|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

Дисциплина Методы комбинаторных вычислений

Название работы Лабораторная работа №2

Студент Долженко Анастасия Тимофеевна

Группа РК6-52Б

Вариант 7

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Долженко А.Т.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Волосатова Т.М.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2024 г.*

**Оглавление**

[Задание 3](#_Toc178644053)

[Алгоритм работы программы 3](#_Toc178644054)

[Приложение 1. Код программы 4](#_Toc178644055)

[Приложение 2. Результаты работы программы 6](#_Toc178644056)

[Список литературы 6](#_Toc178644057)

# Задание

Вариант 7

Перечислить все перестановки из 5-ти десятичных цифр от 1 до 5, располагая их в порядке неубывания суммы инверсий своих элементов.

# Алгоритм работы программы

Программа генерирует перестановки, подсчитывая для каждой количество инверсий. Затем сортирует эти перестановки по количеству инверсий и печатает их на экране.

Генерация следующей перестановки происходит по следующему алгоритму. Покажем получение следующей перестановки после [1, 2, 3, 5, 4]. Условимся, что Pi – элемент перестановки под i-ым индексом.

1. *Сначала мы ищем наибольший индекс k, такой что Pk < Pk+1*

* P3 = 5 > P4 = 4
* P2 = 3 < P3 = 5 => k = 2

1. *Теперь мы ищем наибольший индекс l > k, такой что Pk < Pl*

P2 = 3 < P4 = 4 => l = 4

1. *Меняем местами значения Pk и Pl*: [1, 2, 4, 5, 3]
2. *Разворачиваем порядок элементов после индекса k* (k = 2)

Берем подмассив [5, 3] и разворачиваем его, так мы получаем следующую перестановку [1, 2, 4, 3, 5]

# Приложение 1. Код программы

#include <stdio.h>

#include <stdlib.h>

#define N 5

#define M 120

// подсчет кол-ва инверсий

int countInversions(int \*perm){

int count = 0;

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

for (int j = i + 1; j < N; j++) {

if (perm[i] > perm[j]) {

count++;

}

}

}

return count;

}

int getNextPermutation(int \*perm, int \*\*listPermsWithInversions, int \*index){

// ищем max k: perm[k] < perm[k + 1]

int k = -1;

for (int i = N - 2; i >= 0; i--) {

if (perm[i] < perm[i + 1]) {

k = i;

break;

}

}

if (k == -1) return 0; // перебрали все перестановки, конец

// ищем max l: perm[k] < perm[l]

int l = -1;

for (int i = N - 1; i > k; i--) {

if (perm[k] < perm[i]) {

l = i;

break;

}

}

// меняем местами perm[k] и perm[l]

int temp = perm[k];

perm[k] = perm[l];

perm[l] = temp;

// разворачиваем последовательность от k + 1 до конца

for (int i = k + 1, j = N - 1; i < j; i++, j--) {

temp = perm[i];

perm[i] = perm[j];

perm[j] = temp;

}

// записываем перестановку и кол-во ее инверсий для будущей сортировки

for(int i = 0; i < N; i++) listPermsWithInversions[\*index][i] = perm[i];

listPermsWithInversions[\*index][N] = countInversions(perm);

(\*index) ++;

return 1;

}

// печать текущей перестановки

void printPermutation(int \*perm) {

for (int i = 0; i < N; i++) printf("%d ", perm[i]);

printf("\n");

// printf("%d\n", countInversions(perm));

}

int comparator(const void \*x1, const void \*x2){

const int \*perm1 = \*(const int\*\*)x1; // указатель на строку с перестановкой 1

const int \*perm2 = \*(const int\*\*)x2; // Указатель на строку с перестановкой 2

return perm1[N] - perm2[N];

}

int main()

{

int perm[N];

for (int i = 0; i < N; i++) perm[i] = i + 1;

int \*\*listPermsWithInversions = (int\*\*)malloc(M \* sizeof(int\*));

for (int i = 0; i < M; i++) listPermsWithInversions[i] = (int\*)malloc((N+1) \* sizeof(int));

int index = 0;

for(int i = 0; i < N; i++) listPermsWithInversions[index][i] = perm[i];

listPermsWithInversions[index][N] = countInversions(perm);

index++;

// printPermutation(perm);

while(getNextPermutation(perm, listPermsWithInversions, &index));

// printf("\n\n\n");

// for(int i = 0; i<M; i++){

// for(int j = 0; j < N+1; j++) printf("%d ", listPermsWithInversions[i][j]);

// printf("\n");

// }

qsort(listPermsWithInversions, M, sizeof(int\*), comparator);

// printf("\n\n\n");

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

for(int j = 0; j < N; j++)printf("%d ", listPermsWithInversions[i][j]);

printf("\t%d", listPermsWithInversions[i][N]);

printf("\n");

}

return 0;

}

# Приложение 2. Результаты работы программы

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Автоматически созданное описание

# Список литературы

1. Методы комбинаторных вычислений – Т.М. Волосатова, С.В. Родионов, 2011
2. http://bigor.bmstu.ru/?cnt/?doc=Comby/base.cou