Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 18

По дисциплине «Основы алгоритмизации и программирования»

На тему «Работа с файлами на языке С»

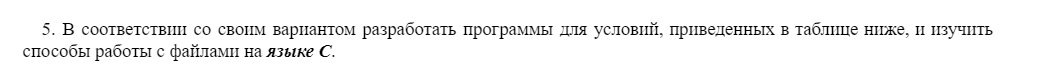
 Выполнила:

Студентка1 курса 6 группы

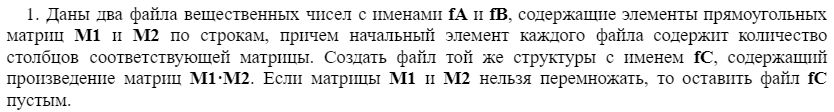
Альшевская Алина Михайловна

Преподаватель: асс. Андронова М.В.

2023, Минск



Вариант 1



#include <iostream>

#include <fstream>

#include <vector>

#include <Windows.h>

using namespace std;

void multiplyMatrices(const char\* fileA, const char\* fileB, const char\* fileC) {

// Открываем файлы для чтения

FILE\* fdA, \* fdB;

if (fopen\_s(&fdA, fileA, "r") != 0 || fopen\_s(&fdB, fileB, "r") != 0) {

cout << "Ошибка открытия файлов." << endl;

return;

}

// Считываем количество столбцов в каждой матрице

int colsA, colsB;

if (fscanf\_s(fdA, "%d", &colsA) == EOF || fscanf\_s(fdB, "%d", &colsB) == EOF) {

cout << "Ошибка чтения количества столбцов." << endl;

fclose(fdA);

fclose(fdB);

return;

}

// Создаем вектор для матрицы A

vector<vector<double>> matrixA;

// Считываем матрицу A из файла

double elementA;

while (fscanf\_s(fdA, "%lf", &elementA) != EOF) {

vector<double> row;

row.push\_back(elementA); // ДОбавляем первый элемент в вектор строки матрицы. поскольку мы не знаем сколько столбцов, то используем такой цикл, который, дойдя до конца файла вернет EOF

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

if (fscanf\_s(fdA, "%lf", &elementA) == EOF) {

cout << "Ошибка чтения матрицы A." << endl;

fclose(fdA);

fclose(fdB);

return;

}

row.push\_back(elementA);

}

matrixA.push\_back(row);

}

// Закрываем файл с матрицей A

fclose(fdA);

// Создаем вектор для матрицы B

vector<vector<double>> matrixB;

// Считываем матрицу B из файла

double elementB;

while (fscanf\_s(fdB, "%lf", &elementB) != EOF) {

vector<double> row;

row.push\_back(elementB);

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

if (fscanf\_s(fdB, "%lf", &elementB) == EOF) {

cout << "Ошибка чтения матрицы B." << endl;

fclose(fdB);

return;

}

row.push\_back(elementB);

}

matrixB.push\_back(row);

}

// Закрываем файл с матрицей B

fclose(fdB);

// Проверяем, можно ли перемножить матрицы

if (colsA != matrixB.size()) {

cout << "Невозможно перемножить матрицы. Количество столбцов первой матрицы не равно количеству строк второй матрицы." << endl;

return;

}

// Открываем файл для записи

FILE\* fdC;

if (fopen\_s(&fdC, fileC, "w") != 0) {

cout << "Ошибка открытия файла для записи." << endl;

return;

}

// Выполняем умножение матриц. .size() используется для получения количества элементов в векторе

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

for (int j = 0; j < matrixB[0].size(); j++) {

double sum = 0.0;

for (int k = 0; k < matrixB.size(); k++) {

sum += matrixA[i][k] \* matrixB[k][j];

}

fprintf\_s(fdC, "%.2f ", sum);

}

fprintf\_s(fdC, "\n");

}

// Закрываем файл с результатами

fclose(fdC);

cout << "Успешно создан файл " << fileC << " с результатом перемножения матриц." << endl;

}

int main() {

SetConsoleOutputCP(1251);

const char\* fileA = "fA.txt";

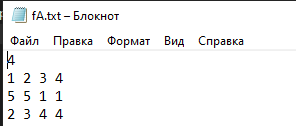
const char\* fileB = "fB.txt";

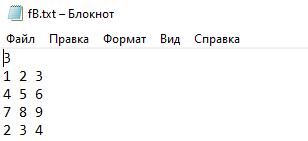
const char\* fileC = "fC.txt";

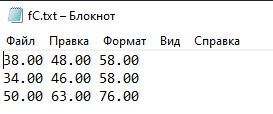
multiplyMatrices(fileA, fileB, fileC);

return 0;

}









#include <iostream>

#include <fstream>

#include <Windows.h>

#include <vector>

using namespace std;

void cute(const char\* fileF, const char\* fileG, const int K)

{

// Открываем файл для чтения

FILE\* filef;

if (fopen\_s(&filef, fileF, "r") != 0) {

printf("Не удалось открыть файл для чтения");

return;

}

// Создаем вектор для хранения чисел

vector<int> numbers;

int num;

// Считываем числа из файла и сохраняем их в вектор

while (fscanf\_s(filef, "%d", &num) != EOF) {

numbers.push\_back(num);

}

// Закрываем файл для чтения

fclose(filef);

// Открываем файл для записи

FILE\* fileg;

if (fopen\_s(&fileg, fileG, "w") != 0) {

printf("Не удалось открыть файл для записи");

return;

}

// Записываем в файл числа из вектора, которые кратны числу K

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

if (numbers[i] % K == 0) {

fprintf\_s(fileg, "%d ", numbers[i]);

}

}

// Закрываем файл для записи

fclose(fileg);

}

int main() {

// Устанавливаем кодировку для корректного вывода на консоль (CP1251)

SetConsoleOutputCP(1251);

// Имена файлов и число K

const char\* fileF = "fileF.txt";

const char\* fileG = "fileG.txt";

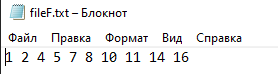
int k = 43;

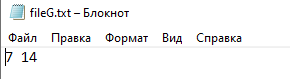
// Вызываем функцию cute с заданными параметрами

cute(fileF, fileG, k);

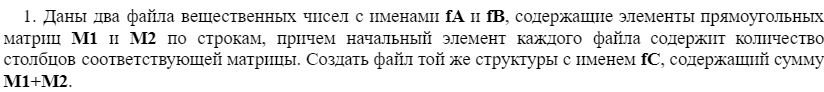
return 0;

}





Вариант 2



#include <iostream>

#include <fstream>

#include <vector>

#include <Windows.h>

using namespace std;

void addMatrices(const char\* fileA, const char\* fileB, const char\* fileC) {

// Открываем файлы для чтения

FILE\* fdA, \* fdB;

if (fopen\_s(&fdA, fileA, "r") != 0 || fopen\_s(&fdB, fileB, "r") != 0) {

cout << "Ошибка открытия файлов." << endl;

return;

}

// Считываем количество столбцов в каждой матрице

int colsA, colsB;

if (fscanf\_s(fdA, "%d", &colsA) == EOF || fscanf\_s(fdB, "%d", &colsB) == EOF) {

cout << "Ошибка чтения количества столбцов." << endl;

fclose(fdA);

fclose(fdB);

return;

}

// Проверяем, что матрицы имеют одинаковые размеры

if (colsA != colsB) {

cout << "Невозможно сложить матрицы. Количество столбцов не совпадает." << endl;

fclose(fdA);

fclose(fdB);

return;

}

// Создаем векторы для матриц A и B

vector<vector<double>> matrixA, matrixB;

// Считываем матрицы A и B из файлов

double elementA, elementB;

while (fscanf\_s(fdA, "%lf", &elementA) != EOF) {

vector<double> rowA, rowB;

rowA.push\_back(elementA); // Добавляем первый элемент в вектор строки матрицы

// Считываем соответствующий элемент из матрицы B

if (fscanf\_s(fdB, "%lf", &elementB) == EOF) {

cout << "Ошибка чтения матрицы B." << endl;

fclose(fdA);

fclose(fdB);

return;

}

rowB.push\_back(elementB);

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

if (fscanf\_s(fdA, "%lf", &elementA) == EOF || fscanf\_s(fdB, "%lf", &elementB) == EOF) {

cout << "Ошибка чтения матрицы A или B." << endl;

fclose(fdA);

fclose(fdB);

return;

}

rowA.push\_back(elementA);

rowB.push\_back(elementB);

}

matrixA.push\_back(rowA);

matrixB.push\_back(rowB);

}

// Закрываем файлы с матрицами A и B

fclose(fdA);

fclose(fdB);

// Открываем файл для записи

FILE\* fdC;

if (fopen\_s(&fdC, fileC, "w") != 0) {

cout << "Ошибка открытия файла для записи." << endl;

return;

}

// Выполняем сложение матриц. .size() используется для получения количества элементов в векторе

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

for (int j = 0; j < matrixA[0].size(); j++) {

fprintf\_s(fdC, "%.2f ", matrixA[i][j] + matrixB[i][j]);

}

fprintf\_s(fdC, "\n");

}

// Закрываем файл с результатами

fclose(fdC);

cout << "Успешно создан файл " << fileC << " с результатом сложения матриц." << endl;

}

int main() {

SetConsoleOutputCP(1251);

const char\* fileA = "fA.txt";

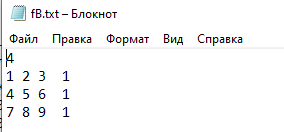
const char\* fileB = "fB.txt";

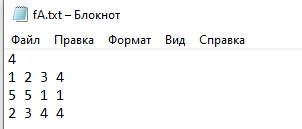
const char\* fileC = "fC.txt";

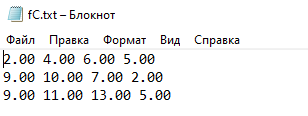
addMatrices(fileA, fileB, fileC);

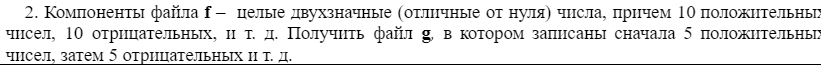
return 0;

}









#include <iostream>

#include <fstream>

#include <Windows.h>

#include <vector>

using namespace std;

void cute(const char\* fileF, const char\* fileG)

{

// Открываем файл для чтения

FILE\* filef;

if (fopen\_s(&filef, fileF, "r") != 0) {

printf("Не удалось открыть файл для чтения");

return;

}

// Создаем вектор для хранения чисел

vector<int> numbers;

int num;

int counter = 0;

// Считываем числа из файла и сохраняем первые 5 положительных чисел

// и первые 5 отрицательных чисел в вектор, используя счетчик

while (fscanf\_s(filef, "%d", &num) != EOF) {

if (counter < 5) {

numbers.push\_back(num);

}

else if (counter == 10) {

counter = 0;

}

counter++;

}

// Закрываем файл для чтения

fclose(filef);

// Открываем файл для записи

FILE\* fileg;

if (fopen\_s(&fileg, fileG, "w") != 0) {

printf("Не удалось открыть файл для записи");

return;

}

// Записываем в файл числа из вектора, которые кратны числу K

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

fprintf\_s(fileg, "%d ", numbers[i]);

}

// Закрываем файл для записи

fclose(fileg);

}

int main() {

SetConsoleOutputCP(1251);

// Имена файлов

const char\* fileF = "fileF.txt";

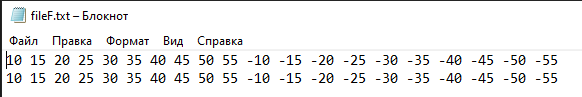
const char\* fileG = "fileG.txt";

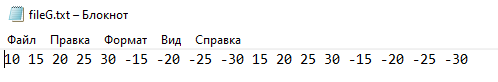
// Вызываем функцию cute с заданными параметрами

cute(fileF, fileG);

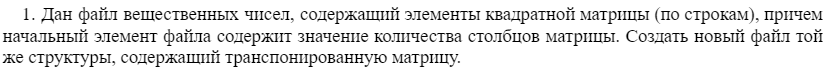
return 0;

}





Вариант 3



#include <iostream>

#include <cstdio>

#include <vector>

#include <Windows.h>

int main() {

SetConsoleOutputCP(1251);

// Открытие файла для чтения

FILE\* inputFile;

if (fopen\_s(&inputFile, "input.txt", "r") != 0) {

printf("Не удалось открыть файл для чтения.\n");

return 1;

}

// Чтение размера матрицы (количество столбцов)

int numColumns;

if (fscanf\_s(inputFile, "%d", &numColumns) != 1) {

printf("Ошибка чтения размера матрицы.\n");

fclose(inputFile);

return 1;

}

// Создание вектора для хранения матрицы

std::vector<std::vector<double>> matrix;

double element;

// Чтение матрицы из файла

while (fscanf\_s(inputFile, "%lf", &element) == 1) {

std::vector<double> row;

row.push\_back(element);

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

fscanf\_s(inputFile, "%lf", &element);

row.push\_back(element);

}

matrix.push\_back(row);

}

// Закрытие файла

fclose(inputFile);

// Открытие нового файла для записи

FILE\* outputFile;

if (fopen\_s(&outputFile, "output.txt", "w") != 0) {

printf("Не удалось открыть файл для записи.\n");

return 1;

}

// Запись транспонированной матрицы в новый файл

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

for (const auto& row : matrix) {//испльзуем иттератор. const указывает, что данные менять не будем, auto позволит компилятору самостоятельно определить тип данных на основании типа данных вектора. итерация будет происходить по matrix, итератор row

fprintf(outputFile, "%.2lf ", row[i]);

}

fprintf(outputFile, "\n");

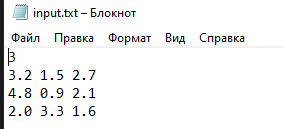
}

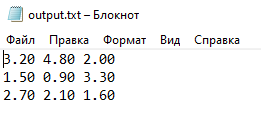
// Закрытие файла

fclose(outputFile);

return 0;

}





Вариант 4



#include <iostream>

#include <cstdio>

#include <Windows.h>

int main() {

SetConsoleOutputCP(1251);

// Открытие файла fileA для чтения

FILE\* fileA;

if (fopen\_s(&fileA, "fileA.txt", "r") != 0 || fileA == nullptr) {

printf("Не удалось открыть файл fileA.txt для чтения.\n");

return 1;

}

// Открытие нового файла fileB для записи положительных чисел

FILE\* fileB;

if (fopen\_s(&fileB, "fileB.txt", "w") != 0 || fileB == nullptr) {

printf("Невозможно открыть файл fileB.txt для записи.\n");

fclose(fileA);

return 1;

}

int number;

// Чтение и запись положительных чисел из файла fileA в файл fileB

while (fscanf\_s(fileA, "%d", &number) == 1) {

if (number > 0) {

fprintf(fileB, "%d ", number);

}

}

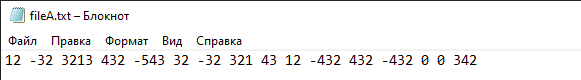
// Закрытие файлов

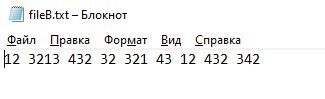
fclose(fileA);

fclose(fileB);

return 0;

}







#include <iostream>

#include <cstdio>

#include <Windows.h>

int main() {

SetConsoleOutputCP(1251); // Установка кодировки символов для вывода консоли

// Ввод числа с клавиатуры

int n;

printf("Введите пороговое значение: ");

std::cin >> n;

// Открытие файла f для чтения

FILE\* fileF;

if (fopen\_s(&fileF, "fileF.txt", "r") != 0 || fileF == nullptr) {

printf("Не удалось открыть файл fileF.txt для чтения.\n");

return 1; // Возврат ошибки

}

// Открытие нового файла g для записи чисел, больших введенного значения

FILE\* fileG;

if (fopen\_s(&fileG, "fileG.txt", "w") != 0 || fileG == nullptr) {

printf("Невозможно открыть файл fileG.txt для записи.\n");

fclose(fileF); // Закрытие открытого файла f

return 1; // Возврат ошибки

}

int number; // Переменная для хранения прочитанного числа

// Чтение и запись чисел из файла f в файл g, если они больше введенного значения

while (fscanf\_s(fileF, "%d", &number) == 1) {

if (number > n) {

fprintf(fileG, "%d ", number); // Запись числа в файл g

}

}

// Закрытие файлов

fclose(fileF);

fclose(fileG);

printf("Операция успешно выполнена.\n");

return 0;

}

