ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ

За курсом «Інформатика і Програмування»

Студента групи МС-19-1

Політіки Владислава Дмитровича

Лабораторна робота №8

**Тема**: "Двовимірні масиви. Робота з файлами"

Варіант 11

**Завдання:**

Складіть програму, яка здійснює обробку вхідних даних за допомогою двовимірних масивів.

При виконанні лабораторної роботи слід врахувати наступне:

* Дати користувачеві можливість вибору: вводити вхідні дані з консолі, чи вводити їх з файлу; у випадку вводу з консолі продемонструвати можливість перенапрямлення потоку вводу/виводу на файл.
* При програмуванні однотипних підзадач (сортування, ввід/вивід масивів, знаходження суми стовпця матриці тощо) слід оформлювати їх у вигляді функцій.

**Варіант 11.** Задайте матриці *А* і *В* розмірністю (*n* x *m*), де *n* і *m* ≥ 6 (*n**m*). Виконайте такі завдання:

1. знайдіть кількість від’ємних елементів в обох матрицях;

2. визначте матрицю *С* як результат транспонування матриці *А*,якщо кількість від’ємних елементів матриці *А* більша кількості від’ємних елементів матриці *В* (в іншому випадку – як результат транспонування матриці *В*);

3. упорядкуйте за спаданням значення елементів заданого стовпця.

**Завдання №1-3**

1. Опис розв’язку:

Для розв’язку задач спершу потрібно задати матриці **А** та **В**. Для цього можна використати одну і ту саму функцію два рази (в програмі - це функція add\_Matrix()), до якої передаються необхідні змінні (наприклад адреса першого елемента масиву, щоб функція працювала з певним масивом). Далі функція розглядає два варіанти вводу матриці - з клавіатури та з файлу. Правильність вводу варіанта перевіряєтся у функції PUT()(алгоритм: поки введений символ - не 1 або 2, то ввід повторюється знову і знову, далі функція повертає обраний варіант).

Якщо користувач вибирає варіант вводу матриці з клавіатури, то він задає розмір та елементи матриці вручну (якщо елементи задаються до двовимірного масиву, то необхідно розглянути цикл в циклі, де зовнішній перебирає всі індекси рядків(від 0 до максимального), а другий - всі індекси стовпців. У внутрішньому циклі ж знаходиться оператор вводу елемента масиву з відповідними індексами). Також далі пропонується користувачу дописати введену матрицю у деякий файл. Це завдання виконує функція add\_to\_file. Вона пропонує варіанти: місцезнаходження файлу записати вручну або ж вибрати запропоноване місцезнаходження файлу. У тому чи іншому варіанті відкривається поток для дозапису певного файла. Дозапис матриці здійснюється за допомогою таких самих циклів що і у випадку вводу матриці, проте у внутрішньому циклі маємо наступне: put << M[i][j] << '\t';put << endl;.Проте, якщо дозапис здійснити неможливо через деякі причини, то за згодою користувача, програма надає можливість дописати матрицю в інший файл (це єдиний випадок, коли функцію треба використати ще раз, тому функція повертає значення true, що дає можливість циклу while продовжити оголошувати цю функцію до поки її значення - не false). Якщо ж користувач передумав дописувати матрицю у деякий файл (користувач при цьому обирає відповідний варіант) чи матриця була успішно дозаписана у файл, то функція повертає false, що буде означати вихід з циклу.

Якщо користувач обирає варіант вводу матриці з файлу, то програма пропонує вибрати варіант для цього (ввести місцезнаходження файлу вручну чи вказати запропоноване місцезнаходження деякого файлу). Для зчитування елементів матриці з файлу відкривається потік. Якщо це зробити вдалося, то за допомогою внутрішнього і зовнішнього циклів, що перебирають всі елементи матриці, записуємо в них значення елементів з файлу (в циклі маємо: save >> file; Mat[i][j] = atoi(file);, де file - масив типу char, Mat - двовимірна матриця, що розглядають). Також здійснюється перевірка кожного розглянутого у файлі елемента. У випадку, коли файл відкрити не вдалося або коли елементи з файлу не відповідають певним вимогам (нецілі числа, початок - це літера тощо), то функція add\_Matrix() оголошується знову.

**Завдання 1.** Далі, відповідно до завдання, треба вивести кількість від’ємних елементів у матрицях (функція calcul\_of\_neg). Вона, перебираючи всі елементи матриці, за певною умовою (елемент матриці менший за нуль), рахує від’ємні елементи, виводить їх та повертає як значення функції.

**Завдання 2.** Наступне завдання - транспонувати матрицю, що має найбільшу кількість від’ємних елементів. Для визначення цієї матриці користуємось умовою (умова neg\_A < neg\_B, де neg\_A - кількість від’ємних елементів матриці А, що була отримана зі значення функції calcul\_of\_neg, neg\_D - кількість від’ємних елементів матриці B, що була отримана зі значення функції, calcul\_of\_neg). За значенням умови (true/false) передаємо функції Transpon\_Mat\_into\_new() певні змінні, адресу першого елемента певного масиву тощо. Ця функція присвоює елементам масиву С, що мають “перевернуті” інденси порівняно з масивом А чи В (залежить від значення умови neg\_A < neg\_B), елементи масиву А чи В. Тут використовуємо ті ж цикли, що і зустрічалися раніше. Далі виводимо транспоновану матрицю, користуючись знову тими ж циклами й оператором cout. До того ж ще програма пропонує дозаписати її до файлу як і матрицю, що вводиться.

**Завдання 3.** Та останнє завдання - упорядкувати за спаданням значення елементів заданого стовпця. Це робить функція sort\_column(). Нехай користувач ввів деякий стовпець J у діапазоні від нуля до індексу останнього елемента рядка матриці включно. Тоді для упорядкування елементів у заданому стовпці будемо розглядати всі індекси рядків L (від 0 до максимального включно), у кожному з яких розглядаються всі індекси R, що дорівнюють тількі можливим значенням, які більші за індекс L. Далі перевіряємо елементи на упорядоченість, що знаходяться у вибраному стовпці J та рядках R та L (у програмі перевіряє на упорядоченість функція comp). Якщо вони не упорядочені (так як потребує завдання), то міняємо місцями не лише ці елементи, а й ту пару, що має однаковий індекс стовпців та різні індекси рядків R та L. Тут розглядаємо всі індекси стовпців I (від 0 до максимального включно). Це можна записати у вигляді коду так:

for (int L = 0; L < n; L++)

for (int R = L + 1; R < n; R++)

if (comp(Mat[L][J - 1], Mat[R][J - 1]))

for (int I = 0; I < m; I++)

SWAP(Mat[L][I], Mat[R][I]);

Тут n - кількість рядків, m - стовпців, SWAP() - функція, що змінює місцями значення двох змінних в оперативній пам’яті. Тож матриця упорядковується, а далі виводиться на екран. До того ж програма ще й пропонує дозаписати отриману матрицю в файл, що було вже розглятуто.

1. Вихідний текст програми розв’язку задачі:

#include <iostream>

#include <Windows.h>

#include <fstream>

using namespace std;

const int c = 100;

int A[c][c], B[c][c], C[c][c];

int str(char lett, char called[])

{

int a;

cout << " Количество " << called << ' ' << lett << " (где " << lett << " >= 6): ";

cin >> a;

while (a < 6)

{

cout << " Внимание! Вы должны ввести число большое либо равное числу 6!" << endl;

cout << " Количество " << called << ' ' << lett << " (где " << lett << " >= 6): ";

cin.clear();

while (cin.get() != '\n');

cin >> a;

}

return a;

}

void PUT(char var[])

{

cin.getline(var, 100);

while ((var[0] != '1' && var[0] != '2') || (strlen(var) != 1))

{

cout << " Внимание! Введённое число должно быть единицей или двойкой!" << endl;

cin.getline(var, 100);

}

}

bool add\_to\_file(int M[][c], char var[], char file[], int n, int m)

{

ofstream put;

cout << " Выберете вариант для дозаписи объявленной матрицы в файл:\n";

cout << " 1) ввести расположение файла вручную\n";

cout << " 2) указать файл D:\\My\_environment.txt\n";

cout << " Введите выбранный Вами вариант: "; PUT(var);

if (var[0] == '1')

{

cout << " Введите расположение файла вручную: ";

cin.getline(file, 200); put.open(file, ios::app);

}

else

put.open("D:\\My\_environment.txt", ios::app);

if (put.is\_open())

{

put << endl << n << ' ' << m << endl;

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

{

for (int j = 0; j < m; j++)

put << M[i][j] << '\t';

put << endl;

}

put.close();

cout << " Дозапись в файл успешно завершена!\n";

return false;

}

else

{

cout << " Файл не доступен! Но Вы можете попробовать выбрать другой\n";

cout << " Хотите ли Вы дозаписать введённую матрицу в другой файл? (1 - да, 2 - нет)\n"; PUT(var);

if (var[0] == '1') return true;

else return false;

}

}

void add\_Matrix(int Mat[][c], char nam\_Mat, int& n, int& m)

{

char var[100], file[200], cal[] = "строк"; bool mist = false;

cout << " Выберете вариант ввода матрицы " << nam\_Mat << ":\n 1) вручную (с клавиатуры);\n 2) с

файла.\n" << " Введите выбранный Вами вариант: "; PUT(var);

if (var[0] == '1')

{

cout << " Ввод матрицы с помощью клавиатуры:\n";

cout << " Введите количество строк и столбцов матрицы " << nam\_Mat << ":\n";

n = str('n', cal);

char cal[] = "столбцов";

m = str('m', cal);

cout << " Задайте матрицу " << nam\_Mat << ", где i - индекс строки, j - столбца, а " << nam\_Mat

<< "[i][j] - элемент в матрице:\n";

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

for (int j = 0; j < m; j++)

{

cout << ' ' << i \* m + j + 1 << ") " << nam\_Mat << '[' << i << "][" << j << "] = "; cin >> Mat[i][j];

}

cin.ignore();

cout << " Хотите ли Вы дозаписать введённую матрицу в файл? (1 - да, 2 - нет)\n";

PUT(var);

if (var[0] == '1')

while(add\_to\_file(Mat, var, file, n, m));

}

else

{

ifstream save;

cout << " Ввод матрицы " << nam\_Mat << " с помощью файла:\n";

cout << " Выберете вариант:\n";

cout << " 1) ввести расположение файла вручную\n";

cout << " 2) указать файл D:\\My\_environment.txt\n";

cout << " Введите выбранный Вами вариант: "; PUT(var);

if (var[0] == '1')

{

cout << " Введите расположение файла вручную: ";

cin.getline(file, 200); save.open(file);

}

else

save.open("D:\\My\_environment.txt");

if (save.is\_open())

{

save >> file; n = atoi(file);

if (atoi(file) == 0 && file[0] != '0') mist = true;

save >> file; m = atoi(file);

if (atoi(file) == 0 && file[0] != '0') mist = true;

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

for (int j = 0; j < m; j++)

{

save >> file; Mat[i][j] = atoi(file);

if (atoi(file) == 0 && file[0] != '0') mist = true;

}

save.close();

cout << " Извлечённые данные из файла:" << endl;

cout << " n = " << n << endl;

cout << " m = " << m << endl;

cout << " Матрица " << nam\_Mat << " размером " << n << '\*' << m << ':' << endl;

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

{

for (int j = 0; j < m; j++)

{

cout << Mat[i][j] << '\t';

}

cout << endl;

}

if (n < 6 || m < 6 || mist) {

cout << " Внимание! Извлечённые параметры не соотвецтвуют условиям ввода матрицы!

Повторите попытку!\n";

add\_Matrix(Mat, nam\_Mat, n, m);

}

}

else {

cout << " Файл не доступен! Но Вы можете попробовать задать матрицу " << nam\_Mat << " ещё

раз!";

add\_Matrix(Mat, nam\_Mat, n, m);

}

}

}

int calcul\_of\_neg(int Mat[][c], char nam\_Mat, int n, int m)

{

int value = 0;

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

for (int j = 0; j < m; j++)

if (Mat[i][j] < 0) value++;

cout << " Количество отрицательных элементов в матрице " << nam\_Mat << ':' << value << '\n';

return value;

}

void Transpon\_Mat\_into\_new(int Mat[][c], int n, int m)

{

char var[100], file[200];

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

for (int j = 0; j < m; j++)

C[j][i] = Mat[i][j];

// вывод матрицы

cout << " Вывод транспонированной матрицы с большим числом отрицательных элементов:\n";

for (int i = 0; i < m; i++)

{

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

cout << C[i][j] << '\t';

cout << '\n';

}

cout << " Хотите ли Вы дозаписать транспонированную матрицу в файл? (1 - да, 2 - нет)\n";

PUT(var);

if (var[0] == '1') while (add\_to\_file(C, var, file, m, n));

}

bool comp(int x, int y)

{

return x < y;

}

void SWAP(int& x, int& y)

{

int t;

t = x; x = y; y = t;

}

void sort\_column(int Mat[][c], char nam\_Mat, int n, int m)

{

char var[100], file[200];

int J;

cout << " Введите столбик матрицы " << nam\_Mat << ", который будет сортирован по убыванию значений

элементов в нём: ";

cin >> J;

while (1 > J || J > m)

{

cout << " Номер столбика должен находиться в диапазоне от 1 до " << m << " включительно!\n"

<< " Повторите попытку. Введите столбик матрицы " << nam\_Mat << ", который будет сортирован

по убыванию значений элементов в нём: ";

cin.clear();

while (cin.get() != '\n');

cin >> J;

}

for (int L = 0; L < n; L++)

for (int R = L + 1; R < n; R++)

if (comp(Mat[L][J - 1], Mat[R][J - 1]))

for (int I = 0; I < m; I++)

SWAP(Mat[L][I], Mat[R][I]);

cout << " Вид сортированной матрицы " << nam\_Mat << " по убыванию значений элементов в столбике "

<< J << '\n';

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

{

for (int j = 0; j < m; j++)

cout << Mat[i][j] << '\t';

cout << '\n';

}

cout << " Хотите ли Вы дозаписать сортированную матрицу " << nam\_Mat << " в файл? (1 - да, 2 - нет)\n";

cin.ignore();

PUT(var);

if (var[0] == '1') while (add\_to\_file(Mat, var, file, n, m));

}

int main()

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int A\_n, A\_m, B\_n, B\_m, neg\_A, neg\_B;

cout << " Задание 1, вариант 11\n";

add\_Matrix(A, 'A', A\_n, A\_m); //ввод матрицы

add\_Matrix(B, 'B', B\_n, B\_m); //ввод матрицы

// определение количества отрицательных значений

cout << " Результаты вычисений:\n";

neg\_A = calcul\_of\_neg(A, 'A', A\_n, A\_m); // определение количества отрицательных элементов в матрице A

neg\_B = calcul\_of\_neg(B, 'B', B\_n, B\_m); // определение количества отрицательных элементов в матрице B

(neg\_A < neg\_B) ? Transpon\_Mat\_into\_new(B, B\_n, B\_m) : Transpon\_Mat\_into\_new(A, A\_n, A\_m);

sort\_column(A, 'A', A\_n, A\_m); // сортировка по убыванию значений элементов заданого столбика

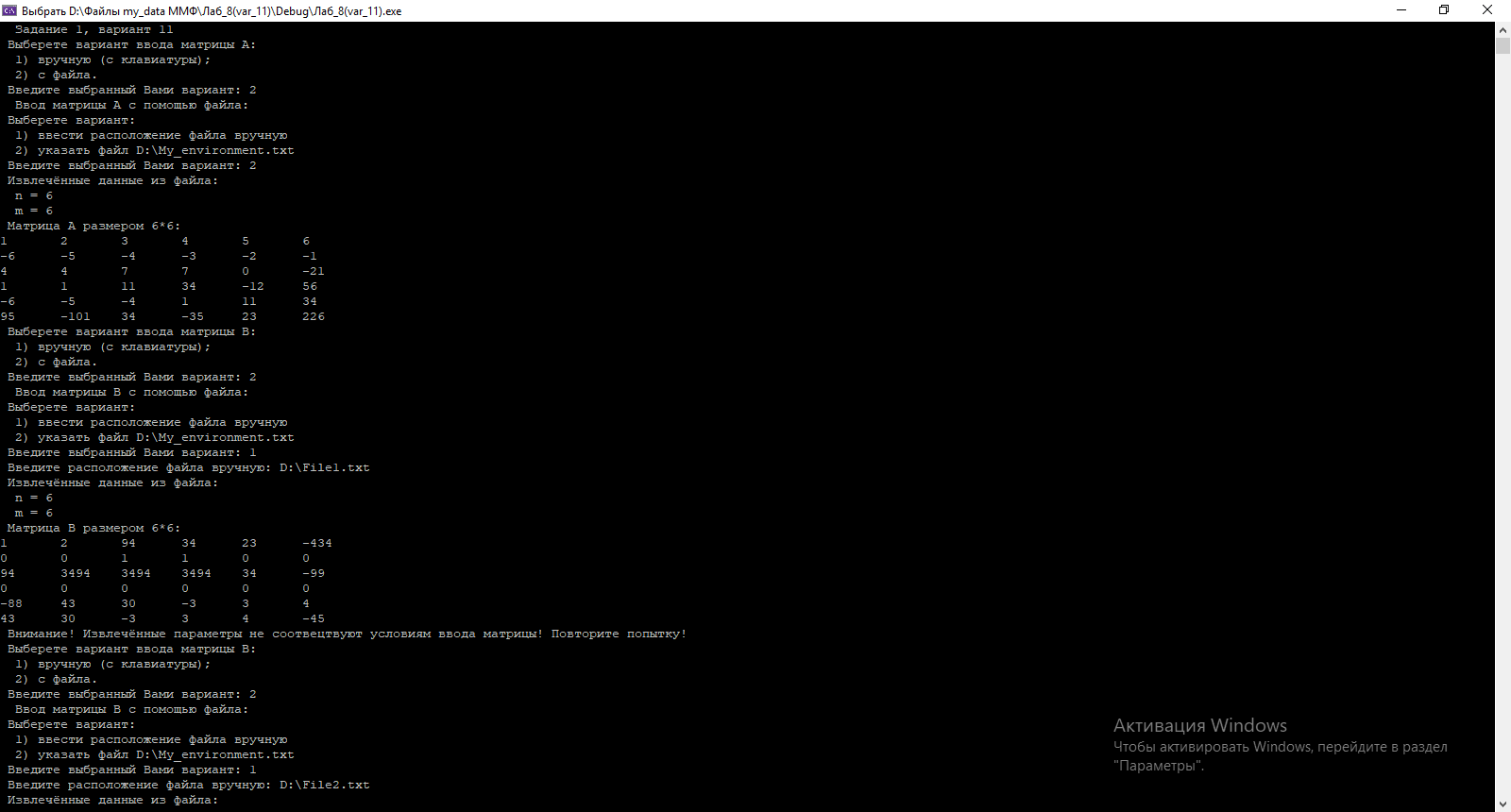
sort\_column(B, 'B', B\_n, B\_m); // сортировка по убыванию значений элементов заданого столбика

return 0;

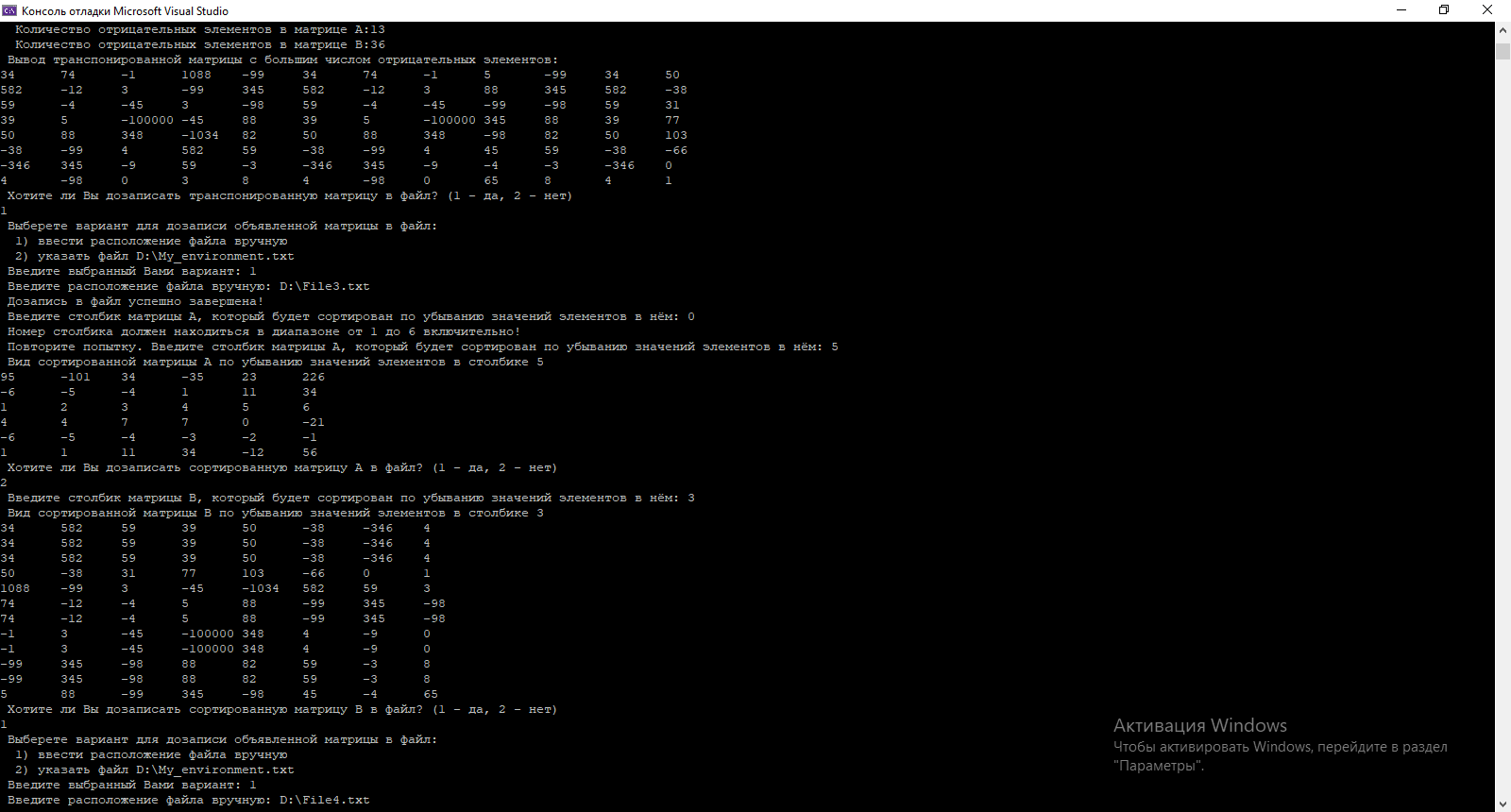
}

1. Опис інтерфейсу:

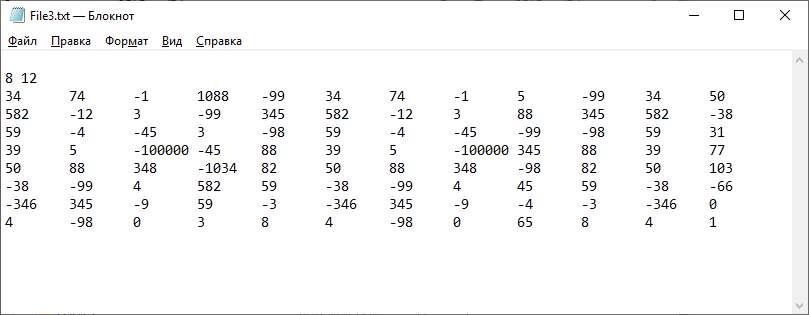
Вигляд консолі після вводу матриць з файлів:

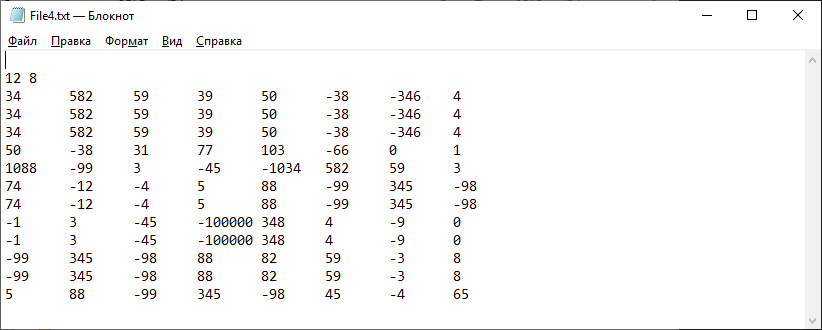


Результати обчислень подані нижче:



До того ж я дописав транспоновану матрицю у файл File3, а сортовану матрицю В у файл File4:





1. Опис тестових прикладів:

Один з тестових прикладів був показан в описі інтерфейсу. Проте, слід зауважити, що програма перевіряє ввід скрізь, крім випадку введення матриць вручну та при введені стовпця за яким програма сортує матриці. Як щодо довільних введених значень, то програма працює без помилок.

5. Аналіз помилок:

Помилок немає.