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 _____ Цаплин И.В.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Научиться использовать рекурсию на примере языка C++.

Задание.

Вариант 2.

Задано конечное множество имен жителей некоторого города, причем для каждого из жителей перечислены имена его детей. Жители X и Y называются родственниками, если (а) либо X – ребенок Y, (б) либо Y – ребенок X, (в) либо существует некоторый Z, такой, что X является родственником Z, а Z является родственником Y. Перечислить все пары жителей города, которые являются родственниками.

Выполнение работы.

Для хранения информации о жителе города реализован класс Citizen, который хранит две строки: имя и список детей данного жителя. Для установления того, являются ли два жителя родственниками реализован рекурсивный метод are_relatives(), который принимает ссылку на другого жителя, указатель на массив объектов класса Citizen, в котором находятся оба жителя, длину данного массива и глубину рекурсии. Метод проверяет является ли объект родителем или ребёнком другого объекта, осуществляя поиск подстроки имени в списке детей с помощью метода find, и возвращает true, если это так. В противном случае он вызывает этот же метод для поиска в массиве объекта, который является родственником для первого и второго объекта одновременно, увеличивая при этом значение глубины рекурсии. Если значение глубины рекурсии достигает значения длины массива, то два объекта не являются родственниками, метод возвращает false.

Программа принимает на вход число жителей города. Затем вводятся имена жителей и списки их детей в формате <Имя жителя> (<Список детей>). Если у жителя нет детей, скобки остаются пустыми. Программа выделяет память под массив объектов класса Citizen, заполняет его входными данными. Затем для каждого объекта вызывает метод are_relatives с каждым

последующим объектом из массива. Если метод are_relatives возвращает true, программа выводит имена жителей в формате <Имя1> - <Имя2>.

Разработанный программный код см. в приложении А.

Тестирование.

Для проведения тестирования был написан bash-скрипт tests_script. Скрипт запускает программу с определёнными входными данными и сравнивает полученные результаты с готовыми ответами. Для каждого теста выводится сообщение TestX passed или TestX failed. Полученные в ходе работы, файлы с выходными данными удаляются.

Результаты тестировния см. в приложении Б.

Выводы.

Научились использовать рекурсию для решения поставленных задач на примере языка программирования C++.

Была написана программа, осуществляющая рекурсивный поиск родственников. Проведено тестирование работы программы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab1.cpp

```
#include <iostream>
#include <memory>

class Citizen{
        public:
        std::string name;
        std::string kids_list;
        Citizen(std::string name = "", std::string  kids_list =
""){
                this->name = name;
                this->kids_list = kids_list;
        }

        bool are_relatives(Citizen& B, std::shared_ptr<Citizen[]>
array,int length, int deep){
                if(name == B.name){
                        return false;
                }
                if((B.kids_list.find(name) != std::string::npos)
|| (kids_list.find(B.name) != std::string::npos)){
                        return true;
                }
                if(deep == length){
                        return false;
                }
                for(int i = 0; i < length; i++){
                        if((this->are_relatives(array[i], array,
length, deep+1)) && (array[i].are_relatives(B, array, length,
deep+1))){
                                                return true;
                                        }
```

```
                }
                return false;
        }
};

int main(){
        int number;
        std::cin >> number;
        auto array = std::shared_ptr<Citizen[]>(new
Citizen[number]);
        for( int i = 0; i < number; i++ ) {
                std::cin >> array[i].name;
                getline(std::cin,array[i].kids_list,'\n');
        }
        for( int i = 0; i < number; i++ ) {
                for( int j = i; j < number; j++ ) {
                        if(array[i].are_relatives( array[j],
array, number, 0)){
                                std::cout << array[i].name << " -
" << array[j].name << "\n";
                        }
                }
        }
        return 0;
}
```

Название файла: tests_script

```
#! /bin/bash


echo Running tests...
for n in {1..5}
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*
```

Название файла: Makefile

```
lab1: Source/lab1.cpp
        g++ Source/lab1.cpp -o lab1

run_tests: lab1
        ./tests_script
```

ПРИЛОЖЕНИЕ Б
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

№	Входные данные	Выходные данные	Комментарии
1	5 Parent1 (Son1, Son2) Son2 () Son1 (Grandson1) Grandson1 (Grandgrandson1) Grandgrandson1 ()	Parent1 - Son2 Parent1 - Son1 Parent1 - Grandson1 Parent1 - Grandgrandson1 Son2 - Son1 Son2 - Grandson1 Son2 - Grandgrandson1 Son1 - Grandson1 Son1 - Grandgrandson1 Grandson1 - Grandgrandson1	Одна разветвлённая семья, каждый житель родственник другому.
2	5 Name1 (Name2) Name2 (Name3) Name3 (Name4) Name4 (Name5) Name5 ()	Name1 - Name2 Name1 - Name3 Name1 - Name4 Name1 - Name5 Name2 - Name3 Name2 - Name4 Name2 - Name5 Name3 - Name4 Name3 - Name5 Name4 - Name5	Одна линейная семья, каждый житель родственник другому.
3	6 Name1 (Name2, Name3, Name4) NoRelatives () Name2 ()	Name1 - Name2 Name1 - Name3 Name1 - Name4 Name1 - Name0 Name2 - Name3	Одна линейная семья, каждый житель

	Name3 () Name4 () Name0 (Name1)	Name2 - Name4 Name2 - Name0 Name3 - Name4 Name3 - Name0 Name4 - Name0	родственник другому. Присутствует житель без родственнико в, его в выводе нет.
4	5 Family1_Name1 (Family1_Name2, Family1_Name3) Family1_Name2 (Family1_Name4) Family1_Name3 () Family2_Name1 (Family2_Name2) Family2_Name2 ()	Family1_Name1 - Family1_Name2 Family1_Name1 - Family1_Name3 Family1_Name2 - Family1_Name3 Family2_Name1 - Family2_Name2	Две отдельные семьи. Жители из разных семей — не родственники.
5	8 NoRelatives1 () NoRelatives2 () NoRelatives3 () NoRelatives4 () NoRelatives5 () NoRelatives6 () NoRelatives7 () NoRelatives8 ()		Город без родственнико в. Пустой вывод программы.