МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра систем автоматизированного проектирования

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Программирование»
Тема: «Файлы для ввода-вывода данных, двумерные массивы Функции»

Студентка гр. 3352 _____ Калюжная М.И.

Преподаватель _____ Калмычков В.А.

Санкт-Петербург
2023

Исходная формулировка

Считать матрицу размера nxn с файла и вывести ее по спирали против часовой стрелки.

Дана квадратная матрица порядка n. Начиная с элемента A0,0, вывести все ее элементы по спирали, двигаясь: б) против часовой стрелки: весь 1й столбец и оставшиеся элементы последней строки, в обратном порядке все оставшиеся элементы последнего столбца и затем 1й строки, оставшиеся элементы 2го столбца и т.д.

Формальная постановка задачи

Создаем 2 массива float A[MAX_N][MAX_N] и float B[MAX_N][MAX_N]

Открываем файл input.txt считываем размер матрицы N и считываем саму матрицу в массив A[i] построчно с проверкой на полноту строк (в противном случае ранг квадратной матрицы будет понижен до размера, удовлетворяющемго условиям, т. е. до максимально возможного, при котором матрица будет квадратной).

Затем считаем массив A по спирали (обработав функцией process) и запишем его в массив B

Принцип считывания по спирали:

1. Сначала фиксируется самый первый столбец матрицы A и программа проходится сверху вниз итерацией (++i) по строкам (от top к bottom), записывая значения в массив B по принципу: B[counter / N][counter % N] = A[i][left], при этом идет увеличение на 1 счетчика counter каждую итерацию. При достижении нижней границы матрицы маркер левой границы left увеличивается на 1 (левая граница сдвигается вправо). После цикла проверяются все маркеры на выполнение условия (left > right) && (bottom < top). *(process_top_to_bottom)*
2. Затем обработаем самую нижнюю строку массива A [bottom], пройдя итерацией (++i) слева направо, записывая значения в массив B по принципу: B[counter / N][counter % N] = A[bottom] [i], при этом идет увеличение на 1 счетчика counter каждую итерацию. При достижении правой границы матрицы маркер нижней границы bottom уменьшается на 1 (нижняя граница сдвигается вверх). После цикла проверяются все маркеры на выполнение условия (left > right) && (bottom < top). *(process_left_to_right)*
3. Обработаем самый правый столбец матрицы A пройдя итерацией (--i) снизу вверх, (т.е. от нижней границы bottom до верхней границы top), записывая значения в массив B по принципу: B[counter / N][counter % N] = A[i][right], при этом идет увеличение на 1 счетчика counter каждую итерацию. При достижении верхней границы матрицы маркер правой границы right уменьшается на 1 (правая граница сдвигается влево). После цикла проверяются все маркеры на выполнение условия (left > right) && (bottom < top). *(process_bottom_to_top)*
4. Потом программа проходит итерацией (--i) по самой верхней строке массива A справа налево (от правой границы к левой), записывая значения в массив B по принципу: B[counter / N] [counter % N] = A[top][i], при этом идет увеличение на 1 счетчика counter каждую итерацию. При достижении левой границы матрицы маркер верхней границы top увеличивается на 1 (верхняя граница сдвигается вниз). После цикла проверяются все маркеры на выполнение условия (left > right) && (bottom < top). *(process_right_to_left)*

В начале цикла: counter = 0 left = 0 top = 0 right = N - 1 bottom = N — 1

Далее границы будут сдвигаться по мере циклического прохождения по матрице по спирали, пока условие (left > right) && (bottom < top) не перестанет выполняться, в этом случае цикл прервется.

Выведем на экран полученный массив B, исходный массив A и ранг матрицы (длина ее стороны).

Контрольный пример

Input.txt:

3

1.1 2.2 3.3

4.4 5.5 6.6

7.7 8.8 9

Output.txt:

Результат:

1.1 4.4 7.7

8.8 9 6.6

3.3 2.2 5.5

Исходник:

1.1 2.2 3.3

4.4 5.5 6.6

7.7 8.8 9

Storona matrix: 3

Формат хранения данных

Имя	Тип	Назначение
i, j	int	Индексные переменные
A[MAX_N][MAX_N] B[MAX_N][MAX_N]	float	Двумерные массивы
temp, num_elements, elements_in_line, counter	int	Вспомогательные переменные
next_char	char	
MAX_N	const unsigned int	Максимальный размер маиссивов
N	int	Реальная длина массива
left, right, top, bottom;	int	Маркеры для считывания матрицы по спирали
MATRIX_LINE	float	Вспомогательные переменные
of, is	fstream	Вспомогательные переменные
filename, inputfile, outputfile	string	Вспомогательные переменные

Ограничение, условленное исполнением на компьютере

Переменные i, j - тип int – диапазон 32768 до 32767
Элементы массива A, B - диапазон от 3,4E–38 до 3,4E+38

Организация интерфейса пользователя

- Пользователь задает различные начальные файлы input.txt, содержащие набор чисел, с учетом того, что самое первое число в файле считается размером квадратной матрицы.
- При считывании программа сначала считывает матрицу построчно, обрабатывая посимвольно каждую строку.

При посимвольной обработке строки используется цикл while (!is.eof() && num_elements < N && num_elements < MAX_N), для которого есть switch:

Switch

1. Если считанный символ — пробел, то пробел пропускается и считывается след. символ, после чего выполняется команда break.
2. Если считанный символ — символ переноса строки (\r), то программа дважды считывает последующие символы и возвращает число элементов в данной строке.
3. В остальных случаях происходит посимвольное считывание элементов с пропусками пробелов и увеличением счетчика num_elements++, который показывает число элементов в строке.

Принцип «урезания» матриц

input.txt	output.txt
Матрица N * N	Матрица N * N
Матрица N * (N-1)	Программа обрезает матрицу по минимальному значению из количества строк и столбцов Матрица (N-1) * (N-1)
Матрица N * 1	Программа обрезает матрицу по минимальному значению из количества строк и столбцов Результат: Матрица 1 * 1, первое значение в файле in.txt
Матрица, строки которой состоят из: N чисел N чисел N чисел N чисел N-1 чисел	Программа обрезает матрицу по минимальному значению из значений во всех строках Результат: Матрица (N-1) * (N-1)
Матрица, строки которой состоят из: N-2 чисел N-3 чисел N чисел N-1 чисел	Программа обрезает матрицу по минимальному значению из значений во всех строках Результат: Матрица (N-3) * (N-3)

Макеты ввода/вывода

1. Чтение файла	input
2. Файл input не найден	Ошибка открытия исходного файла
3. Файл output не найден	Ошибка открытия файла записи
4. Файл input пуст	Ошибка, пустой файл
5. Вывод матрицы	*float A*
6. Результат	*float B*

При выводе матриц в файл исходный и итоговый массивы уже подвергаются «урезанию», а так же вывод матрицы в табличной формк осуществляется с помощью команды setw().

Средства обеспечения ввода/вывода

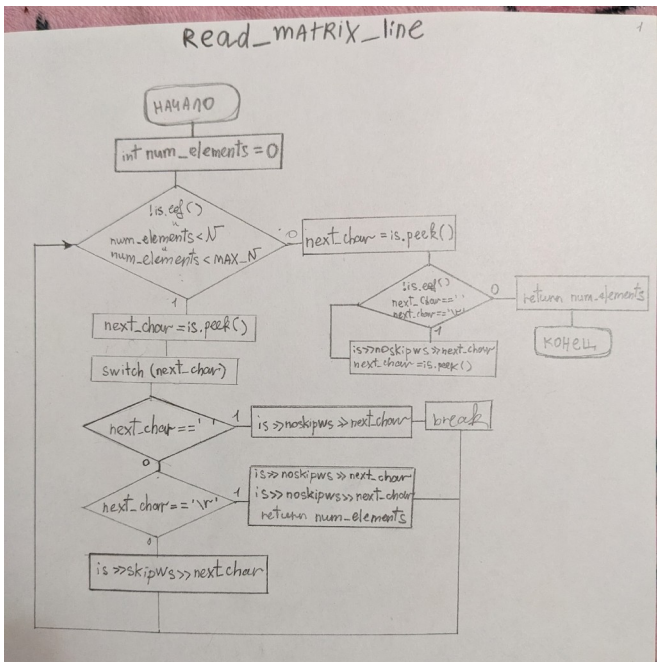
Библиотека	Команды
iostream	cout
fstream	open(), close(), eof(), is_open(), skipws, noskipws, seekg(), tellg()
iomanip	setw(), peek()

Параметры функций:

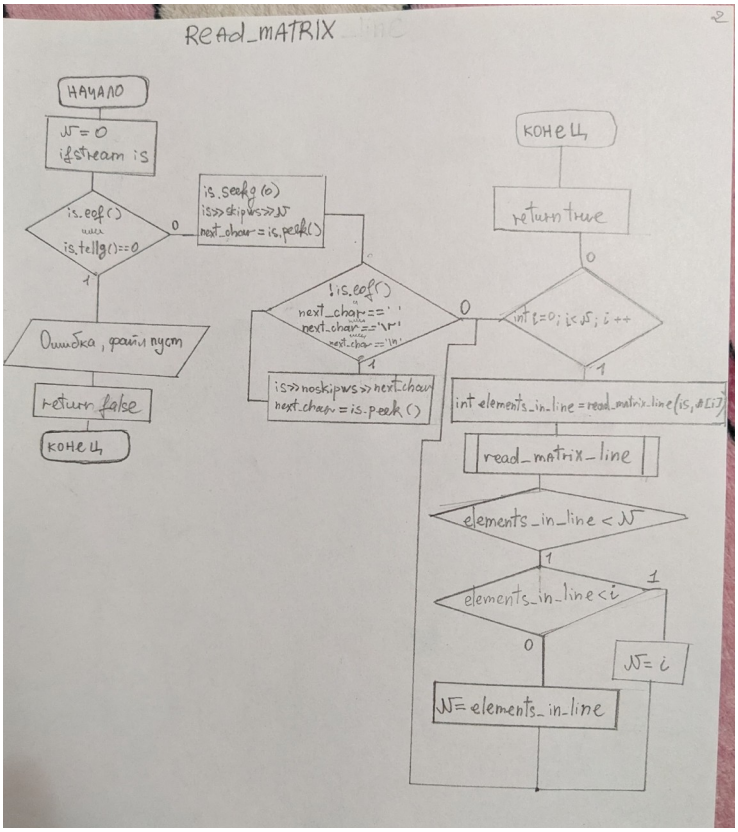
Имя функции	**Назначение**	**Параметры**		
		входные	**выходные**	**модифицируемые**
read_matrix_line	Считать одну строку массива A[i]	is, A[MAX_N]	num_elements	A
read_matrix	Считываем построчно матрицу в массив A	string filename	true/false	A
process_top_to_bottom	Обработка левого столба матрицы A сверху вниз и запись в массив B	top, bottom, left, right, A[MAX_N] [MAX_N], B[MAX_N] [MAX_N], counter	true/false	left, right, i counter, B
process_left_to_right	Обработка нижней строки матрицы A слева направо и запись в массив B	top, bottom, left, right, A[MAX_N] [MAX_N], B[MAX_N] [MAX_N], counter	true/false	bottom, i, counter, B
process_bottom_to_top	Обработка левого столба матрицы A снизу вверх и запись в массив B	top, bottom, left, right, A[MAX_N] [MAX_N], B[MAX_N] [MAX_N], counter	true/false	right, i, counter, B
process_right_to_left	Обработка верхней строки матрицы A справо налево и запись в массив B	top, bottom, left, right, A[MAX_N] [MAX_N], B[MAX_N] [MAX_N], counter	true/false	top, i, counter, B
process	Вызов функций process_top_to_bottom, process_left_to_right, process_bottom_to_top, process_right_to_left	-	-	-
write_matrix_line	Вывод матрицы построчно	of, MATRIX_LINE, N	-	i
write_matrix	Вывод изначальной и обработанной матриц в файл output	filename	-	-
convert	Вызов функций read_matrix, process, write_matrix	inputfile, outputfile	-	-
main	Вызов функции convert для разных файлов	int argc, char *argv[]	0	-

Алгоритм решения

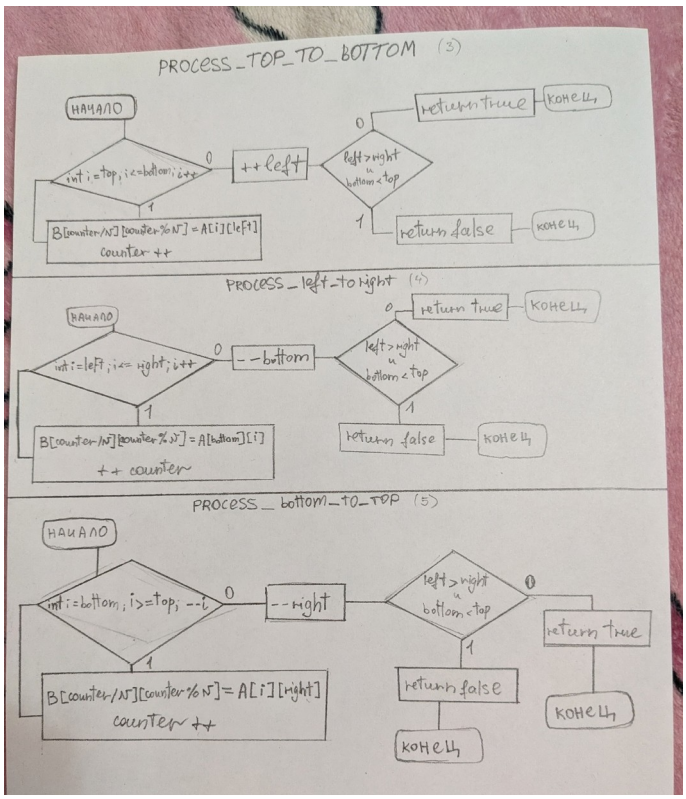
read_matrix_line



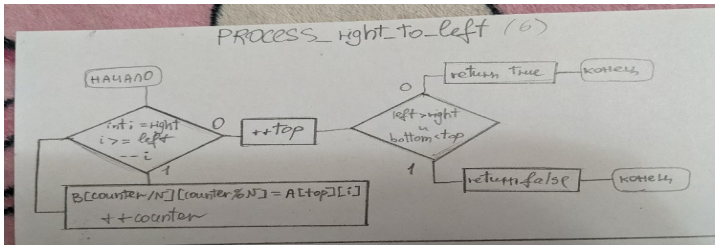
read_matrix



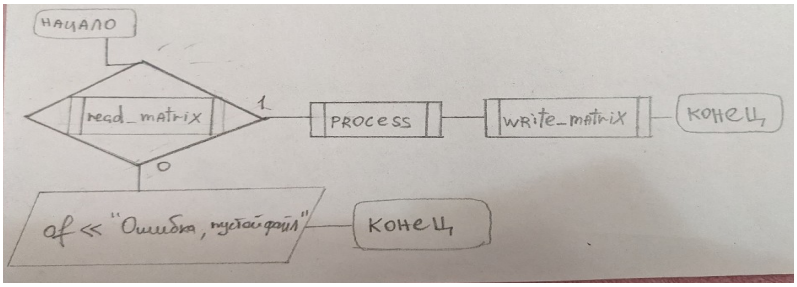
process_top_to_bottom
process_left_to_right
process_bottom_to_top



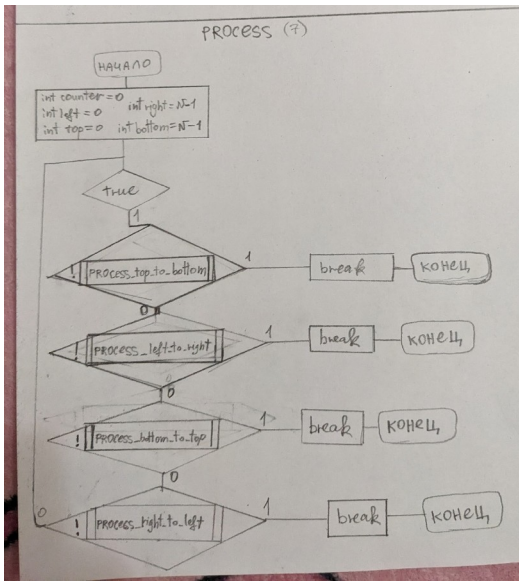
process_right_to_left



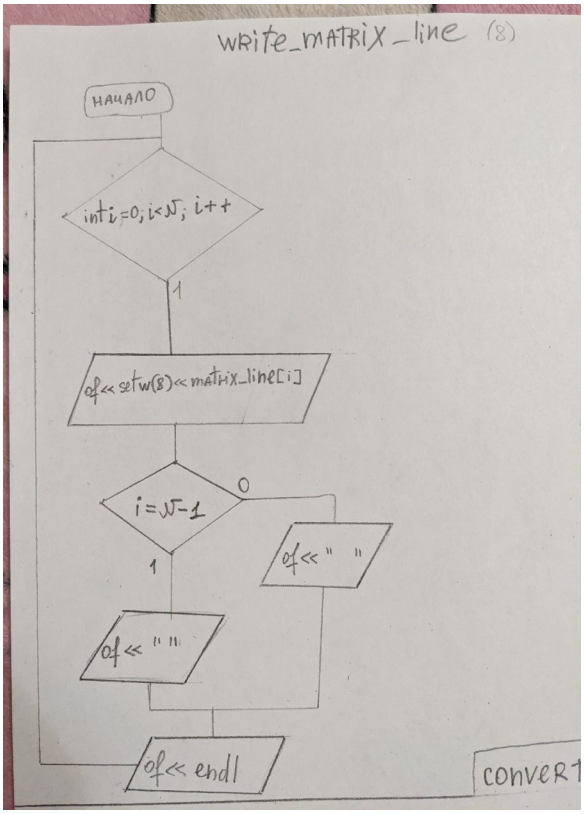
convert



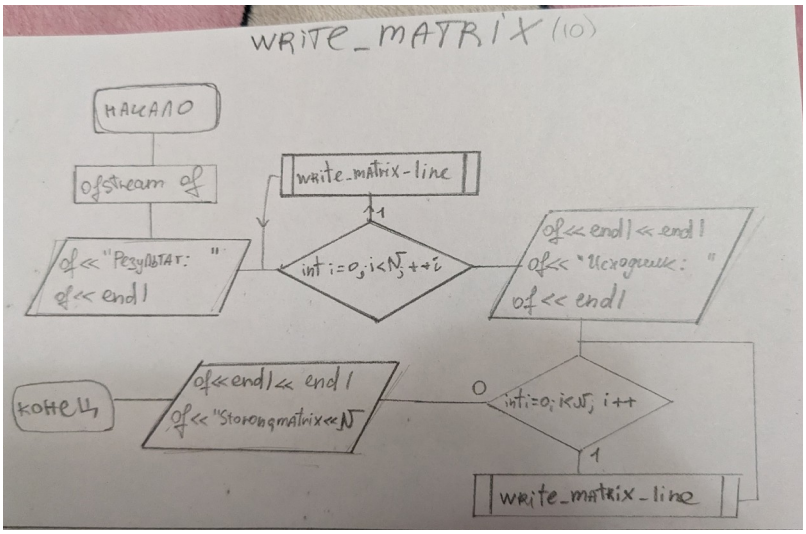
Process



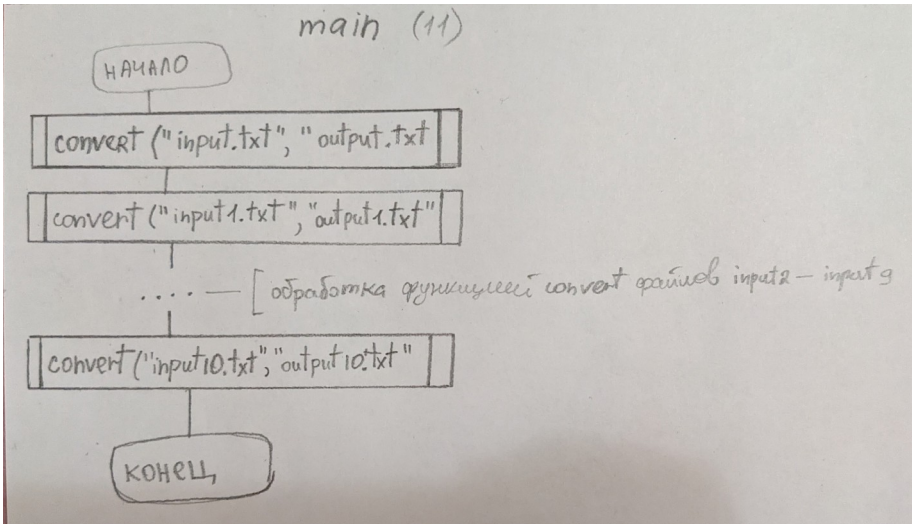
write_matrix_line



write_matrix



main



Программа

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
const unsigned int MAX_N = 32;
float A[MAX_N][MAX_N];
float B[MAX_N][MAX_N];
unsigned int N = 0;
int read_matrix_line(ifstream &is, float A[MAX_N])
{
    int num_elements = 0;
    // обрабатываем строки посимвольно. пропускаем пробелы, когда встречаем перенос строки -
    // считываем его и выходим рано
    while (!is.eof() && num_elements < N && num_elements < MAX_N)
    {
        char next_char;
        next_char = is.peek();
        switch (next_char)
        {
        case ' ': // пробел
            is >> noskipws >> next_char;
            break;
        case '\r': // перенос строки под виндоус "\r\n"
            is >> noskipws >> next_char;
            is >> noskipws >> next_char;
            return num_elements;
        default: // все остальное числа, считываем целиком
            is >> skipws >> A[num_elements++];
        }
    }

    char next_char = is.peek();
    while (!is.eof() && next_char == ' ' || next_char == '\r' || next_char == '\n')
    {
        is >> noskipws >> next_char;
        next_char = is.peek();
    }
    return num_elements;
}

bool read_matrix(std::string filename)
{
    N = 0;
    ifstream is(filename, ios::in | ios::ate);

    if (is.eof() || is.tellg() == 0)
    {
        cout << "ОШИБКА, ПУСТОЙ ФАЙЛ " << filename << endl;
        return false;
    }
    is.seekg(0);
    is >> skipws >> N; // сначала считаем сторону матрицы
    char next_char = is.peek();
    // надо пропустить все лишнее до первого числа
    while (!is.eof() && next_char == ' ' || next_char == '\r' || next_char == '\n')
    {
        is >> noskipws >> next_char;
        next_char = is.peek();
    }
    for (int i = 0; i < N; ++i)
    {   int elements_in_line = read_matrix_line(is, A[i]);
        // если недосчитались элементов в строке сокращаем ранг матрицы
        if (elements_in_line < N)
            N = elements_in_line < i ? i : elements_in_line;
    }
    return true;
}

// возвращает true если надо продолжать обработку
bool process_top_to_bottom(int &top, int &bottom, int &left, int &right, float A[MAX_N][MAX_N], float B[MAX_N][MAX_N], int &counter)
{
    for (int i = top; i <= bottom; ++i)
    {
        B[counter / N][counter % N] = A[i][left];
        ++counter;
    }
    ++left;
    if ((left > right) && (bottom < top))
        return false;
    return true;
}
bool process_left_to_right(int &top, int &bottom, int &left, int &right, float A[MAX_N][MAX_N], float B[MAX_N][MAX_N], int &counter)
{
    for (int i = left; i <= right; ++i)
    {
        B[counter / N][counter % N] = A[bottom][i];
        ++counter;
    }
    --bottom;
    if ((left > right) && (bottom < top))
        return false;
    return true;
}
bool process_bottom_to_top(int &top, int &bottom, int &left, int &right, float A[MAX_N][MAX_N], float B[MAX_N][MAX_N], int &counter)
{
    for (int i = bottom; i >= top; --i)
    {
        B[counter / N][counter % N] = A[i][right];
        ++counter;
```

```
    }
    --right;
    if ((left > right) && (bottom < top))
        return false;
    return true;
}
bool process_right_to_left(int &top, int &bottom, int &left, int &right, float A[MAX_N][MAX_N], float B[MAX_N][MAX_N], int &counter)
{
    for (int i = right; i >= left; --i)
    {
        B[counter / N][counter % N] = A[top][i];
        ++counter;
    }
    ++top;
    if ((left > right) && (bottom < top))
        return false;
    return true;
}

void process()
{    int counter = 0;
    int left, right, top, bottom;
    left = 0;
    top = 0;
    right = N - 1;
    bottom = N - 1;

    while (true)
    {
        if (!process_top_to_bottom(top, bottom, left, right, A, B, counter))
            break;
        if (!process_left_to_right(top, bottom, left, right, A, B, counter))
            break;
        if (!process_bottom_to_top(top, bottom, left, right, A, B, counter))
            break;
        if (!process_right_to_left(top, bottom, left, right, A, B, counter))
            break;
    }
}

void write_matrix_line(ofstream &of, float *MATRIX_LINE, int N)
{
    for (int i = 0; i < N; ++i)
    {
        of << setw(8) << MATRIX_LINE[i] << (i == N - 1 ? "" : " ");
    }
    of << endl;
}

void write_matrix(std::string filename)
{
    ofstream of(filename, ios::out | ios::trunc);
    of << "Результат: ";
    of << endl;
    for (int i = 0; i < N; ++i)
        write_matrix_line(of, B[i] /*начало итой строчки*/, N);

    of << endl;
    of << endl;
    of << "Исходник: ";
    of << endl;
    for (int i = 0; i < N; ++i)
    {
        write_matrix_line(of, A[i] /*начало итой строчки*/, N);
    }
    of << endl;
    of << endl;
    of << "Storona matrix: " << N;
}

void convert(string inputfile, string outputfile)
{
    if (read_matrix(inputfile))
    {
        // cut_matrix(inputfile, N);
        process();
        write_matrix(outputfile);
    }
    else
    {
        ofstream of(outputfile, ios::out | ios::trunc);
        of << "ОШИБКА, ПУСТОЙ ФАЙЛ " << inputfile << endl;
    }
}

int main(int argc, char *argv[])
{
    convert("input.txt", "output.txt");
    convert("input1.txt", "output1.txt");
    convert("input2.txt", "output2.txt");
    convert("input3.txt", "output3.txt");
    convert("input4.txt", "output4.txt");
    convert("input5.txt", "output5.txt");
    convert("input6.txt", "output6.txt");
    convert("input7.txt", "output7.txt");
    convert("input8.txt", "output8.txt");
    convert("input9.txt", "output9.txt");
    convert("input10.txt", "output10.txt");
    return 0;
}
```

Результаты работы программы

INPUT.TXT	OUTPUT.TXT
Input.txt 3 1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9	**Output.txt** Результат: 1.1 4.4 7.7 8.8 9 6.6 3.3 2.2 5.5 Исходник: 1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9 Storona matrix: 3
Input5.txt 3 1 2 3 4 5 6	**Output.txt** Результат: 1 4 5 2 Исходник: 1 2 4 5 Storona matrix: 2
Input2.txt	**Output2.txt** ОШИБКА, ПУСТОЙ ФАЙЛ input2.txt

Вывод

В процессе работы над программой был изучен способ чтения двумерных массивов, проведены манипуляции над ними и работа с файлами. Написанная программа работает исправно.