**Московский авиационный институт**

**(национальный исследовательский институт)**

Институт №3

Системы управления, информатики и электроэнергетики

Кафедра 304

**Отчет по лабораторной работе**

**по учебной дисциплине**

**«Компьютерные технологии»**

**на тему:**

***«Двумерные массивы»***

*Группа:* ***М30-109Б-19***

*Вариант:* ***№9***

*Выполнил:*

***Кравченко Д.С.***

***Кузнецов И.И.***

*Приняли:*

***Давыдкина Е.А.***

***Секретарев В.Е.***

*Москва 2020*

**Содержание**

Задание\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

Блок-схема\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4

Псевдокод\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_12

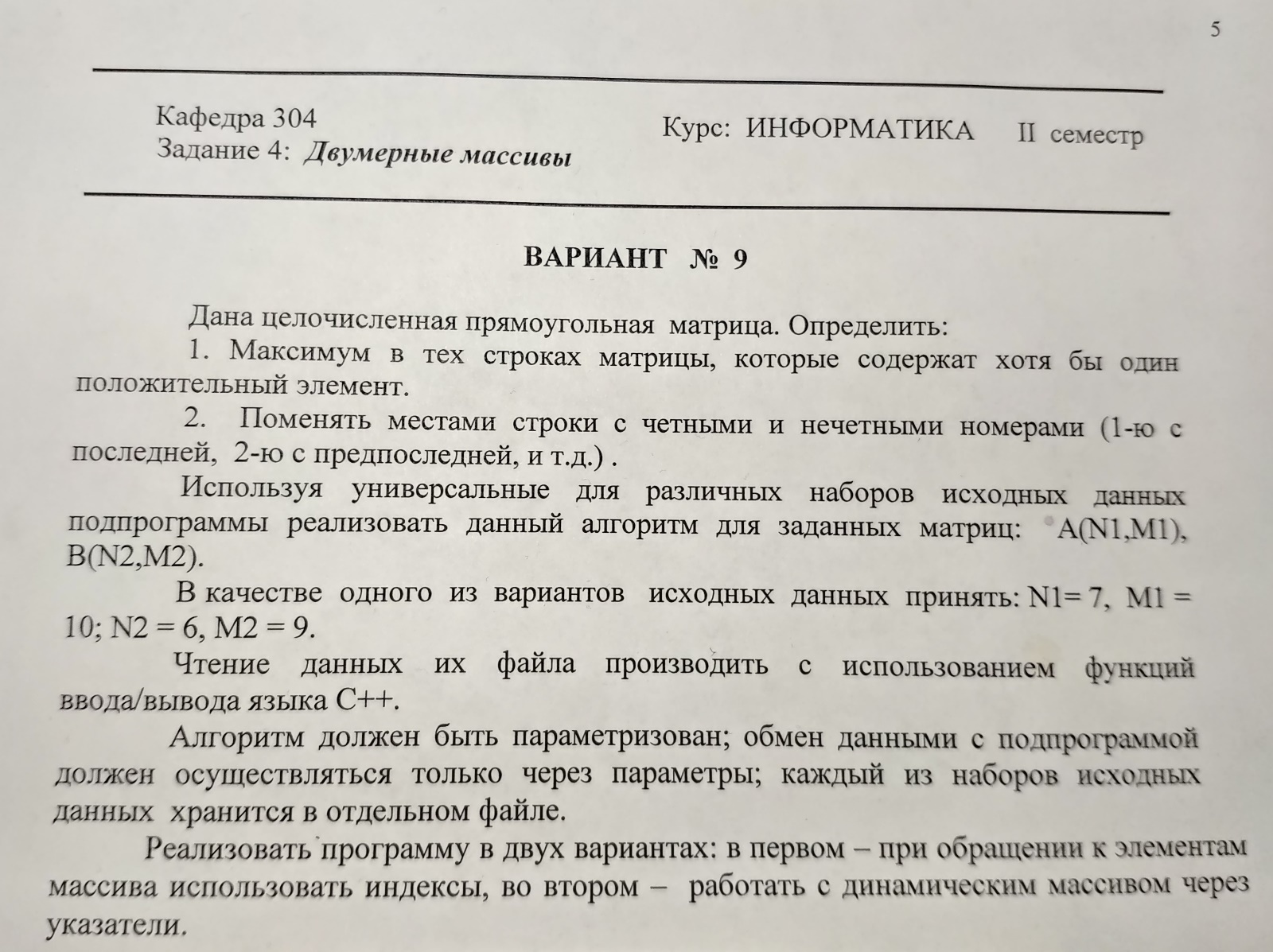
Код программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_19

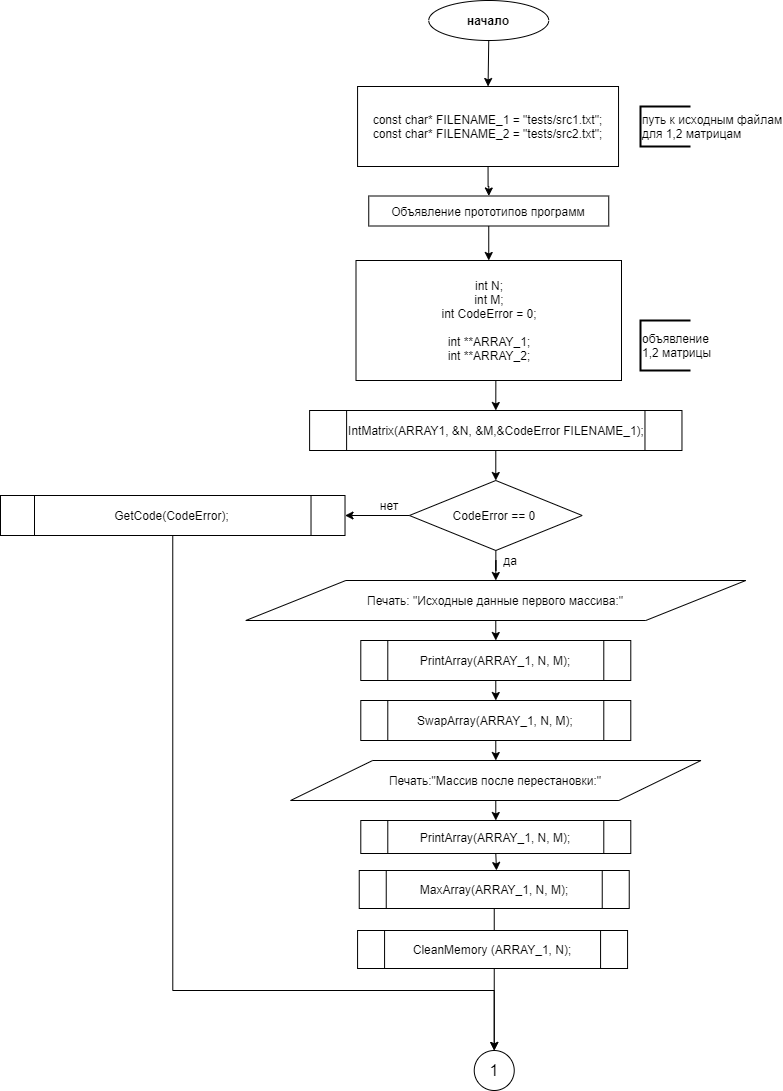
Некорректные тесты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_25

Корректные тесты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_30

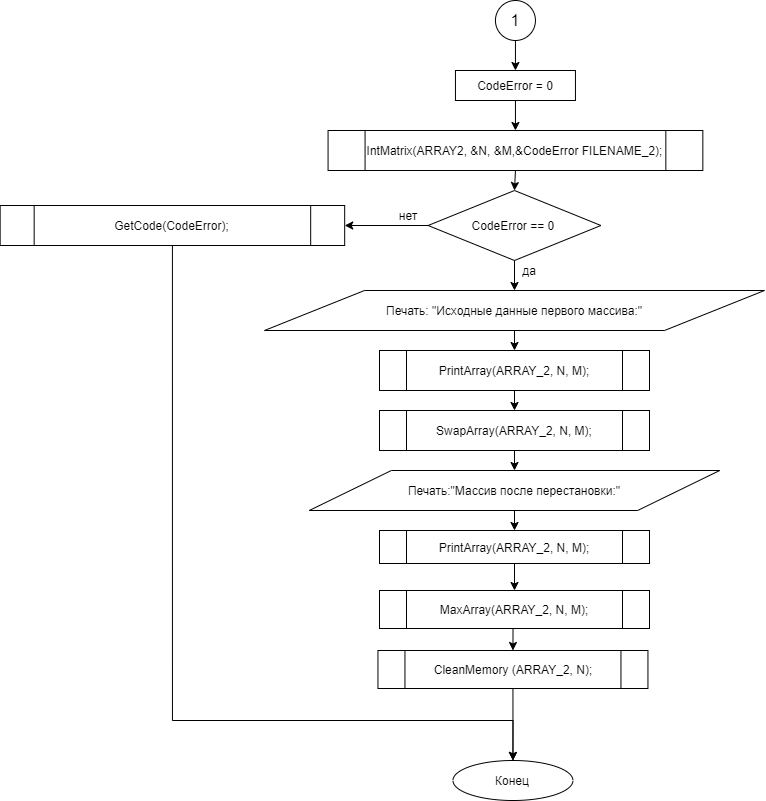
Вывод\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_38

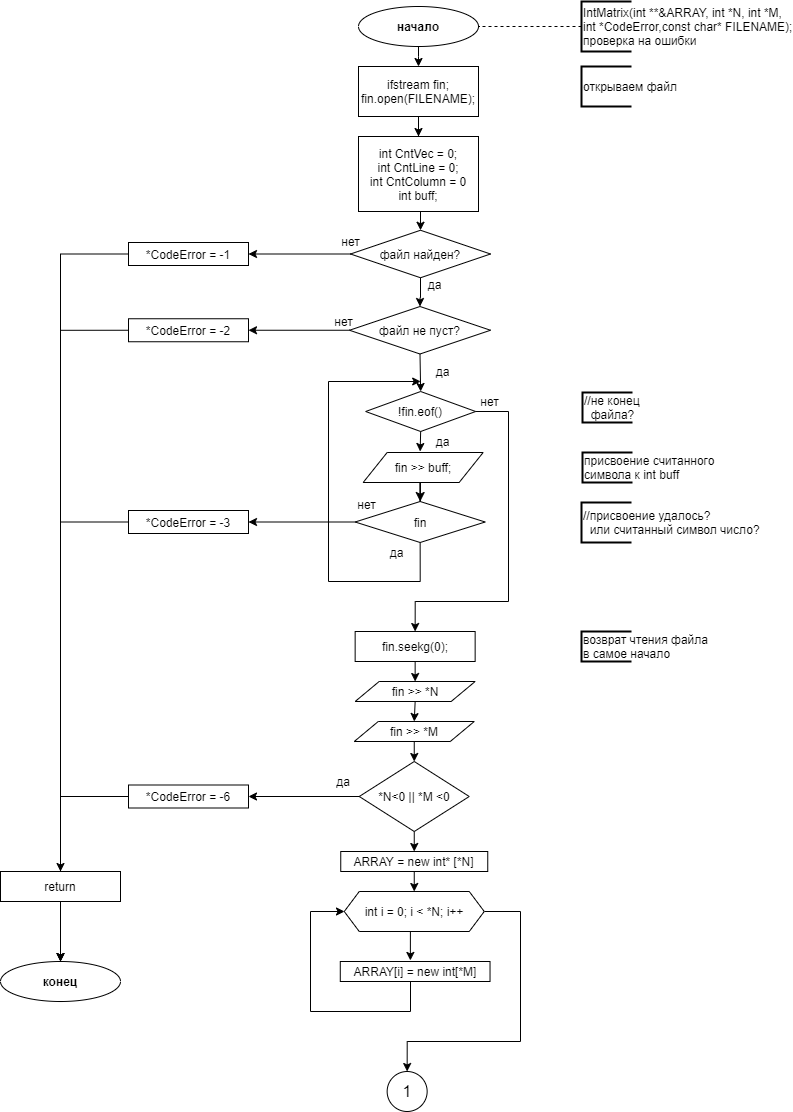
**Задание**

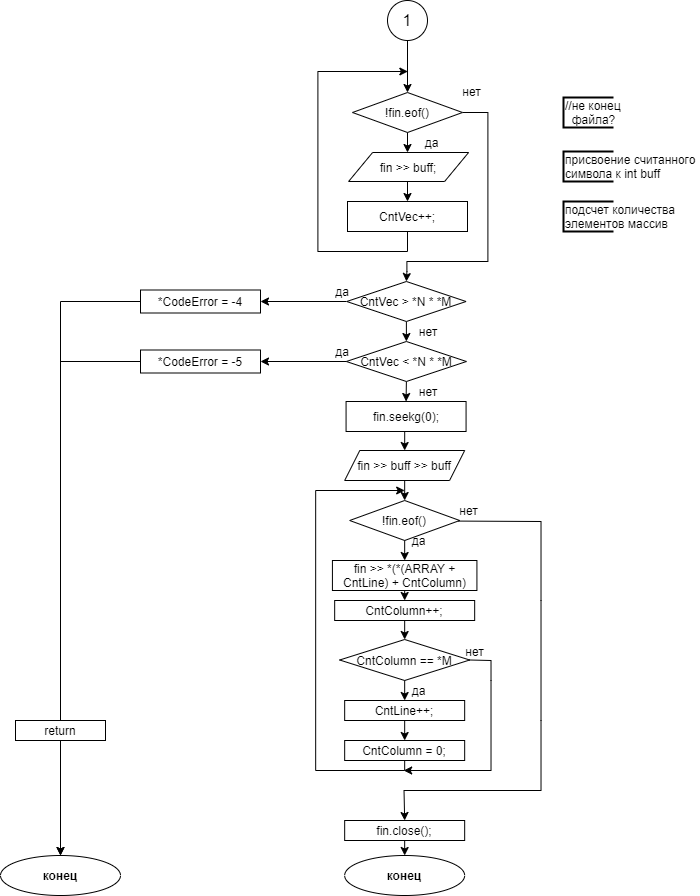


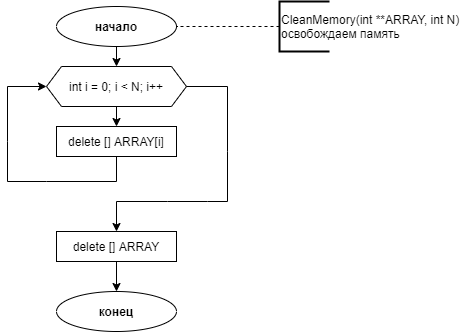


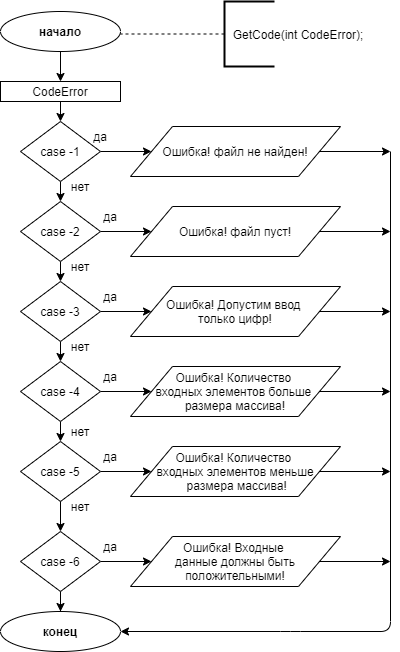
**Блок-схема**

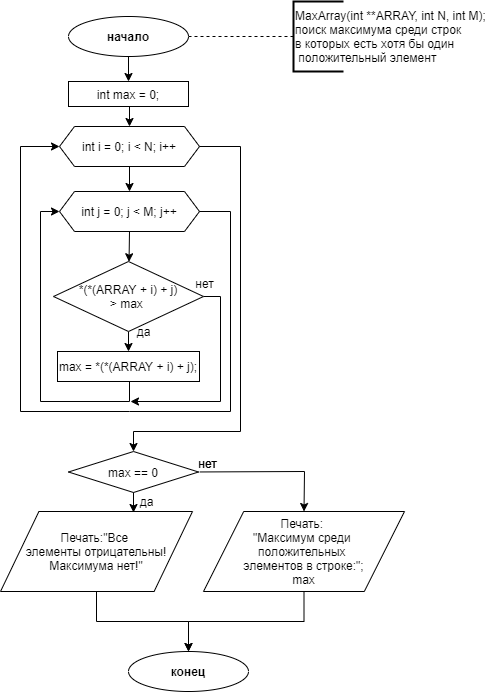


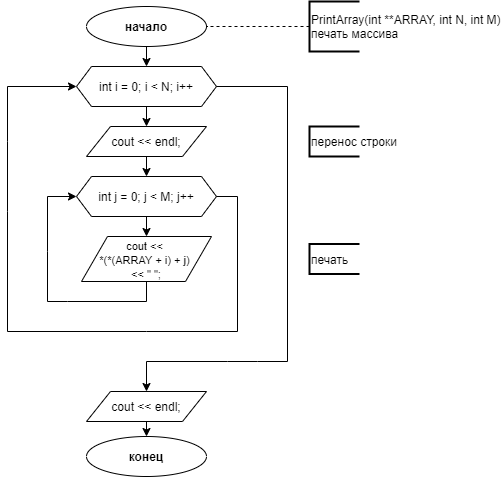


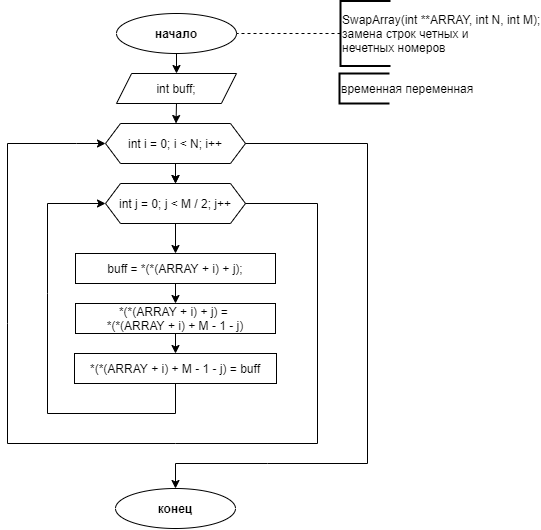










****

**Псевдокод**

**АЛГОРИТМ**

*Глобальные констатные переменные:*

*Символьные:*

**FILENAME\_1** ="tests\_1/src12.txt"; //путь к исходным файлам для 1 матрицы

**FILENAME\_2** = "tests\_2/src12.txt"; //путь к исходным файлам для 2 матрицы

**ОСНОВНЫЙ АЛГОРИТМ:**

**НАЧАЛО**

*Целочисленные переменные:*

**N, M** – размеры матрицы

**CodeError** – код ошибки

\*\***ARRAY\_1** - объявление первой матрицы

\*\***ARRAY\_2** - объявление второй матрицы

Вызов **IntMatrix(ARRAY\_1, &N, &M, &CodeError, FILENAME\_1)**

**ЕСЛИ CodeError == 0**

**НАЧАЛО**

**Вывод:** *Исходные данные первого массива:*

**Вызов** PrintArray(ARRAY\_1, N, M)

**Вызов** SwapArray(ARRAY\_1, N, M)

**Вывод:** *Массив после перестановки:*

**Вызов** PrintArray(ARRAY\_1, N, M)

**Вызов** MaxArray(ARRAY\_1, N, M)

**Вызов** CleanMemory(ARRAY\_1, N)

**КОНЕЦ**

**ИНАЧЕ**

**НАЧАЛО**

**Вызов** GetCode(CodeError)

**КОНЕЦ**

**КОНЕЦ**

**АЛГОРИТМ** **IntMatrix**(int **\*\*\*ARRAY**, int **\*N**, int **\*M**, int **\*CodeError**, const char**\* FILENAME**)

**НАЧАЛО**

*Целочисленные переменные:*

**CntVec = 0 -** счетчик реального размера массива

**CntLine = 0 -** счетчик для ввода

**CntColumn = 0 -** счетчик для ввода

**buff -** буффер для проверки корректности вводимых данных

**ЕСЛИ** файл пуст

**НАЧАЛО**

\*CodeError = -1

Выход из функции

**КОНЕЦ**

**ЕСЛИ** файл пуст

**НАЧАЛО**

\*CodeError= -2

Выход из функции

**КОНЕЦ**

**ПОКА** символы не закончились

**НАЧАЛО ЦИКЛА**

**Ввод** очередного символа в буффер

**ЕСЛИ** очередной символ не цифра

**НАЧАЛО**

\*CodeError = -3

Выход из функции  
 **КОНЕЦ**

**КОНЕЦ ЦИКЛА**

Ввод \***N**

Ввод \***M**

**ЕСЛИ** ((\*N < 0) || (\*M < 0))

**НАЧАЛО**

\*CodeError = -6

Выход из функции

**КОНЕЦ**

*Выделение памяти под двумерный динамический массив*

**ПОКА** символы не закончились

**НАЧАЛО ЦИКЛА**

**Ввод** очередного символа в буффер

CntVec++

**КОНЕЦ ЦИКЛА**

**ЕСЛИ** CntVec > \*N \* \*M

**НАЧАЛО**

\*CodeError = -4

Выход из функции  
**КОНЕЦ**

**ЕСЛИ** CntVec < \*N \* \*M

**НАЧАЛО**

\*CodeError = -5

Выход из функции  
**КОНЕЦ**

**Ввод** buff, buff

**ПОКА** символы не закончились

**НАЧАЛО ЦИКЛА**

**Ввод** очередного символа в ARRAY[CntLine][CntColumn]

CntColumn++

**ЕСЛИ** CntColumn == M

**НАЧАЛО**

CntLine++

CntColumn = 0

**КОНЕЦ**

**КОНЕЦ ЦИКЛА**

**КОНЕЦ**

**АЛГОРИТМ GetCode**(int **CodeError**)

**НАЧАЛО**

**Ввод** CodeError

**ЕСЛИ** CodeError = -1

**НАЧАЛО**

**Вывод:** *Ошибка! файл не найден*

**КОНЕЦ**

**ЕСЛИ** CodeError = -2

**НАЧАЛО**

***Вывод:*** *Ошибка! файл пуст!*

**КОНЕЦ**

**ЕСЛИ** CodeError = -3

**НАЧАЛО**

***Вывод:*** *Ошибка! Допустим ввод только цифр!*

**КОНЕЦ**

**ЕСЛИ** CodeError = -4

**НАЧАЛО**

**Вывод:** *Ошибка! Количество входных элементов больше размера массива!*

**КОНЕЦ**

**ЕСЛИ** CodeError = -5

**НАЧАЛО**

**Вывод:** *Ошибка! Количество входных элементов меньше размера массива!*

**КОНЕЦ**

**ЕСЛИ** CodeError = -6

**НАЧАЛО**

**Вывод:** *Ошибка! Входные данные должны быть положительными!*

**КОНЕЦ**

**КОНЕЦ**

**АЛГОРИТМ SwapArray**(int **\*\*ARRAY**, int **N**, int **M**)

**НАЧАЛО**

*Целочисленные переменные:*

**buff -** буффер для обмена

**НАЧАЛО ЦИКЛА** по **i** до **N**

i++

**НАЧАЛО ЦИКЛА** по **j** до **M/2**

j++

buff = \*(\*(ARRAY + i) + j)

\*(\*(ARRAY + i) + j) = \*(\*(ARRAY + i) + M - 1 - j)

\*(\*(ARRAY + i) + M - 1 - j) = buff

**КОНЕЦ ЦИКЛА**

**КОНЕЦ ЦИКЛА**

**КОНЕЦ**

**АЛГОРИТМ MaxArray**(int **\*\*ARRAY**, int **N**, int **M**)

**НАЧАЛО**

*Целочисленная переменная:*

**max** = 0; //т.к. просят найти максимум среди положительных чисел ставим изначальное значение максимума = 0

**НАЧАЛО ЦИКЛА** по **I** до **N**

i++

**НАЧАЛО ЦИКЛА** по **j** до **M**

j++

**ЕСЛИ** (\*(\*(ARRAY + i) + j) > max)

**НАЧАЛО**

max = \*(\*(ARRAY + i) + j)

**КОНЕЦ**

**КОНЕЦ ЦИКЛА**

**КОНЕЦ ЦИКЛА**

**ЕСЛИ** max == 0

**НАЧАЛО**

**Вывод:** Все элементы отрицательны! Максимума нет!

**КОНЕЦ**

**ИНАЧЕ**

**НАЧАЛО**

**Вывод:** Максимум среди положительных элементов в строке: max

**КОНЕЦ**

**КОНЕЦ**

**АЛГОРИТМ PrintArray**(int **\*\*ARRAY**, int **N**, int **M**)

**НАЧАЛО**

**НАЧАЛО ЦИКЛА** по **i** до **N**

i++

**НАЧАЛО ЦИКЛА** по **j** до **M**

j++

**Вывод:** \*(\*(ARRAY + i) + j)

**КОНЕЦ ЦИКЛА**

**КОНЕЦ ЦИКЛА**

**КОНЕЦ**

**АЛГОРИТМ CleanMemory**(int **\*\*ARRAY**, int **N**)

**НАЧАЛО**

**НАЧАЛО ЦИКЛА** по **j** до **M**

i++

delete[]ARRAY[i]

**КОНЕЦ ЦИКЛА**

delete[] ARRAY

**КОНЕЦ**

**Код программы**

#include <string>

#include <fstream>

#include <iostream>

using namespace std;

const char\* FILENAME\_1 = "../tests\_1/src11.txt"; //путь к исходным файлам для 1 матрицы

const char\* FILENAME\_2 = "../tests\_1/src12.txt"; //путь к исходным файлам для 2 матрицы

// \* -------------------------------------------------------------------------------------------------------- \*

// Н Е К О Р Р Е К Т Н Ы Е Т Е С Т Ы

// src0.txt - тест на отсутствие файла

// src1.txt - тест на пустой файл

// src2.txt - тест на неккоректные символы при вводе (буквы)

// src3.txt - тест на количество входных элементов > размера матрицы

// src4.txt - тест на количество входных элементов < размера матрицы

// src13.txt - тест на отрицательные размеры матрицы (N<0)

// src14.txt - тест на отрицательные размеры матрицы (M<0)

//

// К О Р Р Е К Т Н Ы Е Т Е С Т Ы

// src5.txt - тест на размер матрицы = количество входных элементов

//

// src6.txt - тест без положительных элементов в матрице

// src7.txt - тест с одним положительным элементов в матрице

// src8.txt - тест с несколькими одинаковыми максимумами

//

// src9.txt - тест для проверки перестановок с нечетным количеством столбцов

// src10.txt - тест для проверки перестановок с четным количеством столбцов

//

// src11.txt - тест с размерами матрицами: ARRAY1 - 5 8; ARRAY2 - 4 7

// src12.txt - тест с размерами матрицами: ARRAY1 - 6 9; ARRAY2 - 3 6

//

// \* -------------------------------------------------------------------------------------------------------- \*

void PrintArray(int\*\* ARRAY, int N, int M); //печать массива

void MaxArray(int\*\* ARRAY, int N, int M); //поиск максимума среди строк в которых есть хотя бы один положительный элемент

void SwapArray(int\*\* ARRAY, int N, int M); //замена строк четных и нечетных номеров

void GetCode(int code);

void CleanMemory(int\*\* ARRAY, int N);

void IntMatrix(int\*\*\* ARRAY, int\* N, int\* M, int\* CodeError, const char\* FILENAME); //проверка на ошибки и ввод

int main() {

system("color F0"); //задаем цвет консоли

setlocale(LC\_ALL, "Russian"); //задаем язык

int N; //размеры матрицы

int M; //размеры матрицы

int CodeError = 0; //код ошибки

int\*\* ARRAY\_1; //объявление первой матрицы

int\*\* ARRAY\_2; //объявление второй матрицы

IntMatrix(&ARRAY\_1, &N, &M, &CodeError, FILENAME\_1);

if (CodeError == 0) {

cout << "Исходные данные первого массива: ";

PrintArray(ARRAY\_1, N, M);

SwapArray(ARRAY\_1, N, M);

cout << endl << "Массив после перестановки столбцов(первый - последний, второй - предпоследний и тп): ";

PrintArray(ARRAY\_1, N, M);

MaxArray(ARRAY\_1, N, M);

cout << endl << "-------------------------------------------------" << endl << endl;

CleanMemory(ARRAY\_1, N);

}

else {

GetCode(CodeError);

}

CodeError = 0;

IntMatrix(&ARRAY\_2, &N, &M, &CodeError, FILENAME\_2);

if (CodeError == 0) {

cout << "Исходные данные первого массива: ";

PrintArray(ARRAY\_2, N, M);

SwapArray(ARRAY\_2, N, M);

cout << endl << "Массив после перестановки столбцов(первый - последний, второй - предпоследний и тп): ";

PrintArray(ARRAY\_2, N, M);

MaxArray(ARRAY\_2, N, M);

cout << endl << "-------------------------------------------------" << endl << endl;

CleanMemory(ARRAY\_2, N);

}

else {

GetCode(CodeError);

}

}

void CleanMemory(int\*\* ARRAY, int N) { //освобождаем память

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

delete[] ARRAY[i];

}

delete[] ARRAY;

}

void PrintArray(int\*\* ARRAY, int N, int M) { //печать массива

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

cout << endl;

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

cout << \*(\*(ARRAY + i) + j) << " ";

}

}

cout << endl;

}

void MaxArray(int\*\* ARRAY, int N, int M) {

cout << endl;

int max = 0; //т.к. просят найти максимум среди положительных чисел ставим изначальное значение максимума = 0

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

for (int j = 0; j < M; j++) { //если некоторый элемент больше изначального максимума, то его сохраняем

if (\*(\*(ARRAY + i) + j) > max) { //и после нам не обязательно проверять положительный ли след. элемент,

max = \*(\*(ARRAY + i) + j); //так как новый записанный максимум уже больше нуля

}

}

}

if (max == 0) {

cout << "Все элементы отрицательны! Максимума нет!";

}

else {

cout << "Максимум среди положительных элементов в строке: " << max;

}

cout << endl;

}

void SwapArray(int\*\* ARRAY, int N, int M) {

int buff; //временная переменная

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

for (int j = 0; j < M / 2; j++) {

buff = \*(\*(ARRAY + i) + j);

\*(\*(ARRAY + i) + j) = \*(\*(ARRAY + i) + M - 1 - j);

\*(\*(ARRAY + i) + M - 1 - j) = buff;

}

}

}

void GetCode(int CodeError) {

switch (CodeError)

{

case -1:

cout << "Ошибка! файл не найден" << endl << endl;

break;

case -2:

cout << "Ошибка! файл пуст!" << endl << endl;

break;

case -3:

cout << "Ошибка! Допустим ввод только цифр!" << endl << endl;

break;

case -4:

cout << "Ошибка! Количество входных элементов больше размера массива!" << endl << endl;

break;

case -5:

cout << "Ошибка! Количество входных элементов меньше размера массива!" << endl << endl;

break;

case -6:

cout << "Ошибка! Входные данные должны быть положительными!" << endl << endl;

break;

}

}

void IntMatrix(int\*\*\* ARRAY, int\* N, int\* M, int\* CodeError, const char\* FILENAME) {

ifstream fin; //создаем объект класса ifstream

fin.open(FILENAME); //открываем файл

int CntVec = 0; //счетчик реального размера массива

int CntLine = 0; //счетчик для ввода

int CntColumn = 0; //счетчик для ввода

int buff; // буффер для проверки корректности вводимых данных

if (!fin) // если файл не найден

{

\*CodeError = -1;

return;

}

if (fin.peek() == EOF) //если найденный файл пуст

{

\*CodeError = -2;

return;

}

while (!fin.eof()) { //проверка на ввод цифр

fin >> buff;

if (!fin) {

\*CodeError = -3;

return;

}

}

fin.seekg(0); //возврат чтения файла на самое начало

fin >> \*N; //считываем размеры

fin >> \*M; //считываем размеры

if ((\*N < 0) || (\*M < 0)) {

\*CodeError = -6;

return;

}

\*ARRAY = new int\* [\*N];

for (int i = 0; i < \*N; i++) {

\*(\*ARRAY + i) = new int[\*M];

}

while (!fin.eof()) { //чтение элементов массива, пока элементы не закончатся

fin >> buff;

CntVec++; //подсчет количества элементов массив

}

if (CntVec > \* N \* \*M) { //проверка на выход за границы массива

\*CodeError = -4;

return;

}

if (CntVec < \*N \* \*M) { //проверка на недостаток элементов

\*CodeError = -5;

return;

}

fin.seekg(0); //возврат чтения файла на самое начало

fin >> buff >> buff; //так как вернулись на самое начало, записываем в буффер размеры

while (!fin.eof()) {

fin >> \*(\*(\*ARRAY + CntLine) + CntColumn); //чтение массива из файла

CntColumn++;

if (CntColumn == \*M) {

CntLine++;

CntColumn = 0;

}

}

fin.close(); // закрываем файл

}

**Тесты**

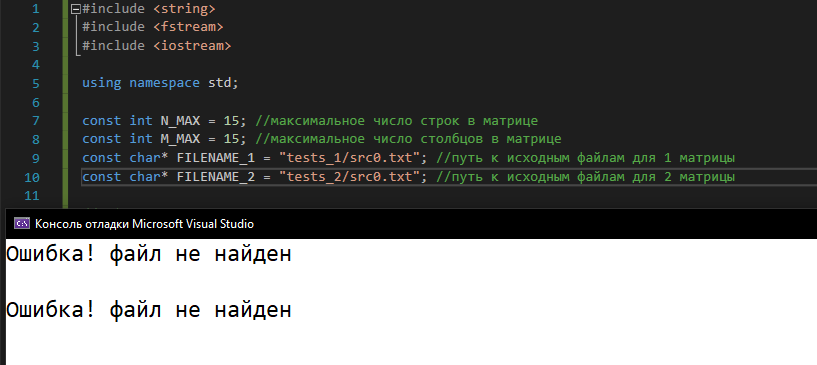
**I Некорректные тесты:**

**Тест №0**

**Цель:** проверить работу программы при условии, файлы не найдены

**Ожидаемый результат:** Вывод сообщения об ошибке.

**Полученный результат:**



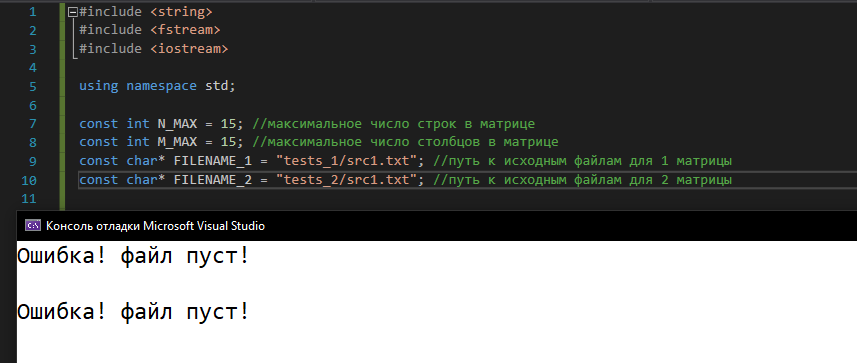
**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Тест №1**

**Цель:** проверить работу программы при условии, что любой из файлов пуст

**Ожидаемый результат:** Вывод сообщения об ошибке.

**Полученный результат:**



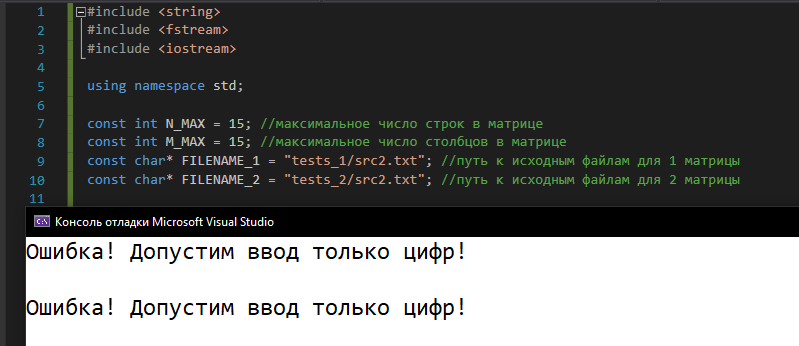
**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Тест №2**

**Цель:** проверить работу программы при условии, что в файле некорректные символы при вводе (буквы)

**Ожидаемый результат:** Вывод сообщения об ошибке.

**Полученный результат:**



**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

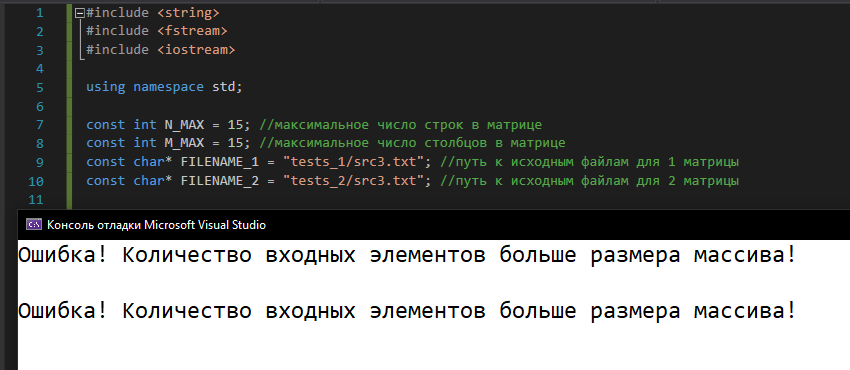
**Тест №3**

**Цель:** проверить работу программы при условии, что

количество входных элементов> размера матрицы

**Ожидаемый результат:** Вывод сообщения об ошибке.

**Полученный результат:**



**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

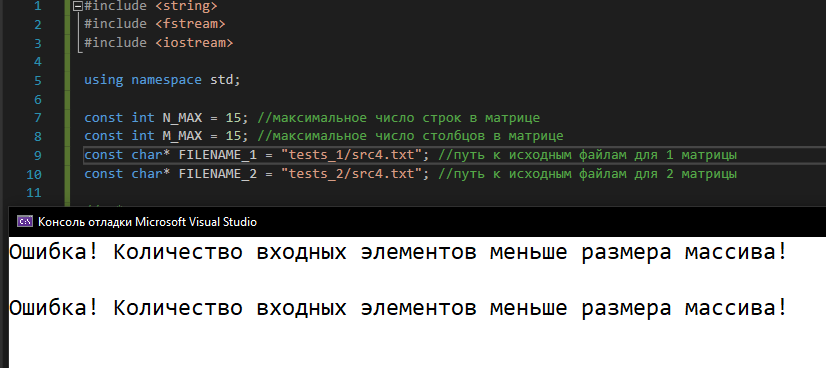
**Тест №4**

**Цель:** проверить работу программы при условии, что

количество входных элементов < размера матрицы

**Ожидаемый результат:**Вывод сообщения об ошибке.

**Полученный результат:**



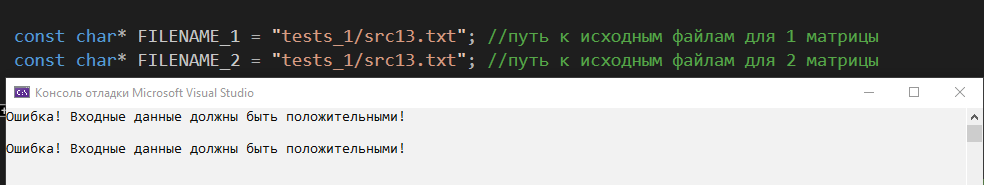
**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Тест №5**

**Цель:** проверить работу программы при условии, что размеры матрицы N < 0

**Ожидаемый результат:** Вывод сообщения об ошибке.

**Полученный результат:**

****

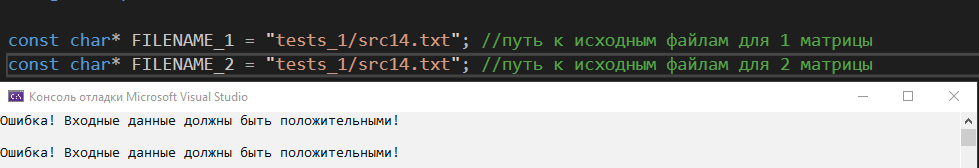
**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Тест №6**

**Цель:** проверить работу программы при условии, что размеры матрицы M < 0

**Ожидаемый результат:** Вывод сообщения об ошибке.

**Полученный результат:**

****

**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

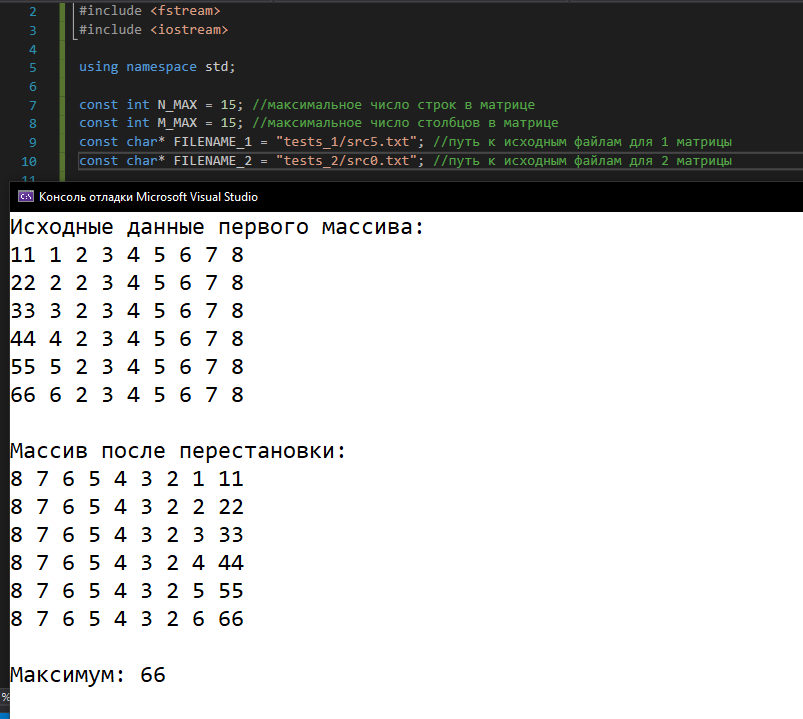
**II Корректные тесты:**

**Тест №7**

**Цель:** проверить работу программы при условии, что размер матрицы = количество входных элементов

**Ожидаемый результат:** Осуществление перестановки и нахождения максимума

**Полученный результат:**



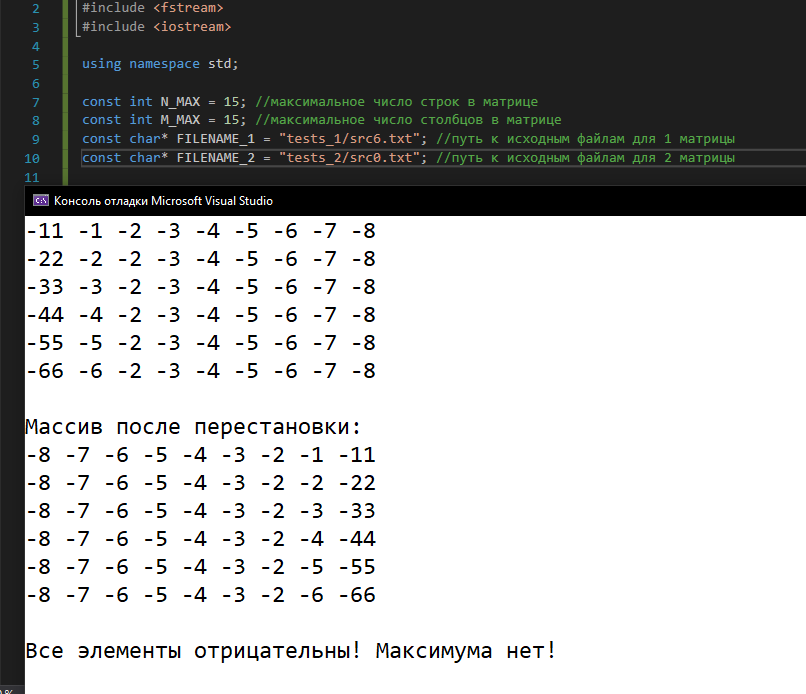
**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Тест №8**

**Цель:** проверить работу программы при условии, что в матрице нет положительных элемент

**Ожидаемый результат:** Осуществление перестановки без нахождения максимума

**Полученный результат:**



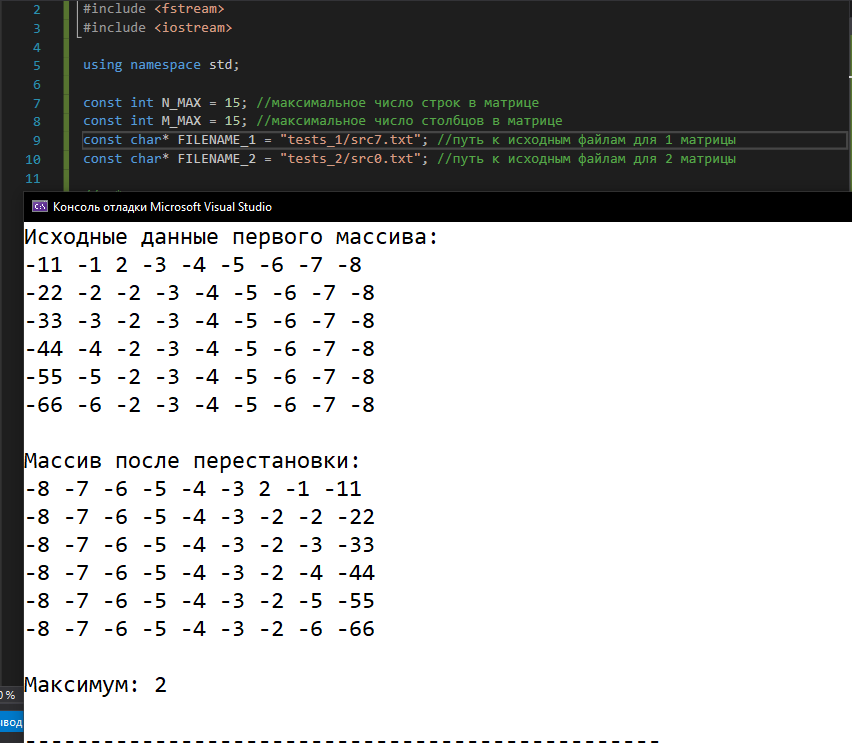
**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Тест №9**

**Цель:** проверить работу программы при условии, что в матрице присутствует только один положительный элемент

**Ожидаемый результат:** Осуществление перестановки и нахождения максимума – того самого единственного положительного элемента

**Полученный результат:**



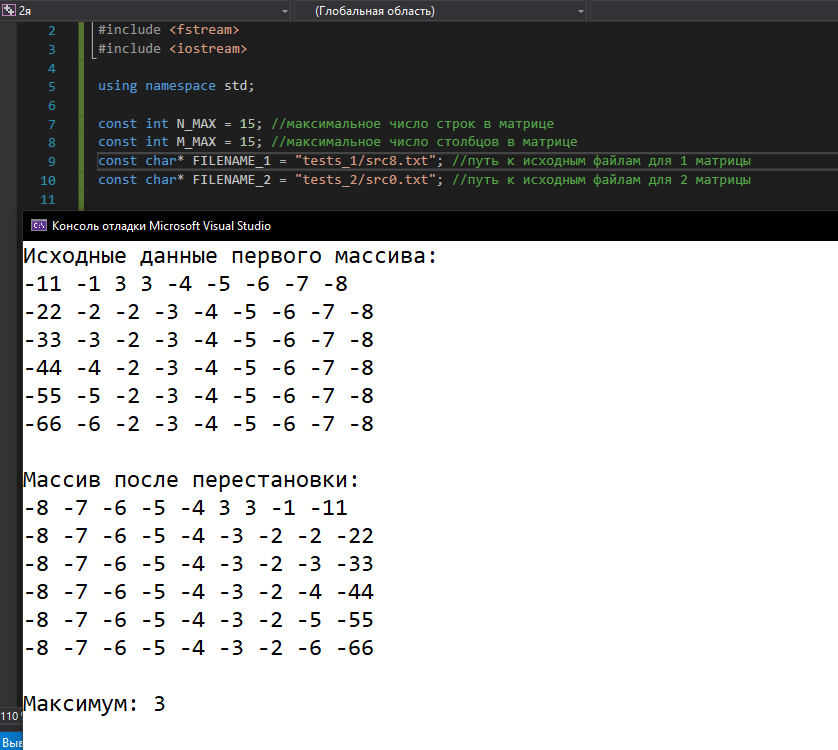
**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Тест №10**

**Цель:** проверить работу программы при условии, что в матрице присутствует несколько одинаковых максимальных положительных элементов

**Ожидаемый результат:** Осуществление перестановки и нахождения максимума

**Полученный результат:**



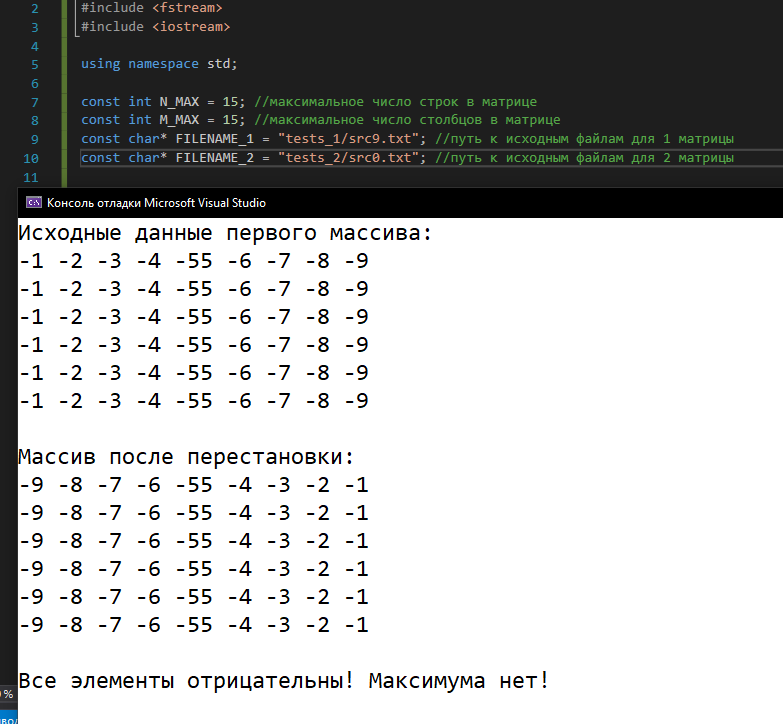
**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Тест №11**

**Цель:** проверить работу программы при условииперестановок с нечетным количеством столбцов

**Ожидаемый результат:** Осуществление перестановки

**Полученный результат:**



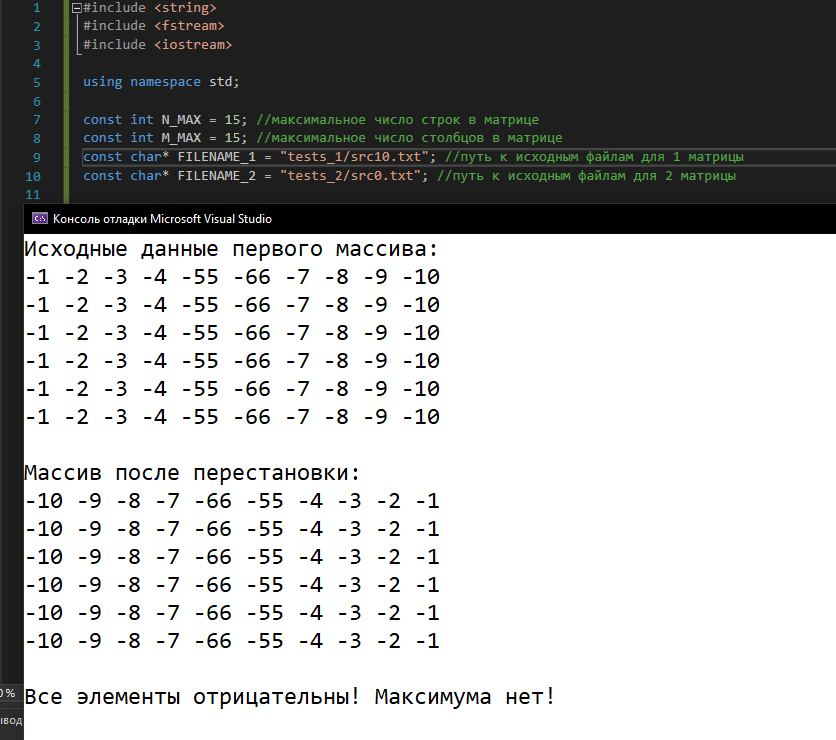
**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Тест №12**

**Цель:** проверить работу программы при условииперестановок с четным количеством столбцов

**Ожидаемый результат:** Осуществление перестановки

**Полученный результат:**



**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

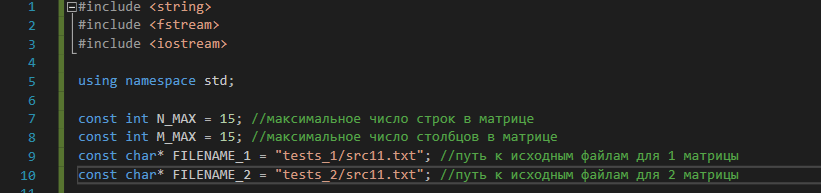
**Тест №13**

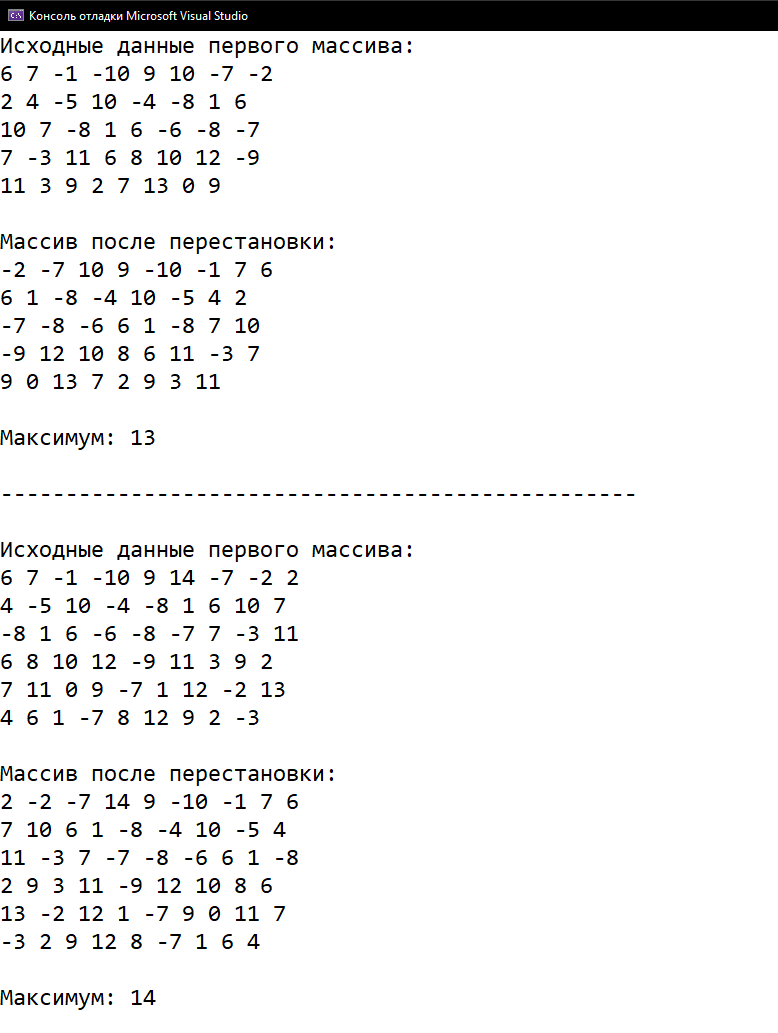
**Цель:** проверить работу программы с размерами матрицами: ARRAY1 - 5 8;

ARRAY2 - 4 7

**Ожидаемый результат:** Осуществление перестановки и нахождения максимума

**Полученный результат:**





**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

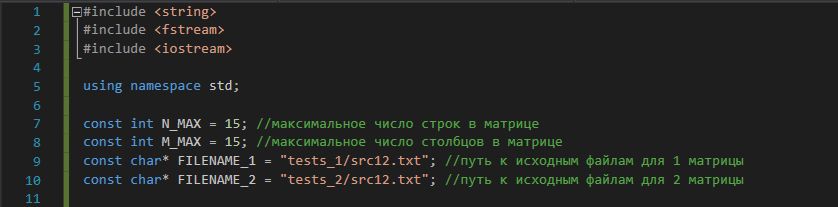
**Тест №14**

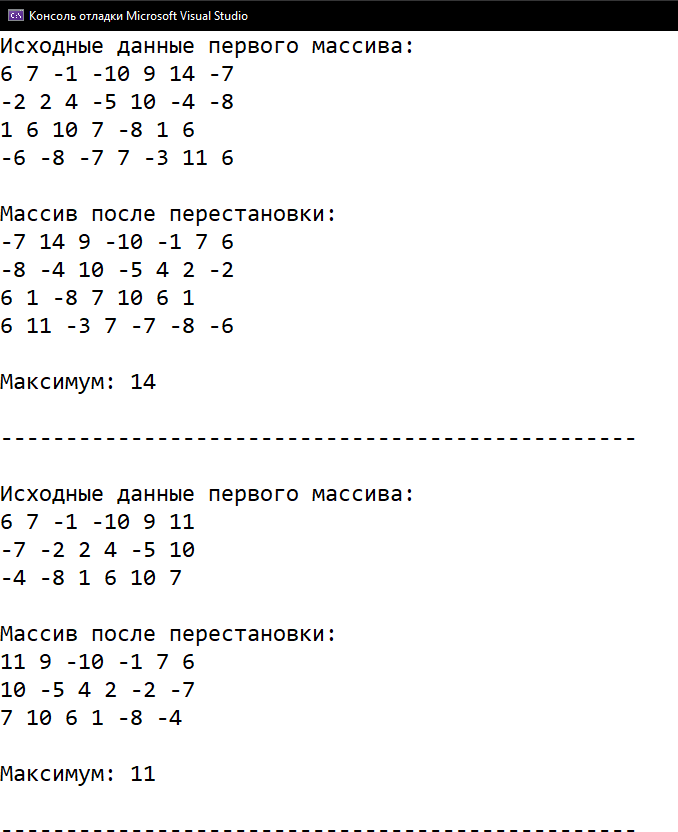
**Цель:** проверить работу программы с размерами матрицами: ARRAY1 - 6 9;

ARRAY2 - 3 6

**Ожидаемый результат:** Осуществление перестановки и нахождения максимума

**Полученный результат:**





**Вывод:** Полученные результаты равны ожидаемым.Тест не выявил ошибок. Работа программы корректна.

**Вывод**

Разработка программы успешно завершена, так как:

**1)** Полученные результаты совпадают с ожидаемыми.

**2)** Набор тестов считаем полным.