**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 |  | Кузнецов Р.В. |
| Преподаватель |  | Филатов А.Ю. |

Санкт-Петербург

2018

## Цель работы.

Изучить рекурсивный подход к решению задач, научиться применять рекурсию.

## Задание.

Вариант 4.

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

## Описание работы алгоритма.

На вход программа запрашивает у пользователя последовательность слов, разделенных пробелом Вводимые данные должны быть различны. На первой строчке в выводе программы записано количество слов в такой последовательности, на последующих через пробел выводятся все перестановки.

Для выполнения работы была реализована рекурсивная лямбда-функция permutationGen. Она захватывает по ссылке вектор строк seq, в котором хранятся исходные данные. Также по значению она принимает итератор на начало вектора строк seq и ссылку на себя, чтобы можно было реализовать рекурсию. Алгоритм ее работы довольно прост и заключается в перестановке каждого элемента с каждым: Для каждого элемента из вектора вызываются несколько таких же функций, для которых текущий элемент поменян местами со всеми последующими, затем эти два элемента меняются местами обратно. Если глубина такой рекурсии достигла такой точки, что перебраны уже все элементы вектора со всеми *(begin == seq.end())*, полученная последовательность выводится в *stdout*. Несмотря на то, что перестановки и замены местами производятся над одним и тем же вектором, по окончании работы функции он эквивалентен начальному, что экономит много памяти.

## Тестирование.

Для тестирования программы была написана еще одна программа на языке C++ и баш скрипт. Скрипт подает на вход программе тесты из папки *Tests*, представляющие собой .*txt* файлы с вводимыми данными и записывает результат в *result.txt*. Затем этот файл используется как входные данные для тестирующей программы. Она генерирует свой (гарантированно правильный) результат с помощью функции стандартной библиотеки *std::next\_permutation*, затем сравнивает эти два результата. Если они равны, то тест пройден, иначе выводится результат обеих алгоритмов с сообщением об ошибке. Для запуска тестирования нужно запустить скрипт *test.sh*. Этот скрипт сначала компилирует файлы из исходников (с помощью скрипта *compile.sh*), затем запускает эту проверочную систему. По окончании проверки, скомпилированные программы удаляются вместе с *result.txt*. Для операционной системы Linux были написаны аналогичные по функционалу *bash*-скрипты. Результат тестирования выводится в *stdout*. Таблица по результатам тестирования продолжается в приложении: Таблица Б.2.

Таблица Б.1 – Результат тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 1 | 1  1 | 1 цифра |
|  | a | 1  a | 1 буква |
|  | a b | 2  a b  b a | 2 буквы |
| 4. | single | 1  single | 1 слово |

## Выводы.

В процессе выполнения работы был изучен рекурсивный подход к решению задач. Также были получены навыки применения конструкций современного С++, таких как лямбда функции.

Была реализована программа, рекурсивным методом выводящая все перестановки вводимых данных. Также была реализована тестирующая программа, проверяющая правильность выводимого результата, и пара *bash*-скриптов, упрощающих использование программы.

# Приложение А Исходный код программы

Название файла: main.cpp

#include <iostream>

#include <vector>

#include <iterator>

#include <string>

#include <algorithm>

template <typename T>

std::ostream& operator<<(std::ostream & out, const std::vector<T> & seq) {

out << \*seq.begin();

for (auto it = seq.begin() + 1; it != seq.end(); it++) {

out << " " << \*it;

}

return out;

}

template <typename T>

std::istream& operator>>(std::istream& in, std::vector<T>& seq) {

while (std::cin.peek() != '\n')

{

std::string tmp;

in >> tmp;

seq.emplace\_back(std::move(tmp));

}

return in;

}

int main() {

std::vector<std::string> seq;

std::cin >> seq;

std::cout<<seq.size()<<"\n";

auto permutationGen = [&seq](std::vector<std::string>::iterator begin, auto && permutationGen) {

//printing

if (begin == seq.end()) {

std::cout << seq << '\n';

return;

}

//recource block

for (auto i = begin; i != seq.end(); i++) {

std::swap(\*begin, \*i);

permutationGen(begin + 1, permutationGen);

std::swap(\*begin, \*i);

}

};

permutationGen(seq.begin(), permutationGen);

return 0;

}

Название файла: unitTests.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <set>

#include <vector>

#include <string>

#include <algorithm>

#include <sstream>

#include <iterator>

unsigned GLwordsInSeq;

std::istream& operator>>(std::istream& in, std::vector<std::string>& seq) {

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

std::string tmp;

in >> tmp;

seq.emplace\_back(std::move(tmp));

}

char c[1];

in.getline(c, 1);

return in;

}

std::istream& operator>>(std::istream& in, std::string& seq) {

getline(in, seq);

return in;

}

unsigned fact(int n) {

unsigned res = 1;

for (int i = 1; i <= n; i++)

res \*= i;

return res;

}

template <typename T>

std::ostream& operator<<(std::ostream & out, const std::vector<T> & seq) {

out << \*seq.begin();

for (auto it = seq.begin() + 1; it != seq.end(); it++) {

out << " " << \*it;

}

return out;

}

template<typename T>

std::string str(T begin, T end)

{

std::stringstream ss;

for (bool first = true; begin != end; begin++)

{

if (!first)

ss << " ";

ss << \*begin;

first = false;

}

return ss.str();

}

int main() {

std::cin >> GLwordsInSeq;

unsigned GLnumOfSeqs = fact(GLwordsInSeq);

std::vector<std::string> seq;

std::cin >> seq;

std::cout << "Running test [" << seq << "]...\n";

std::set<std::string> myPermutations, truePermutations;

myPermutations.insert(str(seq.begin(), seq.end()));

unsigned permCount = GLnumOfSeqs;

for (int i = 0; i < permCount - 1; i++) {

std::string tmp;

std::cin >> tmp;

if (myPermutations.count(tmp)) {

std::cout << "Test failed: your algorithm generated identical permutations\n";

return 0;

}

myPermutations.emplace(std::move(tmp));

}

std::sort(seq.begin(), seq.end());

do {

truePermutations.insert(str(seq.begin(), seq.end()));

} while (std::next\_permutation(seq.begin(), seq.end()));

std::vector<std::string> diff;

std::set\_difference(myPermutations.begin(), myPermutations.end(),

truePermutations.begin(), truePermutations.end(), std::inserter(diff, diff.begin()));

if (!diff.size() && myPermutations.size() == truePermutations.size()) {

std::cout << "Test: completed successfully\n";

return 0;

}

std::cout << "your alg ans\n";

for (auto it = myPermutations.begin(); it != myPermutations.end(); it++)

std::cout << \*it << "\n";

std::cout << "------------\n";

std::cout << "true ans\n";

for (auto it = truePermutations.begin(); it != truePermutations.end(); it++)

std::cout << \*it << "\n";

std::cout << "------------\n";

std::cout << "Test [" << seq << "] completed successfully\n\n";

}

# Приложение Б Тестирование

Таблица Б.2 – Результат тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | a b c d | 4  a b c d  a b d c  a c b d  a c d b  a d c b  a d b c  b a c d  b a d c  b c a d  b c d a  b d c a  b d a c  c b a d  c b d a  c a b d  c a d b  c d a b  c d b a  d b c a  d b a c  d c b a  d c a b  d a c b  d a b c | 4 символа |
|  | 1 20 30 | 3  1 20 30  1 30 20  20 1 30  20 30 1  30 20 1  30 1 20 | 3 числа |
|  | as i said | 3  as i said  as said i  i as said  i said as  said i as  said as i | 3 слова |
| 8. | there is no anime | 4  there is no anime  there is anime no  there no is anime  there no anime is  there anime no is  there anime is no  is there no anime  is there anime no  is no there anime  is no anime there  is anime no there  is anime there no  no is there anime  no is anime there  no there is anime  no there anime is  no anime there is  no anime is there  anime is no there  anime is there no  anime no is there  anime no there is  anime there no is anime there is no | 4 слова |
| 9. | like at all | 3  like at all  like all at  at like all  at all like  all at like  all like at | 3 слова |
| 10. | mixed 1 input 2 | 4  mixed 1 input 2  mixed 1 2 input  mixed input 1 2  mixed input 2 1  mixed 2 input 1  mixed 2 1 input  1 mixed input 2  1 mixed 2 input  1 input mixed 2  1 input 2 mixed  1 2 input mixed  1 2 mixed input  input 1 mixed 2  input 1 2 mixed  input mixed 1 2  input mixed 2 1  input 2 mixed 1  input 2 1 mixed  2 1 input mixed  2 1 mixed input  2 input 1 mixed  2 input mixed 1  2 mixed input 1  2 mixed 1 input | 2 слова и 2 числа в одном тесте |