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

ОТЧЕТ

по практической работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия

Студент гр. 9382 _____ Бочаров Г.С.

Преподаватель _____ Еременко А.А.

Санкт-Петербург
2020

Цель работы.

Понять основные принципы работы рекурсии. Освоить написание алгоритмов с использованием рекурсивных функций на языке 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.

Алгоритм.

1) На основе множества имен создается матрица родства 1-го порядка (матрица A).

Иными словами, если элементты множества имен с номерами i и j состовляют пару родитель → ребенок или ребенок → родитель, ячейкам матрицы с координатами A[i][j] и A[j][i], присваиваются единицы, остальные элементы матрицы равны нулю.

2) Далее элементы множества попарно проверяюстя на родство, следующим образом :

1. Берем номера данных двух элементов множества имен (i и j).
2. Проверяем равняется ли A[i][j] единице, если да, то элементы состоят в родстве и алгоритм завершается. Если нет, то проверяем по очереди на родство с элементом под номером j всех родственников элемента с номером i (при этом их родство с элементом i не учитывается чтобы избежать цикла).

3) Если входе работы алгоритма общий родственник не найден, то элементы множества не состоят в родстве.

Выполнение работы.

1. template<typename StreamT>
std::vector<std::string> getNames(StreamT &in) - функция считывает имена из файла или из консоли и возвращает массив этих имен. На вход подается поток ввода.

2. template<typename StreamT>

void getRelationsMatrix(StreamT &in, std::vector<std::vector<bool>> &relationTable) — функция считывает из входного потока данные о родстве и на их основе заполняет матрицу смежности.

3. bool isFamily(int a, int b, std::vector<std::vector<bool>> &arr) — функция принимает номера двух элементов из массива имен и матрицу смежности. Далее проверяет их на родство по алгоритму описанному выше.

4. isFamilyWithDetails — работает как функция bool isFamily с той лишь разницей, что выводит параметры вхождения в функцию.

5. printNames(std::vector<std::string> names) — функция выводит имена с их номерами в массиве.

6. printRelationsMatrix(std::vector<std::vector<bool>> &arr) — функция выводит матрицу смежности.

7. printPairs(std::vector<std::string> names, std::vector<std::vector<bool>> &arr) — функция принимает на вход массив имен и таблицу смежности. Далее попарно проверяет элементы массива имен на родство. В случае родства 2-я элементов выводит их.

Тестирование.

Таблица 1

Номера тестов	Ввод данных	Вывод данных
1.	0 1 2 3 4 5 6 0 1 2 2 3 4 3 5 6	(подробный вывод) вывод матрицы смежности 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 вывод пар isFamily(0 and 1) (Родственники) 0 <--> 1 isFamily(0 and 2) (Родственники) 0 <--> 2 isFamily(0 and 3) - isFamily(1 and 3) - isFamily(2 and 3) (Родственники) 0 <--> 3 isFamily(0 and 4) - isFamily(1 and 4) -

		isFamily(2 and 4) - isFamily(3 and 4) (Родственники) 0 <--> 4 isFamily(0 and 5) - isFamily(1 and 5) - isFamily(2 and 5) - isFamily(3 and 5) - isFamily(4 and 5) - (Не родственники) isFamily(0 and 6) - isFamily(1 and 6) - isFamily(2 and 6) - isFamily(3 and 6) - isFamily(4 and 6) - (Не родственники) isFamily(1 and 2) - isFamily(0 and 2) (Родственники) 1 <--> 2 isFamily(1 and 3) - isFamily(0 and 3) - isFamily(2 and 3) (Родственники) 1 <--> 3 isFamily(1 and 4) - isFamily(0 and 4) - isFamily(2 and 4) - isFamily(3 and 4) (Родственники) 1 <--> 4 isFamily(1 and 5) - isFamily(0 and 5) - isFamily(2 and 5) - isFamily(3 and 5) - isFamily(4 and 5) - (Не родственники) isFamily(1 and 6) - isFamily(0 and 6) - isFamily(2 and 6) - isFamily(3 and 6) - isFamily(4 and 6) - (Не родственники) isFamily(2 and 3) (Родственники) 2 <--> 3 isFamily(2 and 4) - isFamily(0 and 4) - isFamily(1 and 4) - isFamily(3 and 4) (Родственники) 2 <--> 4 isFamily(2 and 5) - isFamily(0 and 5) - isFamily(1 and 5) - isFamily(3 and 5) - isFamily(4 and 5) - (Не родственники) isFamily(2 and 6) - isFamily(0 and 6) -

		isFamily(1 and 6) - isFamily(3 and 6) - isFamily(4 and 6) - (Не родственники) isFamily(3 and 4) (Родственники) 3 <--> 4 isFamily(3 and 5) - isFamily(2 and 5) - isFamily(0 and 5) - isFamily(1 and 5) - isFamily(4 and 5) - (Не родственники) isFamily(3 and 6) - isFamily(2 and 6) - isFamily(0 and 6) - isFamily(1 and 6) - isFamily(4 and 6) - (Не родственники) isFamily(4 and 5) - isFamily(3 and 5) - isFamily(2 and 5) - isFamily(0 and 5) - isFamily(1 and 5) - (Не родственники) isFamily(4 and 6) - isFamily(3 and 6) - isFamily(2 and 6) - isFamily(0 and 6) - isFamily(1 and 6) - (Не родственники) isFamily(5 and 6) (Родственники) 5 <--> 6
2.	Anna Pit Vova Nik Ted Pudge Jerry N1 N2 N3 N4 N6 N666 0 3 2 1 5 4 6 6 9 7 7 1 8 10 11 12 11	вывод пар Anna <--> Pit Anna <--> Vova Anna <--> Nik Anna <--> Ted Anna <--> Pudge Anna <--> Jerry Anna <--> N1 Anna <--> N2 Anna <--> N3 Pit <--> Vova Pit <--> Nik Pit <--> Ted Pit <--> Pudge Pit <--> Jerry Pit <--> N1 Pit <--> N2 Pit <--> N3 Vova <--> Nik Vova <--> Ted Vova <--> Pudge Vova <--> Jerry

		Vova <--> N1 Vova <--> N2 Vova <--> N3 Nik <--> Ted Nik <--> Pudge Nik <--> Jerry Nik <--> N1 Nik <--> N2 Nik <--> N3 Ted <--> Pudge Ted <--> Jerry Ted <--> N1 Ted <--> N2 Ted <--> N3 Pudge <--> Jerry Pudge <--> N1 Pudge <--> N2 Pudge <--> N3 Jerry <--> N1 Jerry <--> N2 Jerry <--> N3 N1 <--> N2 N1 <--> N3 N2 <--> N3 N4 <--> N6 N4 <--> N666 N6 <--> N666
3.	1 11 111 1111 11111 2 2222 22222 22 222 3 33 0 5 6 2 2 4 7 9 0	вывод пар 1 <--> 111 1 <--> 11111 1 <--> 2 1 <--> 2222 1 <--> 22222 1 <--> 222 111 <--> 11111 111 <--> 2 111 <--> 2222 111 <--> 22222 111 <--> 222 11111 <--> 2 11111 <--> 2222 11111 <--> 22222 11111 <--> 222 2 <--> 2222 2 <--> 22222 2 <--> 222 2222 <--> 22222 2222 <--> 222 22222 <--> 222
4.	\n (пустой тест)	Список имен пуст!
5.	0	ошибка: неверный номер

	1 2 3 6 7	родителя!
6.	0 1 2 3 0 10	ошибка: неверные номер ребенка!
7.	0 1 2 3 4 0 1 2 2 4	(подробный вывод) вывод пар isFamily(0 and 1) (Родственники) 0 <--> 1 isFamily(0 and 2) (Родственники) 0 <--> 2 isFamily(0 and 3) - isFamily(1 and 3) - isFamily(2 and 3) - isFamily(4 and 3) - (Не родственники) isFamily(0 and 4) - isFamily(1 and 4) - isFamily(2 and 4) (Родственники) 0 <--> 4 isFamily(1 and 2) - isFamily(0 and 2) (Родственники) 1 <--> 2 isFamily(1 and 3) - isFamily(0 and 3) - isFamily(2 and 3) - isFamily(4 and 3) - (Не родственники) isFamily(1 and 4) - isFamily(0 and 4) - isFamily(2 and 4) (Родственники) 1 <--> 4 isFamily(2 and 3) - isFamily(0 and 3) -

		isFamily(1 and 3) - isFamily(4 and 3) - (Не родственники) isFamily(2 and 4) (Родственники) 2 <--> 4 isFamily(3 and 4) - (Не родственники)
8.	Ввод из файла Введите имя файла: Pffff	Файл не найден!

Выводы.

В ходе выполнения данной лабораторной работы был написан рекурсивный алгоритм проверки элементов множества на родство.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

int spaceCount = 0; // Счетчик количества пробелов, для вывода
порядка рекурсивных вызовов

//функция считывания списка имен из файла
template<typename StreamT>
std::vector<std::string> getNames(StreamT &in) {
    std::vector<std::string> names;
    std::string line;

    while (std::getline(in, line) && !line.empty()) {
        names.push_back(line);
    }

    if (names.empty()) {
        throw std::runtime_error("Список имен пуст!");
    }
    return names;
}

//функция формирует матрицу смежности используя данные из файла
template<typename StreamT>
void getRelationsMatrix(StreamT &in,
std::vector<std::vector<bool>> &relationTable) {
    std::string line;
    int temp;
    int parent;
    while (std::getline(in, line) && !line.empty()) {
        std::istringstream str(line);
        if (!(str >> parent))
            throw std::runtime_error("ошибка: неверный номер
родителя!");
        else {
            if (parent >= relationTable.size() || parent < 0)
                throw std::runtime_error("ошибка: неверный номер
родителя!");
            else {
                while (str >> temp) {
```

```
                    if (temp >= relationTable.size() || temp < 0)
{
                        throw std::runtime_error("ошибка: неверные
номер ребенка!");
                    }
                    relationTable[parent][temp] = true;
                    relationTable[temp][parent] = true;
                }
            }
        }
    }
}

//функция рекурсивно проверяет, являются ли 2 элемента
родственниками
bool isFamily(int a, int b, std::vector<std::vector<bool>> &arr) {
    if (a == b)
        return false;
    if (arr[a][b]) {
        return true;
    } else {
        for (int i = 0; i < arr.size(); i++) {
            if (arr[a][i]) {
                arr[a][i] = false;
                arr[i][a] = false;
                if (isFamily(i, b, arr)) {
                    arr[a][i] = true;
                    arr[i][a] = true;
                    return true;
                }
                arr[a][i] = true;
                arr[i][a] = true;
            }
        }
    }
    return false;
}

//функция рекурсивно проверяет, являются ли 2 элемента
родственниками и выводит порядок вызовов
bool isFamilyWithDetails(int a, int b,
std::vector<std::vector<bool>> &arr) {
    std::string sp(spaceCount, ' '); // строка с табами
    std::cout << sp << "isFamily(" << a << " and " << b << ")  ";
    if (a == b)
        return false;
```

```
    if (arr[a][b]) {
        std::cout << "(Родственники)" << std::endl;
        return true;
    } else {
        spaceCount++;
        std::cout << "-" << std::endl;
        for (int i = 0; i < arr.size(); i++) {
            if (arr[a][i]) {
                arr[a][i] = false;
                arr[i][a] = false;
                if (isFamilyWithDetails(i, b, arr)) {
                    arr[a][i] = true;
                    arr[i][a] = true;
                    spaceCount--;
                    return true;
                }
                arr[a][i] = true;
                arr[i][a] = true;
            }
        }
    }
    spaceCount--;
    if (spaceCount == 0)
        std::cout << "(Не родственники)" << std::endl;
    return false;
}

//функция выводит имена и их номера в списке
void printNames(std::vector<std::string> names) {
    std::cout << "Вывод имен" << std::endl;
    for (int i = 0; i < names.size(); i++)
        std::cout << i << " : " << names[i] << std::endl;
}

std::vector<std::vector<bool>> createArray(int size) {
    std::vector<std::vector<bool>> res;
    res.resize(size);
    for (auto &v : res) {
        v.resize(size);
    }
    return res;
}

//функция выводит матрицу смежности
void printRelationsMatrix(std::vector<std::vector<bool>> &arr) {
    std::cout << "вывод матрицы смежности" << std::endl;
```

```
    for (int i = 0; i < arr.size(); i++) {
        for (int j = 0; j < arr.size(); j++) {
            std::cout << arr[i][j] << " ";
        }
        std::cout << std::endl;
    }

}

//функция выводит пары родственников
void printPairs(std::vector<std::string> names,
std::vector<std::vector<bool>> &arr) {
    std::cout << "вывод пар" << std::endl;
    for (int i = 0; i < arr.size() - 1; i++)
        for (int j = i + 1; j < arr.size(); j++)
            if (isFamily(i, j, arr))
                std::cout << names[i] << " <--> " << names[j] <<
std::endl;
}

void printPairsWithDetails(std::vector<std::string> names,
std::vector<std::vector<bool>> &arr) {
    std::cout << "вывод пар" << std::endl;
    for (int i = 0; i < arr.size() - 1; i++)
        for (int j = i + 1; j < arr.size(); j++)
            if (isFamilyWithDetails(i, j, arr))
                std::cout << names[i] << " <--> " << names[j] <<
std::endl;
}

int main() {
    try {
        std::cout << "1 - read from file  2 - read from console"
<< std::endl;


        int readFormat = 0;
        std::cin >> readFormat;
        std::cin.ignore();
        std::ifstream in;
        std::vector<std::string> names;// массив имен жителей

        if (readFormat == 1) {
            std::cout << "Введите имя файла: ";
            std::string fileName;
            std::cin >> fileName;
```

```
            in.open(fileName);
            if (in) {
                names = getNames(in);
            } else {
                throw std::runtime_error("Файл не найден!");
            }
        } else if (readFormat == 2) {

            std::cout << "введите сиписок имен через enter" <<
std::endl;
            std::cout << "Конец ввода - пустая строка" <<
std::endl;
            names = getNames(std::cin);
        } else
            throw std::runtime_error("Неправильный формат ввода");
        std::cout << "Количество жителей :" << names.size() <<
std::endl;

        std::vector<std::vector<bool>> relationsMatrix =
createArray(names.size());

        if (readFormat == 1) {


            getRelationsMatrix(in, relationsMatrix);
        } else {

            std::cout
                    << "Введитете номер родителя и номера его
детей через пробел: <№ родителя> <№ ребенка 1> <№ ребенка 2> ..."
                    << std::endl;
            std::cout << "Конец ввода - пустая строка" <<
std::endl;
            getRelationsMatrix(std::cin, relationsMatrix);
        }

        std::cout << "Введите код выбраного действия" <<
std::endl;
        int printValue = 0;
        std::cout
                << " 1 - вывести пары\n"
                   " 2 - вывести пары с деталями поиска\n"
                   " 3 - вывести список имен\n"
                   " 4 - вывести матрицу смежности\n"
                   " 0 - выход из программы"
                << std::endl;
```

```
        while (std::cin >> printValue) {
            switch (printValue) {
                case 1: {
                    printPairs(names, relationsMatrix);
                    break;
                }
                case 2: {
                    printPairsWithDetails(names, relationsMatrix);
                    break;
                }
                case 3: {
                    printNames(names);
                    break;
                }
                case 4: {
                    printRelationsMatrix(relationsMatrix);
                    break;
                }
                case 0: {
                    return 0;
                }
                default: {
                    throw std::runtime_error("Ошибка: выбрано
неправильное действие");
                }
            }
        }
    } catch (std::exception &e) {
        std::cerr << e.what() << std::endl;
    }
    return 0;
}
```