисел i и j ($i < j$ и $i, j \in 1..n$), указывающих, что i-й и j-й города соединены дорогами.

Основные теоретические положения.

Рекурсия – определение части функции через саму себя, то есть это функция, которая вызывает саму себя, непосредственно (в своём теле) или косвенно (через другую функцию).

Всё решение сводится к решению основного или, как ещё его называют, базового случая. Существует такое понятие как шаг рекурсии или рекурсивный вызов. В случае, когда рекурсивная функция вызывается для решения сложной задачи (не базового случая) выполняется некоторое количество рекурсивных вызовов или шагов, с целью сведения задачи к более простой. И так до тех пор пока не получится базовое решение.

Выполнение работы.

Написание работы производилось на базе операционной системы Ubuntu Linux в среде разработки Geany. Сборка, отладка и тестирование производились с использованием компилятора G++.

Для реализации программы был разработан CLI с несколькими вариантами ввода (с консоли, из файла).

Дороги между городами сохраняются в списке World. Добавление дороги в список осуществляется с помощью функции push(), которая принимает список дорог, точку отправления и конечный пункт. Вспомогательная функция

print_step() выводит на экран шаг рекурсии, функция read_file() считывает данные из файла, имя которого пользователь вводит с консоли.

Решение задачи осуществляется функцией solve(), которая принимает на вход список дорог, начальную точку, конечную точку и счетчик для вывода шага рекурсии. Эта функция проходит по списку в поиске любой дороги из начальной точки и рекурсивно ищет путь в конечный пункт из всех городов, в которые можно попасть из текущего города.

Оценка эффективности алгоритма.

Алгоритм, реализованный в программе, имеет линейную зависимость от количества дорог, то есть сложность оценивается как O(n). Рассуждения отталкиваются от того, что функция solve обрабатывает одну дорогу ровно один раз, заменяя значение начального пункта на -1.

Количество занимаемой памяти также зависит от количества дорог, так как функция работает со списком, в который они изначально заносятся.

Выводы.

В ходе выполнения лабораторной работы была реализована программа, которая рекурсивным методом находит путь из одного города в другой. Получены практические навыки написания рекурсивных программ на C++.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <cstdlib>
#include <string>
#include <fstream>
#include <iostream>
#include <sstream>

using namespace std;

struct City {
    int id; //i
    int road; //j
    struct City* next_city;
};

struct World {
    City* head;
};

string read_file(string filename);
void push (World& world, int& i, int j);
bool solve (World& world, int a, int b, int& step_number);
void print_step(int step, int a, int b);

void push (World& world, int& i, int j) {
    City* current = new City();
    current->id = i;
    current->road = j;
    current->next_city = nullptr;
    if (!world.head) {
        world.head = current;
```

```
        }
        else {
            City* last = world.head;
            while (last->next_city) last = last->next_city;
            last->next_city = current;
        }
    }

    void print_step(int step, int a, int b) {
         cout << "\033[4;36mStep [" << step << "]: looking
for path from " << a << " to " << b << "\033[0m" << endl;
         return;
    }

    bool solve (World& world, int a, int b, int& step_number)
{
         if (a == b) return true;
         print_step(step_number, a, b);
        bool result = false;
        City* current = world.head;
        while (current != nullptr) {
              if (current->id == -1) {
                   current = current->next_city;
                   continue;
              }
            if (current->id == a) {
                   cout << "found " << current->id << "->" <<
current->road << endl;
                   if (current->road == b) return true;
                current->id = -1;
                  result |= solve(world, current->road, b, +
+step_number);
                if (result) {
                       return true;
```

```
                } else {
                    cout << "\033[4;31mpath from " << current->road << " to " << b << " not found\033[0m" << endl;
                }
            }
            current = current->next_city;
        }
        return false;
    }

    void read_file(string filename, World& world, int& start, int& dest) {
        ifstream input_file(filename);
        if (!input_file.is_open()) {
            cerr << "Error opening file\n" << endl;
            return;
        }
        int n;
        input_file >> n;
        for(int x1,x2, i = 0; i < n; i++)
                input_file >> x1 >> x2, push(world, x1, x2);
        input_file.close();
        cout << "Find path from ";
         cin >> start;
         cout << " to ";
         cin >> dest;
         cout << endl;
        return;
    }

    int main (int argc, char** argv) {
        World world;
        world.head = nullptr;
```

```
          int choise;
          cout << "[0] Read from file\n[1] Read from console\
n";
          cin >> choise;
          if (choise != 0 && choise != 1) {
               cout << "Incorrect input\n";
               exit(1);
          }
          int start, destination;
          if (choise == 1) {
               int n;
               cout << "How many roads?" << endl;
               cin >> n;
               cout << "input roads separated by spaces" <<
endl;
               for(int x1,x2, i = 0; i < n; i++)
                    cin >> x1 >> x2, push(world, x1, x2);
               cout << "Find path from ";
               cin >> start;
               cout << " to ";
               cin >> destination;
               cout << endl;
          }
          else {
               cout << "Введите имя файла" << endl;
               string result;
               cin >> result;
               cout << "Считывание из " << result << endl;
               read_file(result, world, start, destination);
          }
          int counter = 0;
          int result;
          result = solve(world, start, destination, counter);
          cout << "result: " << result << endl;
          return 0; }
```