**Министерство образования и науки Российской Федерации**

*Федеральное государственное бюджетное образовательное учреждение   
высшего профессионального образования*   
**«АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И.И. ПОЛЗУНОВА» (АлтГТУ)**

Факультет информационных технологий

Кафедра прикладной математики

Отчёт по дисциплине «Алгоритмы и структуры данных»

Лабораторная работа №6

«Комбинаторика»

Выполнил:

Студент гр. ПИ-32

Глушков Г.Г.

Проверил:

ст. преподаватель Н.Д. Бубнова

Барнаул 2015

* Формулировка задачи:
* Опытный взломщик по имени Brute Force решил посчитать, сколько можно придумать паролей из заданного пароля путем перестановки букв. Первоначальный пароль задается в «input.txt».
* Алгоритм решения:

Перестановку будем последовательно получать слева направо. Сначала мы имеем N пустых позиций перестановки. Если выписать в столбец все перестановки в лексикографическом порядке то увидим, что их можно разбить на N групп размерностью по (N-1)! каждая. Зная номер перестановки можно без проблем определить в какую группу она попадет. Номер группы и уже полученное начало перестановки однозначно определяют текущий элемент перестановки. Текст программы с комментариями:

// ConsoleApplication1.cpp: определяет точку входа для консольного приложения.

//Опытный взломщик по имени Brute Force решил посчитать, сколько можно придумать паролей

//из заданного пароля путем перестановки букв. Первоначальный пароль задается в «input.txt».

#include "stdafx.h"

#include <iostream>

#include <vector>

#include <fstream>

#include <string>

using namespace std;

int fact[13] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, 39916800, 479001600};

int notUsed(vector<bool> &used, int blockNum) {

int j, pos = 0;

for (j = 1; j < used.size(); j++) {

if (!used[j]) pos++;

if (blockNum == pos)

break;

}

return j;

}

vector<int> permutation\_by\_num(int n, int num) {

vector<int> res(n);

vector<bool> used(n+1,false);

for (int i = 0; i < n; i++) {

int blockNum = (num - 1) / fact[n - i - 1] + 1;

int j = notUsed(used, blockNum);

res[i] = j;

used[j] = true;

num = (num - 1) % fact[n - i - 1] + 1;

}

return res;

}

void output(vector<int> &perm , string s) {

for (int i=0;i<perm.size();i++)

// cout<<perm[i]<<' ';

cout<<s[perm[i]-1]<<' ';

}

template<int N>

struct Factorial

{

enum { value = N \* Factorial<N-1>::value };

};

template<>

struct Factorial<1>

{

enum { value = 1 };

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

ifstream TextFile1("t.txt"); //открываю файл

string s1;

while (!TextFile1.eof()) //пока не конец файла..

{

if (TextFile1.eof()) break; //если конец закончить считывание

TextFile1 >> s1; //или пока не закончится файл,

} //записывать его содержимое в s1

TextFile1.close();

int n,num,num1;

const int fact4 = Factorial<5>::value;

cout << fact4<<endl;

n = s1.size();

const int fact = n;

cin>>num>>num1;

for(int i =num; i < num1; i++){

vector<int> perm = permutation\_by\_num(n,i);

output(perm, s1);

cout << endl;

}

// vector<int> perm = permutation\_by\_num(n,num);

// output(perm);

system("pause");

return 0;

}

* Тесты:

Первая строка – количество всевозможных перестановок

Вторая строка – образец

Третья и четвертая строка – диапазон генерации.

Последующие строки – ответ к заданию.



