**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МОЭВМ**

# ОТЧЕТ

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных »**

**Тема: Рекурсия**

| | | |
|---|---|---|
| Студент гр. 9384 | | Давыдов Д.С. |
| Преподаватель | | Ефремов М.А. |

Санкт-Петербург

2020

**Цель работы.**

Написать программу, считывающую натуральные числа (или символы) из файла, а затем записывая все их перестановки, рассчитанные с использованием рекурсии, в отдельный файл.

**Задание.**

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

**Анализ задачи.**

Задача заключается в написании алгоритма нахождения всех перестановок заданных n чисел или символов с использованием в нем рекурсии.

Для решении этой задачи был использован принцип построения рекурсивного дерева вглубь.

**Реализация:**

Реализовано следующим образом. Начинаем с первого символа `int pos = 0`. Хотим заменить его с каким либо другим (важное замечание - замену можно делать только с элементом находящимся дальше заменяемого), но видим, что перед ним еще один символ, который тоже можно заменить. Так доходим до последнего элемента, сделали это с помощью цикла for, где новая итерация не начинается ни в одном из рекурсивном вызове функции, понимаем что элементов для замены не осталось (ведь это последний элемент) `pos >= arrayIn.size()`. Записываем этот элемент как первую перестановку `arrayOut[permutationNumber] = arrayIn;` , увеличиваем номер будущей перестановки `permutationNumber++;`. Возвращаемся к предыдущему рекурсивному вызову (прошлому элементу) — `return`, и тогда уже заменяем этот элемент со следующее стоящим (из которого мы вышли), то есть начали

новую итерацию в цикле for в этом рекурсивном выводе, тем самым увеличив `i++` и заменив элемент не самим с собой, а со следующее стоящим `swap(arrayIn[i+1], arrayIn[pos])`;. Снова производим рекурсивный вызов, то есть (***Важно!!!)*** получив другую перестановку, уже в ней, начиная с того элемента который мы заменили, снова ищем следующее стоящие элементы которые мы можем заменить (то есть повторяем все предыдущие шаги), но всегда цикл во второй перестановке не может уйти дальше, так что эта перестановка сразу записывается. Важное замечание: мы должны вернуть нашу перестановку в исходное состояние (в состояние до того, как мы нашли элемент, который заменили). Реализовано это с помощью повторной замены элементов, когда мы выходим из рекурсивного вызова, после строчки вызова функции (вызова когда мы нашли элементы которые хотим заменить). Так мы доходим до самого первого элемента и заменяем его со следующее стоящим, цикл опять повторяется. В конце получаем все возможные перестановки.

Разработанный код смотреть в приложении А.

**Выполнение работы.**

Пользователь вводит название файла, откуда будут считаны данные. Далее создается поток по считыванию текста из файла `std::ifstream in`. Проверяем существует ли данный файл in.is_open, если нет — то выводим сообщение "No such file" и прекращаем работу программы.

Пользователь вводит количество чисел или символов (далее элементы)

В цикле `while`, пока в потоке in находятся символы, считываем символ кроме пробела и переноса строки `\n`, добавляем его в строку `arrIn.push_back(c)`

Если размер строки не совпадает с n, введенным пользователем, программа выводит сообщение `"The number of elements does not match"` и прекращает работу.

Считается количество перестановок `int sizeArrOut =`

`factorial(arrIn.size())` по формуле равной n! , где n – количество элементов. Массив перестановок представляет из себя двумерный массив.

Создаем массив строк (перестановок) `std::string arrOut[permutationsNumber]`.

Получаем все перестановки в массиве `arrOut`, путем вызова функции permutations.

Далее записываем все перестановки в файл путем вызова функции `writeInFile`. В ней пользователь вводит название файла, в который будут записаны все перестановки. Если такого файла не будет существовать, то он создаться автоматически.

Разработанный код смотреть в приложении А.

**Тестирование.**

Результаты тестирования представлены в табл. 1.

Таблица 1 - результаты тестирования.

| № теста | Входные данные | Выходные данные | Комментарии |
|---|---|---|---|
| 1 | 4 //пользователь ввел количество элементов<br><br>1 u<br>I<br>4 | Представлены в файле test1.txt<br>1uI4<br>1u4I<br>1Iu4<br>1I4u<br>14Iu<br>14uI<br>u1I4<br>u14I<br>uI14<br>uI41<br>u4I1<br>u41I<br>Iu14<br>Iu41<br>I1u4<br>I14u | Элементы перестановок могут состоять как из символов, так и цифр одновременно, пробелы и символы переноса строки не мешают программе считывать их |

| | | | |
|---|---|---|---|
| | | I41u<br>I4u1<br>4uI1<br>4u1I<br>4Iu1<br>4I1u<br>41Iu<br>41uI | |
| 2 | 3 //пользователь ввел количество элементов<br><br>1234 | „The number of elements does not match" | Введенное пользователем число элементов не совпадает с количеством введенных символов. |
| 3 | 3 //пользователь ввел количество элементов<br><br>//Пользователь ввел название несуществующего файла | "No such file" | Введенное пользователем название файла не принадлежит ни к одному из файлов (его не существует) |
| 4 | 3 //пользователь ввел количество элементов<br><br>123 | Представлены в файле test3.txt<br>123<br>132<br>213<br>231<br>321<br>312 | Успешная перестановка |
| 5 | 3 //пользователь ввел количество элементов<br><br>)+, | Представлены в файле test4.txt<br>)+,<br>),+<br>+),<br>+,)<br>,+)<br>,)+ | Перестановки могут осуществляться не только с буквами. |

**Вывод**

В ходе выполнения лабораторной работы была создана рабочая программа с необходимым API интерфейсом, которая считывает из файла строчку, разделяет её на элементы, а затем записывает все их перестановки в отдельный файл. На практике были закреплены навыки по написанию и использованию рекурсивных функций, реализации и представлении в программном виде численных и символьных перестановок, по работе с файлами и строками.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

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

using namespace std;

int permutationNumber = 0;

void permutations(string* arrayOut, string arrayIn, int pos = 0)
{
    if (pos >= arrayIn.size())
    {
        arrayOut[permutationNumber] = arrayIn;
        cout << arrayIn << '\n';
        permutationNumber++;
        return;
    }
    else
    {
        for (int i = pos; i < arrayIn.size(); ++i)
        {
            swap(arrayIn[i], arrayIn[pos]);
            permutations(arrayOut,arrayIn, pos + 1);
            swap(arrayIn[i], arrayIn[pos]);
        }
    }
}

int factorial(int i) //Функция считающая факториал
{
    if(i == 0) return 1;
    else return i*factorial(i-1);
```

```
}

void writeInFile(string *arrOut, int sizeOut) // Функция записывает
перестановки в файл
{
    std::string filePathOut; // Готовим файл для работы с программой
    std::cout << "Enter output file:\n";
    std::cin >> filePathOut;
    std::ofstream out;
    out.open(filePathOut);

    if (out.is_open())
    {
        for (int i = 0; i < sizeOut; i++)
        {
            out<<arrOut[i];
            out << '\n';
        }
    }
    out.close();
}

int main(){
    string filePathIn; // Готовим файл для работы с программой
    cout << "Enter input file:\n";
    cin >> filePathIn;
    std::ifstream in;
    in.open(filePathIn);
    if(!in.is_open()){ cout<<"No such file";return 1;}
    string arrIn;
    cout<<"Enter number of elements:\n";
    int n;
    cin>>n;
    char c;
    while(in.get(c)){
        if(c!=' ' && c!='\n'){
        arrIn.push_back(c);
```

```
        }
    }
     if(n!=arrIn.size()){cout<<"The number of elements does not match";
return 1;}
    int permutationsNumber = factorial(arrIn.size());
    std::string arrOut[permutationsNumber];
    cout<<"Number of permutations:\n"<<permutationsNumber<<endl;
    cout<<"Your permutations:\n";
    permutations(arrOut,arrIn);
    writeInFile(arrOut,permutationsNumber);
    return 0;
}
```

Название файла: makefile

```
main1: main.o
     g++ main.o -o main1
main.o: main.cpp
     g++ -c main.cpp
clean:
     rm *.o main1
```