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:

Студентка 1 курса 7 группы

Бабич Виолетта Станиславовна

Преподаватель: асс. Андронова М.В.

2023, Минск

|  |  |
| --- | --- |
| **2** | 1. Даны два файла вещественных чисел с именами **fA** и **fB**, содержащие элементы прямоугольных матриц **M1** и **M2** по строкам, причем начальный элемент каждого файла содержит количество столбцов соответствующей матрицы. Создать файл той же структуры с именем **fC**, содержащий сумму **M1+M2**.  2. Компоненты файла**f** –  целые двухзначные (отличные от нуля) числа, причем 10 положительных чисел, 10 отрицательных, и т. д. Получить файл **g***,* в котором записаны сначала 5 положительных чисел, затем 5 отрицательных и т. д. |

№1

#include <stdio.h>

using namespace std;

int main() {

//объявление указателей на файлы

FILE\* fileA, \* fileB, \* fileC;

//объявление переменных для хранения информации

int columnsA, columnsB;

// Открытие файлов для чтения

fopen\_s(&fileA, "fA.txt", "r");

fopen\_s(&fileB, "fB.txt", "r");

//Создание файла

fopen\_s(&fileC, "fC.txt", "w");

// Проверка на успешное открытие файлов

if (fileA == NULL || fileB == NULL || fileC == NULL) {

perror("Ошибка открытия файлов");

return 1;

}

// Чтение количества столбцов из файлов

fscanf\_s(fileA, "%d", &columnsA);

fscanf\_s(fileB, "%d", &columnsB);

// Проверка на соответствие размерностей матриц

if (columnsA != columnsB) {

printf("Ошибка: количество столбцов матриц не совпадает\n");

return 1;

}

// Запись количества столбцов в файл-результат

fprintf(fileC, "%d\n", columnsA);

// Сложение матриц и запись результата в файл-результат

double elementA, elementB;

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

fscanf\_s(fileA, "%lf", &elementA);

fscanf\_s(fileB, "%lf", &elementB);

fprintf(fileC, "%.2f ", elementA + elementB);

}

cout << "Матрицы сложены, результат записан в файл fC.txt." << endl;

// Закрытие файлов

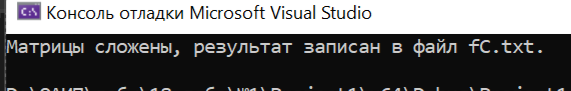
fclose(fileA);

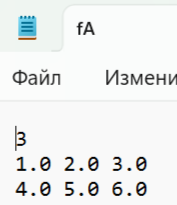
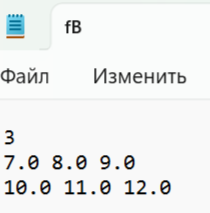
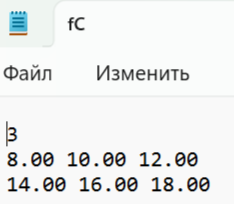
fclose(fileB);

fclose(fileC);

return 0;

}



№2

#include <stdio.h>

int main() {

FILE\* fileF, \* fileG;

int numbers[20]; // Массив для хранения чисел из файла f.txt

// Открытие файлов

fopen\_s(&fileF, "f.txt", "r");

fopen\_s(&fileG, "g.txt", "w");

// Проверка на успешное открытие файлов

if (fileF == NULL || fileG == NULL) {

perror("Ошибка открытия файлов");

return 1;

}

// Чтение чисел из файла f.txt

for (int i = 0; i < 20; i++) {

fscanf\_s(fileF, "%d", &numbers[i]);

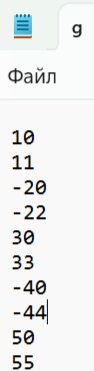
}

// Запись чисел в файл g.txt (положительные, затем отрицательные и т.д.)

for (int i = 0; i < 5; i++) {

fprintf(fileG, "%d\n", numbers[i]);

fprintf(fileG, "%d\n", numbers[i + 10]); // Сдвиг на 10 для отрицательных чисел

 }

std::cout << "Файл g.txt создан" << std::endl;

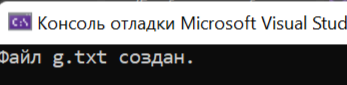
// Закрытие файлов

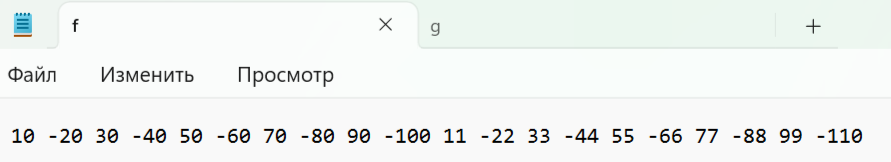
fclose(fileF);

fclose(fileG);

return 0;

}





|  |  |
| --- | --- |
| **3** | 1. Дан файл вещественных чисел, содержащий элементы квадратной матрицы (по строкам), причем начальный элемент файла содержит значение количества столбцов матрицы. Создать новый файл той же структуры, содержащий транспонированную матрицу.  2. Даны три файла целых чисел одинакового размера с именами **NameA**, **NameB** и **NameC**. Создать новый файл с именем **NameD**, в котором чередовались бы элементы исходных файлов с одним и тем же номером: **A0, B0, C0, A1, B1, C1, A2, B2, C2, ...** |

**№1**

#include <stdio.h>

int main() {

FILE\* inputFile, \* outputFile;

int cols;

// Открытие файла с входными данными

inputFile = fopen("input.txt", "r");

if (inputFile == NULL) {

perror("Ошибка открытия входного файла");

return 1;

}

// Чтение количества столбцов из файла

fscanf\_s(inputFile, "%d", &cols);

// Открытие файла для записи результата

outputFile = fopen("output.txt", "w");

if (outputFile == NULL) {

perror("Ошибка открытия выходного файла");

fclose(inputFile);

return 1;

}

// Запись количества столбцов в новый файл

fprintf(outputFile, "%d\n", cols);

// Выделение памяти под матрицу

double\*\* matrix = (double\*\*)malloc(cols \* sizeof(double\*));

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

matrix[i] = (double\*)malloc(cols \* sizeof(double));

}

// Чтение и транспонирование матрицы

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

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

fscanf\_s(inputFile, "%lf", &matrix[i][j]);

}

}

// Запись транспонированной матрицы в новый файл

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

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

fprintf(outputFile, "%.2f ", matrix[j][i]);

}

fprintf(outputFile, "\n");

}

// Освобождение памяти

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

free(matrix[i]);

}

free(matrix);

std::cout << "Транспонированная матрица записана в файл " << outputFile << std::endl;

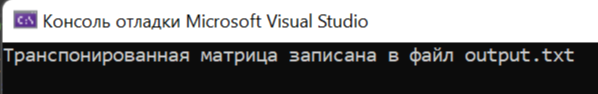
// Закрытие файлов

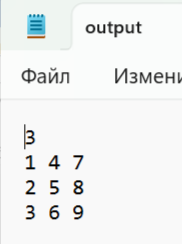
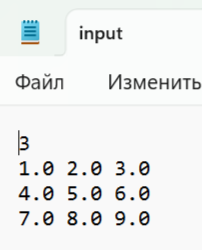
fclose(inputFile);

fclose(outputFile);

return 0;

}

****

****

**№2**

#include <stdio.h>

int main() {

FILE\* fileA, \* fileB, \* fileC, \* fileD;

int size; // Размер файлов

// Открытие файлов

fopen\_s(&fileA, "NameA.txt", "r");

fopen\_s(&fileB, "NameB.txt", "r");

fopen\_s(&fileC, "NameC.txt", "r");

fopen\_s(&fileD, "NameD.txt", "w");

// Проверка на успешное открытие файлов

if (fileA == NULL || fileB == NULL || fileC == NULL || fileD == NULL) {

perror("Ошибка открытия файлов");

return 1;

}

// Чтение размера файлов

fscanf\_s(fileA, "%d", &size);

// Запись размера в новый файл

fprintf(fileD, "%d\n", size);

// Чередование элементов и запись в новый файл

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

int elementA, elementB, elementC;

fscanf\_s(fileA, "%d", &elementA);

fscanf\_s(fileB, "%d", &elementB);

fscanf\_s(fileC, "%d", &elementC);

fprintf(fileD, "%d %d %d ", elementA, elementB, elementC);

}

std::cout << "Файл " << nameD << " успешно создан" << std::endl;

// Закрытие файлов

fclose(fileA);

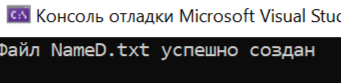
fclose(fileB);

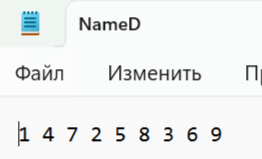
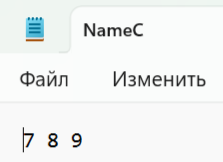
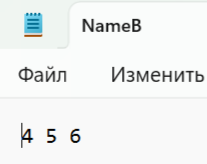
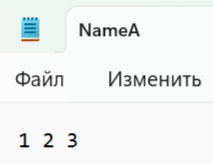
fclose(fileC);

fclose(fileD);

return 0;

}

****

****

|  |  |
| --- | --- |
| **4** | 1. Компоненты файла **fileA** – целые отличные от нуля положительные и отрицательные числа. Получить файл **fileB**, состоящий из положительных чисел.  2. Компоненты файла **f** – целые двухзначные числа. Получить файл **g**, образованный из **f** включением только чисел больших некоторого числа, вводимого с клавиатуры. |

**№1**

#include <stdio.h>

int main() {

FILE\* fileA, \* fileB;

int number;

// Открытие файла с исходными данными

fopen\_s(&fileA, "fileA.txt", "r");

if (fileA == NULL) {

perror("Ошибка открытия файла fileA.txt");

return 1;

}

// Открытие файла для записи положительных чисел

fopen\_s(&fileB, "fileB.txt", "w");

if (fileB == NULL) {

perror("Ошибка открытия файла fileB.txt");

fclose(fileA);

return 1;

}

// Чтение и запись положительных чисел

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

if (number > 0) {

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

}

}

std::cout << "файл " << fileB << " успешно создан" << std::endl;

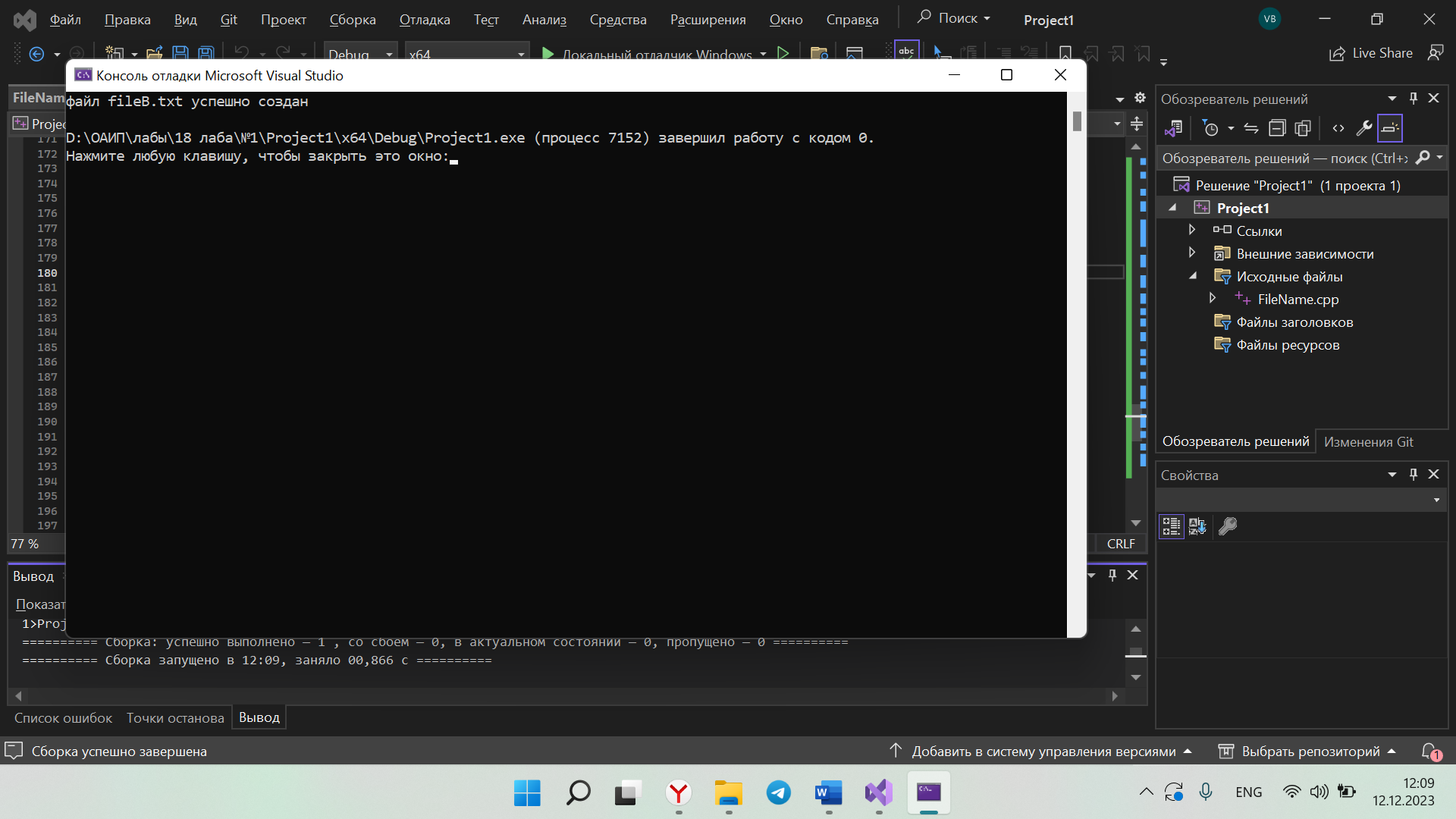
// Закрытие файлов

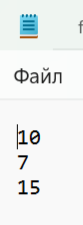
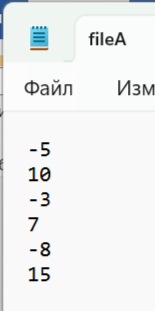
fclose(fileA);

fclose(fileB);

return 0;

}

****

****

**№2**

#include <stdio.h>

#include <iostream>

int main() {

setlocale(LC\_ALL, "rus");

FILE\* fileF, \* fileG;

int threshold, number;

// Получение значения порога с клавиатуры

printf("Введите пороговое значение: ");

scanf\_s("%d", &threshold);

// Открытие файла с исходными данными

fopen\_s(&fileF, "f.txt", "r");

if (fileF == NULL) {

perror("Ошибка открытия файла f.txt");

return 1;

}

// Открытие файла для записи чисел, больших порога

fopen\_s(&fileG, "g.txt", "w");

if (fileG == NULL) {

perror("Ошибка открытия файла g.txt");

fclose(fileF);

return 1;

}

// Чтение и запись чисел, больших порога

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

if (number > threshold) {

fprintf(fileG, "%d\n", number);

}

}

std::cout << "Файл " << fileG << " успешно создан" << std::endl;

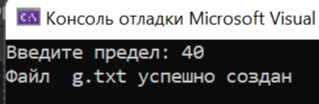
// Закрытие файлов

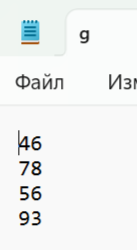
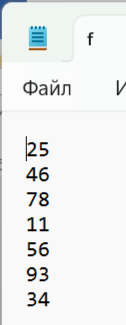
fclose(fileF);

fclose(fileG);

return 0;

}

****

****

|  |  |
| --- | --- |
| **6** | 1. Компоненты файла **fA** – вещественные числа (положительные и отрицательные). Определить и вывести на экран порядковый номер того из них, которое наиболее близко к введенному пользователем целому числу.  2. Создать текстовый файл **F1** не менее, чем из 6 строк, и записать в него информацию. Скопировать в файл **F2** только четные строки из **F1**. |

**№1**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

int main() {

setlocale(LC\_ALL, "rus");

FILE\* fileA;

double targetNumber, currentNumber;

int lineNumber = 0, closestLineNumber = 0;

double closestDistance = -1; // Используем отрицательное значение для инициализации

// Получение значения от пользователя

std::cout << "Введите целое число: ";

std::cin >> targetNumber;

// Открытие файла с исходными данными

fopen\_s(&fileA, "fA.txt", "r");

if (fileA == NULL) {

perror("Ошибка открытия файла fA.txt");

return 1;

}

// Чтение и анализ вещественных чисел из файла

while (fscanf\_s(fileA, "%lf", &currentNumber) != EOF) {

lineNumber++;

// Определение близости текущего числа к введенному

double distance = abs(targetNumber - currentNumber);

// Если это ближайшее число, обновляем переменные

if (closestDistance == -1 || distance < closestDistance) {

closestDistance = distance;

closestLineNumber = lineNumber;

}

}

// Выводим результат на экран

std::cout << "Cамое близкое число к " << targetNumber << " это " << currentNumber

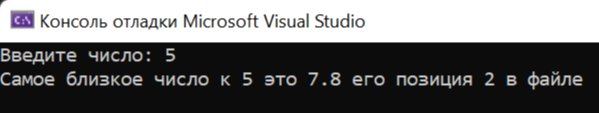
<< " его позиция " << closestLineNumber << " в файле" << std::endl;

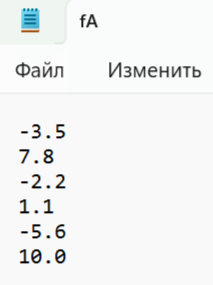
// Закрытие файла

fclose(fileA);

return 0;

}

****

****

**№2**

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE\* fileF1, \* fileF2;

char line[256]; // Предполагаем, что строка не будет длиннее 255 символов

int lineNumber = 0;

// Создание и запись в файл F1.txt

fopen\_s(&fileF1, "F1.txt", "w");

if (fileF1 == NULL) {

perror("Ошибка открытия файла F1.txt");

return 1;

}

fprintf(fileF1, "Первая строка\n");

fprintf(fileF1, "Вторая строка\n");

fprintf(fileF1, "Третья строка\n");

fprintf(fileF1, "Четвертая строка\n");

fprintf(fileF1, "Пятая строка\n");

fprintf(fileF1, "Шестая строка\n");

// Закрытие файла F1.txt

fclose(fileF1);

// Копирование четных строк в F2.txt

fopen\_s(&fileF1, "F1.txt", "r");

fopen\_s(&fileF2, "F2.txt", "w");

if (fileF1 == NULL || fileF2 == NULL) {

perror("Ошибка открытия файлов F1.txt или F2.txt");

return 1;

}

// Чтение и копирование четных строк

while (fgets(line, sizeof(line), fileF1) != NULL) {

lineNumber++;

if (lineNumber % 2 == 0) {

fprintf(fileF2, "%s", line);

}

}

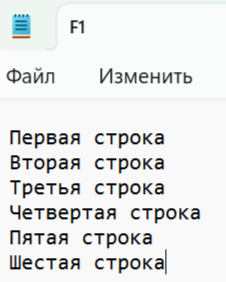
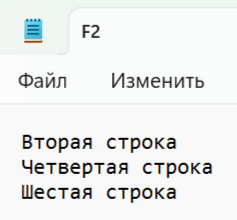
// Закрытие файлов

fclose(fileF1);

fclose(fileF2);

return 0;

}

**** ****