Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 10

По дисциплине «Основы алгоритмизации и программирования»

На тему « **Рекурсивные алгоритмы**»

Выполнил:

Федорович Вадим

Студент 1 курса 8 группы

Преподаватель: асс. Андронова М.В.

Минск, 2024

Основа:

#include <iostream>

using namespace std;

double Fackt(double x, double n) {

if (n == 0)

return 1;

if (n == 1)

return x;

return x\*x /n /(n - 1)\* Fackt(x, n-2);

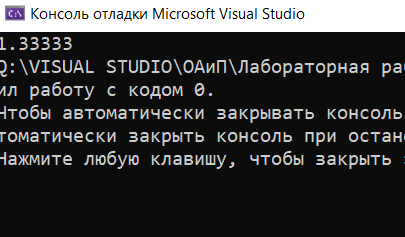
}

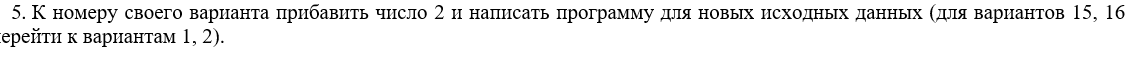
int main() {

setlocale(0, "");

cout << Fackt(2, 3);

}





#include <iostream>

using namespace std;

int FN(int n, int m) {

if (n == 0)

return 1;

if (n < m)

return -1;

return 2\*FN(n-1,m);

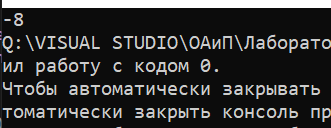
}

int main() {

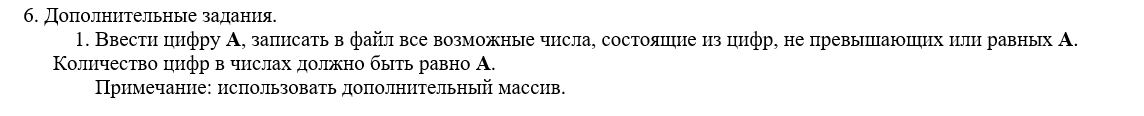
setlocale(0, "");

cout << FN(5, 3);

}



Допы:



#include <iostream>

#include <fstream>

using namespace std;

// Функция для записи числа в файл

void writeToFile(char\* currentNumber, int length, ofstream& file) {

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

cout << currentNumber[i];

file << currentNumber[i];

}

cout << endl;

file << endl;

}

// Функция для генерации чисел рекурсивно

void generate(int A, char\* currentNumber, int currentLength, ofstream& file) {

if (currentLength == A) {

writeToFile(currentNumber, A, file); // Записываем число в файл

return;

}

for (int digit = 1; digit <= A; ++digit) {

bool validDigit = true;

if (currentLength > 0) {

int lastDigit = currentNumber[currentLength - 1] - '0';

if (lastDigit > digit)

validDigit = false;

}

if (validDigit) {

currentNumber[currentLength] = '0' + digit;

generate(A, currentNumber, currentLength + 1, file);

}

}

}

// Функция для генерации чисел

void genNum(int A) {

ofstream outputFile("output.txt");

if (!outputFile) {

cerr << "Unable to create output file!" << endl;

return;

}

char currentNumber[10]; // Предполагаем, что максимальная длина числа - 10 цифр

generate(A, currentNumber, 0, outputFile);

outputFile.close();

}

int main() {

setlocale(0, "");

int A;

cout << "Введите цифру A: ";

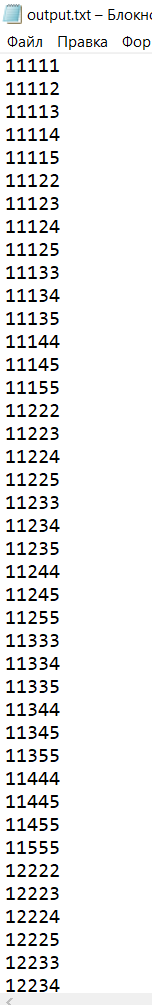
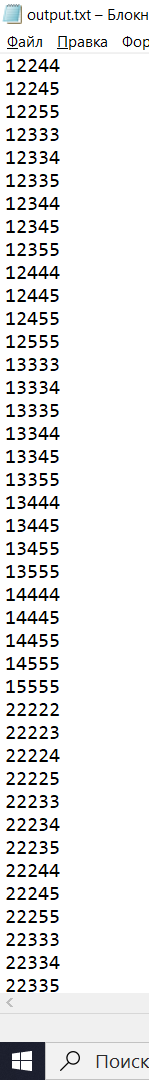
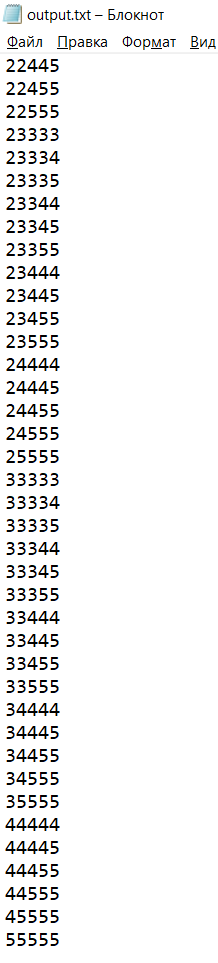
cin >> A;

genNum(A);

cout << "Генерация чисел завершена. Результаты записаны в файл output.txt." << endl;

return 0;

}



#include <iostream>

using namespace std;

// Функция для обмена значений двух переменных

void swap(int& a, int& b) {

int temp = a;

a = b;

b = temp;

}

// Рекурсивная функция для генерации всех перестановок

void generatePermutations(int numbers[], int start, int end) {

if (start == end) {

// Базовый случай: если start и end совпадают, значит массив отсортирован,

// и мы можем его распечатать как перестановку

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

cout << numbers[i] << " ";

}

cout << endl;

}

else {

// Рекурсивный случай: для каждой позиции от start до end

for (int i = start; i <= end; ++i) {

swap(numbers[start], numbers[i]);

generatePermutations(numbers, start + 1, end);

swap(numbers[start], numbers[i]);

}

}

}

int main() {

setlocale(0, "");

int n = 5;

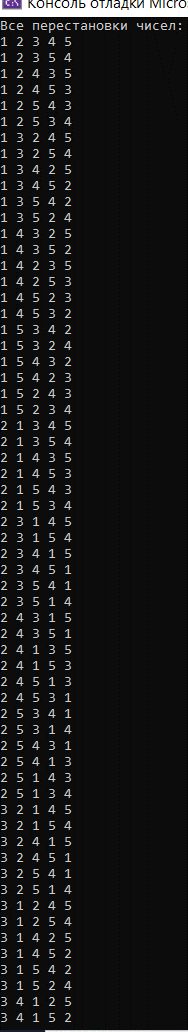
int numbers[] = { 1, 2, 3, 4, 5 };

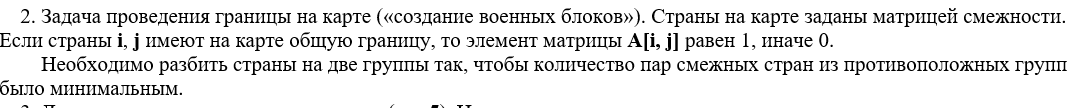
cout << "Все перестановки чисел:" << endl;

generatePermutations(numbers, 0, n - 1);

return 0;

}





#include <iostream>

#include <vector>

using namespace std;

// Функция для поиска минимального количества смежных пар стран из противоположных групп

void findMinBorderPairs(vector<vector<int>>& adjacencyMatrix) {

int n = adjacencyMatrix.size();

vector<int> group(n, -1); // Массив для хранения принадлежности стран к группе (-1 - неопределено, 0 - группа 1, 1 - группа 2)

// Начинаем с первой страны

group[0] = 0;

// Обходим все страны

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

// Если страна еще не принадлежит какой-либо группе

if (group[i] == -1) {

// Присваиваем ей противоположную группу относительно соседей

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

if (adjacencyMatrix[i][j]) {

group[i] = 1 - group[j];

break;

}

}

}

}

// Считаем количество пар смежных стран из противоположных групп

int minBorderPairs = 0;

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

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

if (adjacencyMatrix[i][j] && group[i] != group[j]) {

++minBorderPairs;

}

}

}

cout << "Минимальное количество пар смежных стран из противоположных групп: " << minBorderPairs << endl;

}

int main() {

setlocale(0, "");

// Пример матрицы смежности

vector<vector<int>> adjacencyMatrix = {

{0, 1, 1, 0},

{1, 0, 1, 0},

{1, 1, 0, 1},

{0, 0, 1, 0}

};

findMinBorderPairs(adjacencyMatrix);

return 0;

}

