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. 8304 _____ Бутко А.М.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2019

Цель работы.

Изучить основы рекурсии, построение и составление алгоритмов.

Задание.

4 вариант.

Напечатать все перестановки заданных n различных натуральных чисел (или символов).

Описание алгоритма.

Количество перестановок для N различных элементов составляет N!. Алгоритм выводит перестановки в лексикографическом порядке (т.е. первой будет перестановка (1 2 .. N, последней – N N-1 .. 1).

Рассмотрим алгоритм. Для решения данной задачи была реализована функция permutation, которая принимает вектор и “базовый” индекс первого элемента. В функции реализован цикл, в котором происходит перестановка элемента под “базовым” индексом с элементом под индексом итератора цикла. Происходит рекурсивный вызов функции, но теперь с “базовым” индексом будет следующий (слева направо) элемент вектора. Рекурсия прекратится когда “базовый” индекс ставен равен индексу последнего элемента.

Описание основных структур данных и функций.

Программа может считывать информацию из пользовательского файла и из консоли. Результат записывается в файл.

1) Функция permutation – описана выше.
2) Функция vectorCheck.

Данная функция производит проверку введенных данных на наличие некорректных значений (не могут быть введены смешанные значения, пустой набор, набор из символов и чисел).

3) Функция check.

Данная функция отвечает за вывод сообщений об ошибках и запуск основной рекурсивной функции, если ошибки встречены не были.

5) Функция fromFile().

Функция считывает полный путь до файла, где расположены тесты, и путь до файла куда сохранить результаты тестов.

6) Функция fromConsole().

По аналогии с предыдущей функцией, только вводится не путь, а сразу тесты.

7) Функция menu().

Ведет основной диалог с пользователем.

Тестирование программы.

TEST:	RESULT:
#1 1 2 3	123 132 213 231 321 312
#2 a b c	abc acb bac bca cba cab
#3 1 a 2	ERROR: Numbers OR letters only!
#4 10 20	1020 2010

#5 ab bc	abbc bcab
#6 10a b c	ERROR: Numbers OR letters only!
#7	ERROR: Empty array!
#8 1	1

Вывод.

Задача была решена рекурсивным методом. Алгоритм, использованный при решении я считаю не таким быстрым как нерекурсивный Алгоритм Нарайаны, но гораздо более прозрачным и понятным.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
using namespace std;
void permutation(ofstream &file, vector<string> &arr, int i, int n) {
    if (i == arr.size() - 1) {
        for (int j = 0; j < n; j++) file <<' ';
        for (const string &j : arr) file << j;
        file << '\n';
    } else {
        for (unsigned long long j = i; j < arr.size(); j++) {
            swap(arr[i], arr[j]);
            permutation(file, arr, i + 1, n++);
            swap(arr[i], arr[j]);
        }
    }
}

int vectorCheck(vector<string> &arr) {
    int flag = 0, global_counter_num = 0, global_counter_ch = 0;
    for (auto &i : arr) {
        int counter_num = 0, counter_ch = 0;
        for (char j : i) {
            if (isdigit(j)) counter_num++;
            if (isalpha(j)) counter_ch++;
        }
        if (counter_num == i.length()) global_counter_num++;
        if (counter_ch == i.length()) global_counter_ch++;
    }
    if (global_counter_ch == arr.size() || global_counter_num == arr.size())
flag = 1;
    return flag;
}
void check(vector<string> &arr, ofstream &file) {
    if (arr.empty()) file << "ERROR: Empty array!" << endl;
    else if (vectorCheck(arr)) permutation(file, arr, 0, 0);
    else file << "ERROR: Numbers OR letters only!" << endl;
}
int fromFile() {
```

```
        string file_name, log_file;
        cout << " Enter test-file location: " << endl;
        cin >> file_name;
        ifstream file;
        file.open(file_name);
        if (!file.is_open()) {
            cout << "ERROR: File is not open" << endl;
            return 0;
        }
        cout << " Enter where to save results (location with <name>.txt): " <<
endl;
        cin >> log_file;
        ofstream log(log_file);
        if (!log.is_open()) {
            cout << "ERROR: File is not open" << endl;
            return 0;
        }
        string str, word;
        int i = 1;
        while (!file.eof()) {
            getline(file, str);
            cout << str << endl;
            istringstream iss(str);
            vector<string> array;
            while (iss >> word) array.push_back(word);
            array.erase(array.begin());
            log << "TEST #" << i << endl;
            log << str << endl;
            log << "RESULT:" << endl;
            check(array, log);
            log << endl;
            i++;
        }
        log.close();
        return 0;
    }
    int fromConsole() {
        string log_file, str, word;
        cout << " Enter your test: " << endl;
        cin.ignore();
        getline(cin, str, '\n');
        cout << " Enter where to save results (location with <name>.txt): " <<
endl;
```

```
    cin >> log_file;
    ofstream log(log_file);
    if (!log.is_open()) {
        cout << "ERROR: File is not open" << endl;
        return 0;
    } else cout << "SAVED";
    istringstream iss(str);
    vector<string> array;
    while (iss >> word) array.push_back(word);
    check(array, log);
    log << endl;
    return 0;
}
void menu() {
    int choice = 0;
    cout << "* * * * SEARCH FOR PERMUTATIONS * * * *" << endl;
    cout << "  What type of test do you want to do?" << endl;
    cout << "        1) from the file" << endl;
    cout << "        2) from the console" << endl;
    cout << "  enter any other number to exit." << endl;
    cout << "* * * * * * * * * * * * * * * * * * * *" << endl;
    cin >> choice;
    switch (choice) {
        case 1:
            cout << "* * * * SEARCH FOR PERMUTATIONS * * * *" << endl;
            cout << "* * * * * * * FROM THE FILE * * * * * *" << endl;
            fromFile();
            break;
        case 2:
            cout << "* * * * SEARCH FOR PERMUTATIONS * * * *" << endl;
            cout << "* * * * *  FROM THE CONSOLE * * * * * *" << endl;
            fromConsole();
            break;
        default:
            cout << "               * EXIT *                " << endl;
            break;
    }
}
int main() {
    menu();
    return 0;
}
```