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”

Студентка гр. 7381 _____ Алясова А.Н.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2018

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

Пояснение задания.

На вход программе подается сначала количество городов, потом количество последовательностей пар чисел (дорог), затем сами последовательности пар чисел i и j ($i < j$ и $i, j \in 1..n$), которые указывают, что i-й и j-й города соединены дорогами, в конце подается два города, между которыми или существует путь или нет.

Описание алгоритма.

Для определения существования пути между заданными городами мы создаем матрицу смежности и работаем по ней. Предположим, что нам нужно найти путь из города 1, начальный город, в город 2, конечный. Если прямого пути из города 1 в город 2 не существует, мы берем город 1 и просматриваем города, в которые можно попасть из этого города. Когда выбирали город, называем его текущим и рекурсивно идем дальше, принимая текущий город за начальный, а конечный остается прежним. Как только появиться прямой путь из начального города в конечный, завершаем программу и выводим YES. Если же, просмотрев все возможные пути, прямого пути с конечным так и не получилось обнаружить, завершаем программу и выводим NO.

Описание функций.

- int find_way(int deep, int first, int second, bool **matr, int city_count); - рекурсивная функция, которая на вход принимает следующие аргументы:

  1) int deep – требуется для оформления вывода, контролирует количество отступов;

  2) int first – первый город;

  3) int second – второй город;

  4) bool **matr – двумерный массив булевого типа, представляет собой матрицу смежности, которая содержит в себе информацию о известных нам дорогах;

  5) int city_count – количество городов;

- int main(); – главная функция, в ней содержится основной код программы, функция не принимает аргументов.

Тестирование.

На вход программе подается количество городов, количество дорог, дороги, два города, между которыми нужно определить существует ли путь или нет, на выход – информация о проверке со всеми рекурсивными вызовами и краткий результат: YES, если путь существует и NO в противном случае. Приведем примеры в табл. 1. Для автоматизации тестирования, был написан bash-скрипт.

Таблица 1 - Тестирование

Входные данные	Результат
5 4 1 2	YES

Продолжение таблицы 1

2 3 2 4 4 5 1 5	YES
8 7 1 3 1 4 2 3 5 8 5 6 6 7 7 8 4 5	NO
1 0 1 1	YES
4 2 1 2 1 3 1 4	NO Matr[1][2] = true => matr[1][2]=false, делаем это для того,чтобы избежать зацикливания, идем дальше => matr[2][3 или 4] = false => возвращаемся в 1, смотрим matr[1][3] = true => меняем matr[1][3] = false, дальше рассматриваем matr[3][2 или 4] = false => дороги не существует => return 0;

Вывод.

В процессе выполнения лабораторной работы были получены навыки по написанию рекурсивных функций, bash-скриптов и автоматизации тестирования. Работа была написана на языке программирования C++.

ПРИЛОЖЕНИЕ А

Содержимое файла main.cpp

```
#include <iostream>
#include <string>
#include <string.h>

using namespace std;

int find_way(int deep, int first, int second, bool **matr, int
city_count) { //функция для нахождения пути
	for(int i = 0; i < deep; i++)
		cout << "\t";
	cout << "->rec with " << first << " " << second << endl;
	if(matr[first-1][second-1] || first == second) {
		for(int i = 0; i < deep; i++)
			cout << "\t";
			return 1;
	}
	for(int i = 0; i < city_count; i++) {
		if(matr[first-1][i-1]) {
			matr[first-1][i-1] = false;
			if(find_way(deep+1, i, second, matr, city_count)) {
				for(int i = 0; i < deep; i++)
					cout << "\t";
				cout << "<-rec" << endl;
				return 1;
			}
		}
	}
	for(int i = 0; i < deep; i++)
		cout << "\t";
	cout << "<-rec" << endl;

	return 0;
```

```
}

void proverka(int &a, char* stroka){ //проверка на дурака
	bool test = true;
	do {
		cout << stroka << endl;
		cin >> a;
		if ( !(test = cin.good()) )
			cout << "Вы ввели не число!" << endl;
		cin.clear() ;
		cin.ignore(1) ;
	} while(!test);
return;
}

int main(){

	int city_count;				//ввод кол-ва городов
	char str[] = "Введите количество городов:";
	proverka(city_count, str);

	if (city_count == 0){			//проверка на пустоту
списка
			cout << "Пустой список";
			 return 0;
	}

	bool **matr = new bool *[city_count];
	for (int i = 0; i < city_count; i++)
		matr[i] = new bool [city_count];

	int road_count;
	char str1[] = "Введите количество дорог: ";
	proverka(road_count, str1);
```

```
	cout << "Введите дороги:\n";
	for (int k = 0; k < road_count; k++) {
		int i, j;
		cin >> i >> j;

		try {
			if (i > city_count || i < 1 || j > city_count || j <
1)
				throw "Не существует указаного города.";
		}
		catch (const char * string){
			cout << string << endl << "Повторите ввод: ";
			k--;
			continue;
		}
			matr[i-1][j-1] = true;
			matr[j-1][i-1] = true;
	}

	cout << "Введите начальный и конечный города: " << endl    int
start, finish;
	cin >> start >> finish;

	if (find_way(0, start, finish, matr, city_count))
		cout << "YES" << endl;
	else
		cout << "NO" << endl;

	for(int i = 0 ; i < city_count; i++)       //очищение памяти
		free(matr[i]);
	free(matr);
	return 0;
}
```

ПРИЛОЖЕНИЕ В

Содержимое файла Exec.sh

```
g++ ./Sourse/lb1.cpp -o Lab1
echo -e '_______\nTest 1:'
cat ./Tests/Test1.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test1.txt
echo -e ''
echo -e '_______\nTest 2:'
cat ./Tests/Test2.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test2.txt
echo -e ''
echo -e '_______\nTest 3:'
cat ./Tests/Test3.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test3.txt
echo -e ''
echo -e '_______\nTest 4:'
cat ./Tests/Test4.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test4.txt
echo -e ''
echo -e '_______\nTest 5:'
cat ./Tests/Test5.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test5.txt
echo -e ''
echo -e '_______\nTest 6:'
cat ./Tests/Test6.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test6.txt
echo -e ''
echo -e '_______\nTest 7:'
```

```
cat ./Tests/Test7.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test7.txt
echo -e ''
echo -e '_______\nTest 8:'
cat ./Tests/Test8.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test8.txt

echo -e ''
echo -e '_______\nTest 10:'
cat ./Tests/Test10.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test10.txt
echo -e ''
echo -e '_______\nTest 11:'
cat ./Tests/Test11.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test11.txt
echo -e ''
echo -e '_______\nTest 12:'
cat ./Tests/Test12.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test12.txt
echo -e ''
echo -e '_______\nTest 13:'
cat ./Tests/Test13.txt
echo -e '_______\nTesting:\n'
./Lab1 < ./Tests/Test13.txt
```