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. 9304 _____ Боблаков Д.С.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Изучить понятие рекурсии и методы её применения.

Научиться грамотно реализовывать программы с использованием рекурсии в языке C++.

Задание.

Задание №3

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 1..n), указывающих, что i-й и j-й города соединены дорогами.

Выполнение работы.

Программа считывает число городов *n*. Затем она считывает два числа *from* и *to*, соответствующих городам, которые необходимо проверить на связанность. Далее в связи со спецификой нумерации массива в языке C++ уменьшаем *from* и *to* на единицу. После этого создаем двумерный массив (далее таблица) размером *n***n* и инициализируем его нулями. Эта таблица будет хранить «карту городов». Затем считываем пары значений *i* и *j*, соответствующие связанным парам городов. Эти пары конвертируются в единицы в двух соответствующих местах таблицы. При вводе значений 0 0 программа перестает считывать данные. Далее создается массив *bool* watched* с логическими переменными размером n, значения которого будут отвечать за предыдущее нахождение в каждом городе. Инициализирован этот массив значениями *false*.

Затем вызывается функция *recSearch(),* которая принимает номера искомых городов, указатель на таблицу и её размер, массив *bool* watched*. Эта функция, начиная с первого искомого города проходится по всем возможным городам, доступным для него, и ищет другой искомый город. Если такой не

находится, программа переходит в ранее не просмотренный город, а затем рекурсивно вызывается функция *recSearch()* с измененным первым параметром на новый город. При нахождении связи между двумя искомыми городами функция возвращает *true*, в остальных случаях *false*.

После завершении рекурсивной функции в зависимости от результата ее работы выводится *false* или *true*. В конце работы программы очищается память из под двух массивов.

Разработанный программный код см. в приложении А.

Тестирование.

Для проведения тестирования был написан bash-скрипт ./tests_script. Скрипт запускает программу с входными данными из папки ./Tests/tests и сравнивает полученные результаты с готовыми ответами из ./Tests/true_results. Для каждого теста выводится сообщение Test# passed или Test# failed. Полученные в ходе работы, файлы с выходными данными удаляются.

Результаты тестировния см. в приложении Б.

Выводы.

Было изучено понятие рекурсии, а также методы её применения.

Была реализована программа на языке C++ с использованием рекурсивного алгоритма.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include "recSearch.h"


int main() {
    unsigned int from, to,n,i,j;
    std::cin>>n;
    std::cin>>from>>to;
    from--;
    to--;
    int** arr=new int*[n];
    for (int k = 0; k < n; ++k) {
        arr[k]=new int[n];
    }
    for (int k = 0; k < n; ++k) {
        for (int l = 0; l < n; ++l) {
            arr[k][l]=0;
        }
    }
    do {
        std::cin>>i;
        std::cin>>j;
        if(i>n || j>n ||((i>=j))&& (i!=0)){
            std::cout<<"Error: incorrect value\n";
            return EXIT_FAILURE;
        }
        if(i==0 ||j==0){
            break;
        }
        arr[i-1][j-1]=1;
        arr[j-1][i-1]=1;
    }while(i!=0 || j!=0);

    bool* watched = new bool[n];
    for (int k = 0; k < n; ++k) {
        watched[k]=false;
    }
    bool isTrue=recSearch(from,to,n,arr,watched);
    if (isTrue){
        std::cout<<"true";
    }    else std::cout<<"false";

    for (int k = 0; k < n; ++k) {
        delete arr[k];
    }
    delete[] arr;
    delete[] watched;
    return 0;
}
```

Название файла: recSearch.cpp

```
#include "recSearch.h"
```

```
bool recSearch(unsigned int from, unsigned int to,unsigned int n,int**
arr,bool* watched){
    unsigned int now=from;
    for (int next = 0; next < n; ++next) {
        if((arr[now][next]==1) && next==to){
            return true;
        }
        if((arr[now][next]==1) && !watched[next]){
            watched[now]=true;

            if(recSearch(next, to, n, arr, watched))
                return true;
        }
    }
    return false;
}
```

Название файла: recSearch.h

```
#ifndef ADS_1_RECSEARCH_H
#define ADS_1_RECSEARCH_H
#include <iostream>
bool recSearch( unsigned int from, unsigned int to,unsigned n,int**
arr,bool* watched);
#endif
```

Название файла: Makefile

```
lab1:
        g++ source/main.cpp source/recSearch.cpp -o lab1

tests: lab1
        ./tests_script
```

Название файла: ./tests_script

```
#! /bin/bash

echo Running tests...
for n in {1..10}
do
    ./lab1 < "./Tests/tests/test$n.txt" > "./Tests/out/out$n.txt"
    if cmp "./Tests/out/out$n.txt" "./Tests/true_results/true_out$n.txt"
> /dev/null; then
            echo "Test$n passed"
    else
            echo "Test$n failed"
    fi
done
rm ./Tests/out/out*
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ ПРОГРАММЫ

Результаты тестирования представлены в табл.1

Для экономии места символ ‘\n’ заменен на ‘/’.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	11/1 11/1 2/1 3/3 4/3 5/5 6/5 7/7 8/7 9/9 10/9 11/0 0	true	Дерево с ветвлением вправо
2.	4/1 4/1 2/1 3/2 3/3 4/0 0	true	Цикл с ответвлением
3.	5/1 5/2 3/3 4/4 5/0 0	false	Цепь с оторванной вершиной
4.	5/3 5/1 2/1 3/1 4/1 5/2 3/2 4/2 5/3 4/3 5/4 5/0 0	true	Сеть со связью узлов каждый с каждым
5.	5/1 5/1 3/2 4/2 5/1 4/3 5/0 0	true	5-конечная звезда
6.	6/1 4/1 2/1 3/2 3/4 5/4 6/5 6/0 0	false	Два независимых цикла
7.	5/1 4/1 2/2 3/4 5/0 0	false	Две независимые цепи
8.	8/1 8/1 2/1 3/2 3/4 5/5 6/4 6/7 8/0 0	false	Два независимых цикла и одна цепь
9.	6/1 6/1 2/1 3/2 3/3 4/4 5/4 6/5 6/0 0	true	Два цикла связаны одной дорогой
10.	7/1 6/1 2/1 3/2 4/2 5/3 6/3 7/0 0	true	Двоичное дерево