Содержание

Введение

По заданию 3.17 необходимо разработать программу, формирующую от датчика случайных чисел две матрицы размером $m \times n$ и $r \times s$ соответственно. В ходе работы программы должны быть найдены:

- произведение 2-х матриц;
- определители обеих матриц;
- результат транспонирования матрицы с большим по значению определителем.

При этом программа должна содержать главную функцию и две неглавных функции. Формирование матриц, печать матриц с сопроводительным текстом, а также печать значения определителей должны быть выполнены в главной функции. В неглавных функциях должны быть выполнены операции перемножения матриц и их транспонирование.

Теоретическая часть

1.1 Матрицы

Матрица A размера $m \times n$ — это прямоугольная таблица чисел, расположенных в m строках и n столбцах где a_{ij} ($i = 1, \ldots, m; j = 1, \ldots, n$) — это элементы матрицы A. Первый индекс i — это номер строки, второй индекс j — это номер столбца, на пересечении которых расположен элемент a_{ij}.

Произведением матрицы $A=(a_{ij})_{m\times k}$ на матрицу $B=(b_{ij})_{k\times n}$ называется матрица $C=(c_{ij})_{m\times n}=A\cdot B$ размера $m\times n$, элементы которой c_{ij} определяются равенством $c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+ \ldots a_{ik}b_{kj}$. Таким образом, элемент матрицы $C=A\cdot B$, расположенный в i-й строке и j-м столбце, равен сумме произведений элементов i-й строки матрицы A на соответствующие элементы j-го столбца матрицы B.

Определитель (детерминант) матрицы — некоторое число, с которым можно сопоставить любую квадратную матрицу $A=(a_{ij})_{n \times n}$.

|A|, Δ , det A - символы, которыми обозначают определитель матрицы. Способ вычисления определителя выбирают в зависимости от порядка матрицы. Определитель существует только для квадратной матрицы (размера n × n)

Для матрицы первого порядка значение определителя равно единственному элементу этой матрицы. Для матрицы второго порядка (2×2) значение определителя вычисляется как $a_{11} \times a_{22} - a_{12} \times a_{21}$. Для матриц более высоких порядков (выше второго) m×m определитель можно вычислить, применив рекурсивную формулу:

$$\sum_{j=1}^{m} (-1)^{1+j} \cdot a_{1j} \cdot M_{1j}$$

Формула 1 - Определитель матрицы m порядка

где Mij — дополнительный минор квадратной матрицы — определитель матрицы, полученной из исходной вычеркиванием i-ой строки и j-ого столбца (в данной формуле i = 1).

Транспонирование матрицы - это операция над матрицей, при которой ее строки и столбцы меняются местами.

$$a^{T}_{ij} = a_{ji}$$

Формула 2 - Расположение транспонированного элемента матрицы.

1.2 Рекурсия

В программировании рекурсия тесно связана с функциями, точнее именно благодаря функциям в программировании существует такое понятие как рекурсия или рекурсивная функция.

Простыми словами, рекурсия – определение части функции (метода) через саму себя, то есть это функция, которая вызывает саму себя, непосредственно (в своём теле) или косвенно (через другую функцию). Типичными рекурсивными задачами являются задачи: нахождения n!, числа Фибоначчи. Такие задачи мы уже решали, но с использованием циклов, то есть итеративно.

Вообще говоря, всё то, что решается итеративно можно решить рекурсивно, то есть с использованием рекурсивной функции. Всё решение сводится к решению основного или, как ещё его называют, базового случая. Существует такое понятие как шаг рекурсии или рекурсивный вызов. В случае, когда рекурсивная функция вызывается для решения сложной задачи (не базового случая) выполняется некоторое количество рекурсивных вызовов или шагов, с целью сведения задачи к более простой. И так до тех пор, пока не получим базовое решение.

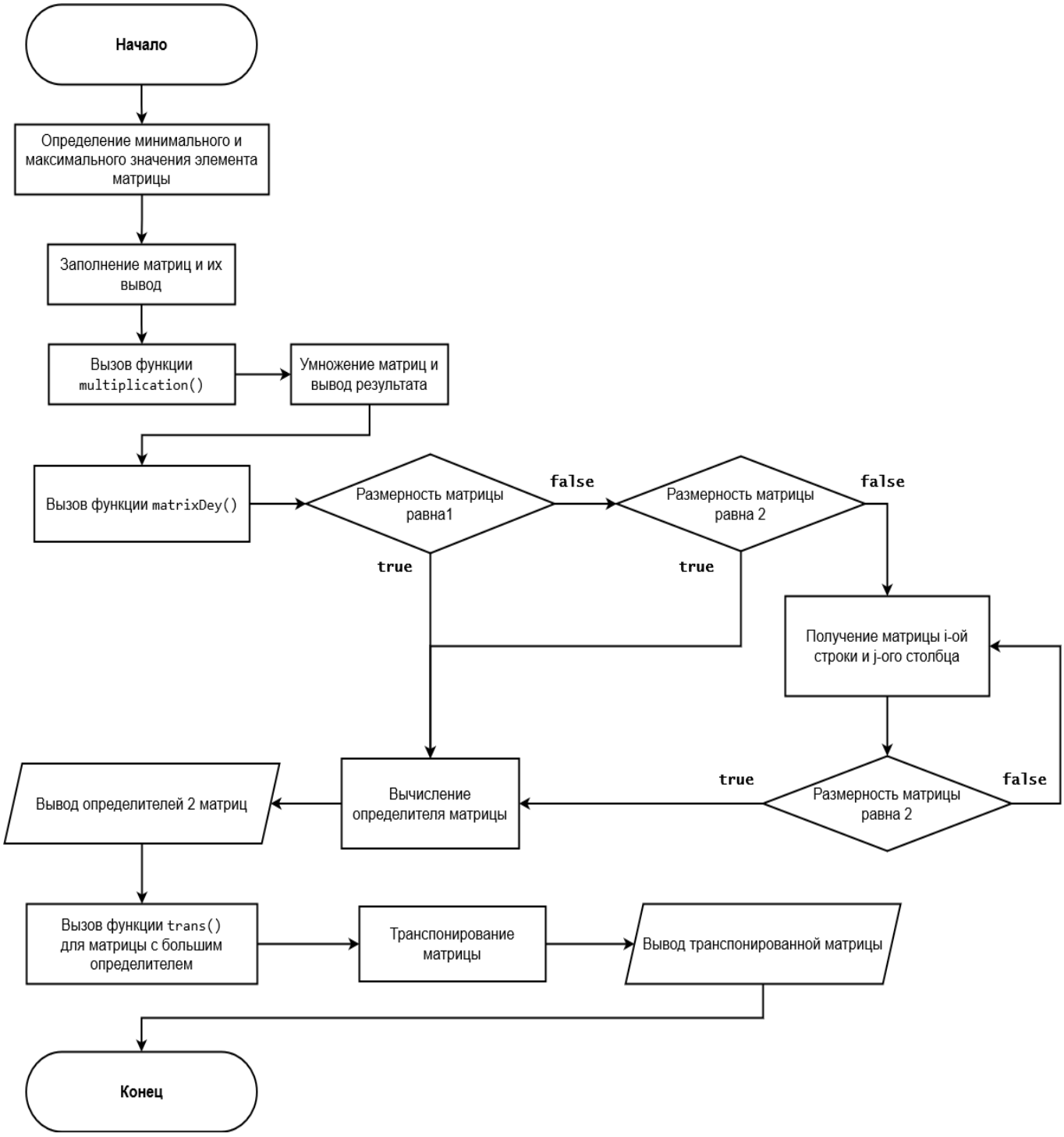
Пример - программа, вычисляющая n!:

```
int main(int argc, char* argv[])
{
	int n; // локальная переменная для передачи введенного числа
	cout << "Enter n!: ";
	cin >> n;
	cout << n << "!" << "=" << factorial(n) << endl; // вызов рекурсивной функции
	return 0;
}

unsigned long int factorial(unsigned long int f) // рекурсивная функция для нахождения n!
{
	if (f == 1 || f == 0) // базовое или частное решение
		return 1;
	result = f * factorial(f - 1); // функция вызывает саму себя, причём её аргумент уже на 1 меньше
	return result;
}
```

1.3 Алгоритм работы

Перед началом разработки программы следует продумать её общий алгоритм.



Блок-схема 1 - Алгоритм работы программы.

Практическая часть

2.1 Создание программы

Для ввода необходимых значений и вывода результатов работы программы и сопроводительного текста будет достаточно командной консоли.

Для создания подобной программы в качестве среды разработки подходит Visual Studio 2019. При создании проекта выбираем шаблон консольного приложения C++.

2.2 Создание и вывод матриц

Для начала нужно объявить переменные матриц A и B - начальные матрицы, которые будут заполнены случайными числами. Матрицы задаются как двумерный массив.

Случайные числа в языке программирования C++ могут быть сгенерированы функцией rand() из стандартной библиотеки C++. Функция rand() генерирует числа в диапазоне от 0 до RAND_MAX. RAND_MAX — это константа, определённая в библиотеке cstdlib. Для MVS RAND_MAX = 32767, но оно может быть и больше, в зависимости от компилятора.

Для того чтобы масштабировать интервал генерации чисел нужно воспользоваться, операцией нахождения остатка от деления %. Для определения интервала генерации чисел от заданного числа до заданного числа используется следующая формула:

min + rand() % (max - min + 1)

Фрагмент кода 1 - Заполнение матриц.

где min и max - целочисленные переменные, заданные пользователем.

Функция srand() из библиотеки stdlib.h предназначена для установки начальной точки, из которой происходит генерирование случайных чисел. Для того, чтобы производить рандомизацию автоматически, то есть, не меняя каждый раз аргумент в функции srand() нужно воспользоваться функцией time() из библиотеки time.h с аргументом 0.

Так как:

- По условию умножения матриц количество столбцов первой матрицы - множителя должно быть равным количеству строк второй матрицы
- Из свойств определителя следует что определитель существует только для квадратной матрицы (размера n × n)

то размерность начальных матриц будет равной и будет задаваться пользователем в целочисленной переменной N.

Заполнение первых двух матриц происходит следующим образом:

1. Матрице присваивается новое значение в виде одномерного массива с числом элементов N
2. С помощью цикла for перебираются все элементы нового массива
3. Каждому элементу присваивается новое значение в виде одномерного массива с числом элементов N после чего все элементы нового массива перебираются с помощью цикла for
4. Каждому элементу присваивается новое численное значение по формуле min + rand() % (max - min + 1)

После заполнения матриц матрицы выводятся в консоль. Для этого в цикле for перебираются все строчки i, и для каждой найденной строки с помощью цикла for перебираются все возможные столбцы j

2.3 Умножение матриц

В функции multiplication() происходит умножение начальных матриц и вывод матрицы произведения. Эта функция принимает два двумерных массива A и B - начальные матрицы и их размерность N. В начале функции объявляется переменная матрицы произведения C. После этого выполняется алгоритм заполнения матрицы C:

1. Матрице присваивается новое значение в виде одномерного массива с числом элементов N
2. С помощью цикла for перебираются все элементы нового массива
3. Каждому элементу присваивается новое значение в виде одномерного массива с числом элементов N после чего все элементы нового массива перебираются с помощью цикла for
4. С помощью цикла for прибавляем к каждому элементу произведение элементов начальных матриц по формуле $c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+ \dots a_{ik}b_{kj}$.

После этого матрица C выводится в консоль.

2.4 Определители матриц

В функции matrixDet() осуществляется поиск определителя матрицы.

Функция принимает двумерный массив matrix - матрицу, определитель которой нужно найти и размерность этой матрицы. В начале функции объявляются целочисленные переменные определителя и степени элемента degree (-1^(1+j) из формулы определителя).

После этого обозначаются два условия выхода из рекурсии (для матриц первого и второго порядка). В остальных случаях объявляется двумерный массив для матрицы newMatrix без строки и столбца (N – 1).

Далее выполняется разложение по первой строке матрицы matrix. Чтобы удалить из матрицы i-ю строку и j-ый столбец используется функция getMatrixWithoutRowAndCol() в которую передаются матрицы matrix и newMatrix а также столбец и строка, которые надо убрать из матрицы. Нужная нам матрица будет записана в newMatrix.

После используя рекурсивный вызов находится дополнительный минор элемента matrix[0][j] - определитель матрицы newMatrix. В случае, если размерность матрицы newMatrix больше двух, то её определитель находится по такой-же формуле. Функция matrixDet() будет вызываться до тех пор, пока размерность матрицы newMatrix не уменьшится до двух, тогда будет возможно найти её определитель, а значит можно будет найти и определители newMatrix для каждого шага рекурсии.

Далее к определителю матрицы matrix прибавляется произведение нужного элемента и его дополнительного минора (при этом происходит домножение на степень degree, после чего меняется степень элемента). При выходе из цикла на каждом шаге рекурсии матрица newMatrix удаляется.

После всех вычислений функция возращает значение определителя, которое выводится и сравнивается в главной функции.

2.5 Транспонирование матриц

После определения матрицы с наибольшим определителем выполняется функция trans() в которую передаётся нужная матрица matrix и её размерность. С помощью двух циклов for перебираются все элементы матрицы, при этом каждый элемент строки i и столбца j перемещается в строку j и столбец i в соответствии с формулой $a^T_{ij} = a_{ji}$. Для этого ипользуется следующий алгоритм:

1. Значение элемента a_{ij} записывается в переменную s.
2. Элемент a_{ij} приравнивается к элементу a_{ji}.
3. В элемент a_{ji} записывается значение переменной s.

После всех преобразований транспонированная матрица выводится в консоль.

Заключение

В ходе работы были изучены способы работы с матрицами, возможности основных циклов и применения рекурсии в функциях C++. На основе полученных знаний была написана программа способная создавать матрицы и производить их умножение, транспонирование и нахождение определителя, в соответствии с заданием на C++.

Полученные навыки самостоятельного изучения информации и создания программ, выполняющих основные действия с матрицами, способствует развитию у студентов навыков программирования.

Список использованных источников

1. Матрицы и определители URL - http://matematika.electrichelp.ru/matricy-i-opredeliteli/ .
2. Понижение порядка определителя. Разложение определителя по строке (столбцу). URL - https://math1.ru/education/matrix/detr.html .
3. Стивен Прата. Язык программирования C++. Лекции и упражнения. 6 издание, 2017.
4. Рекурсия в C++ URL - http://cppstudio.com/post/418/.

Исходный код

```
#include <iostream>
using namespace std;

void getMatrixWithoutRowAndCol(double** matrix, int N, int row, int col,
double** newMatrix) {
    int offsetRow = 0;
    int offsetCol = 0;

    for (int i = 0; i < N - 1; i++) {
        if (i == row) {
            offsetRow = 1;
        }
        offsetCol = 0;
        for (int j = 0; j < N - 1; j++) {
            if (j == col) {
                offsetCol = 1;
            }
            newMatrix[i][j] = matrix[i + offsetRow][j + offsetCol];
        }
    }
}
int matrixDet(double** matrix, int N) {
    int det = 0;
    int degree = 1;
    if (N == 1) {
        return matrix[0][0];
    }
    else if (N == 2) {
        return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
    }
    else {
        double** newMatrix = new double* [N - 1];
        for (int i = 0; i < N - 1; i++) {
            newMatrix[i] = new double[N - 1];
        }

        for (int j = 0; j < N; j++) {
            getMatrixWithoutRowAndCol(matrix, N, 0, j, newMatrix);
```

```
            det = det + (degree * matrix[0][j] * matrixDet(newMatrix, N -
1));
            degree = -degree;
        }
        for (int i = 0; i < N - 1; i++) {
            delete[] newMatrix[i];
        }
        delete[] newMatrix;
    }
    return det;
}

void trans(double** matrix, int N) {
    int i, j;
    double s;

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

        for (j = i + 1; j < N; j++) {
            s = matrix[i][j];
            matrix[i][j] = matrix[j][i];
            matrix[j][i] = s;
        }
    }
    cout << endl << "Траспонированная матрица с большим определителем" <<
endl;
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < N; j++)
        {
            cout << matrix[i][j] << " ";
        }

        cout << endl;
    }
}
void multiplication(double** A, double** B, int N) {
    int i, j, k;
    double** C;
    C = new double* [N];

    for (i = 0; i < N; i++)
    {
        C[i] = new double[N];
```

```
        for (j = 0; j < N; j++)
        {
            C[i][j] = 0;

            for (k = 0; k < N; k++) {

                C[i][j] += A[i][k] * B[k][j];

            }

        }
    }
    cout << endl << "Матрица произведения" << endl;
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < N; j++)
        {
            cout << C[i][j] << " ";
        }

        cout << endl;
    }
}

int main()
{
    srand(time(0));
    double** A, ** B;
    int min, max, i, j, k, detA, detB, N;
    //Использование кириллицы
    system("chcp 1251");
    system("cls");

    cout << "Введите размерность матриц" << endl;
    cin >> N;
    cout << "Введите минимальное значение элементов матрицы" << endl;
    cin >> min;
    cout << "Введите максимальное значение элементов матрицы" << endl;
    cin >> max;

    A = new double* [N];
    B = new double* [N];
```

```
    for (i = 0; i < N; i++) {
        A[i] = new double[N];
        B[i] = new double[N];

        for (j = 0; j < N; j++) {
             A[i][j] = min + rand() % (max - min + 1);
            B[i][j] = min + rand() % (max - min + 1);
         }
    }
            cout << endl << "Первая матрица" << endl;
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < N; j++)
        {
            cout << A[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl << "Вторая матрица" << endl;
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < N; j++)
        {
            cout << B[i][j] << " ";
        }
        cout << endl;
    }
    multiplication(A, B, N);

    detA = matrixDet(A, N);
    detB = matrixDet(B, N);

    cout << endl << "Определитель первой матрицы" << endl << detA << endl;
    cout << endl << "Определитель второй матрицы" << endl << detB << endl;

    if (detA > detB) {
        trans(A, N);
    }
    else {
        trans(B, N);
    }
    return 0;
}
```

Пример работы программы

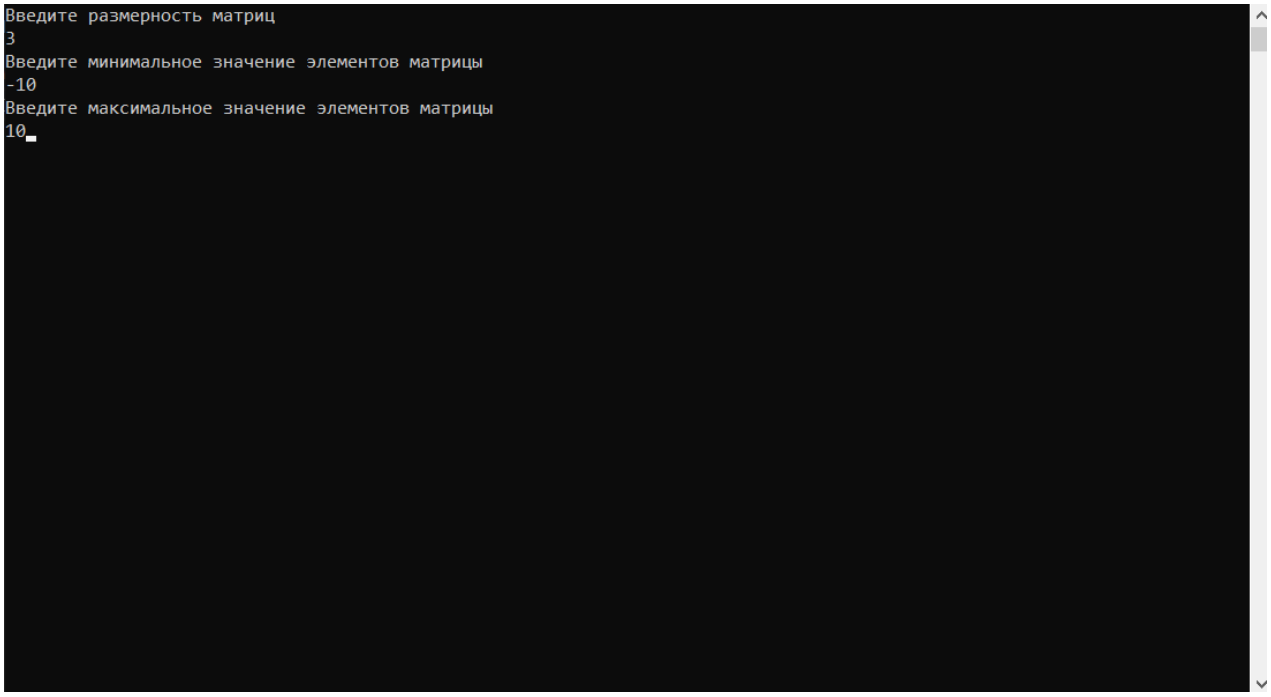


Рис.7 - Внесение начальных данных.

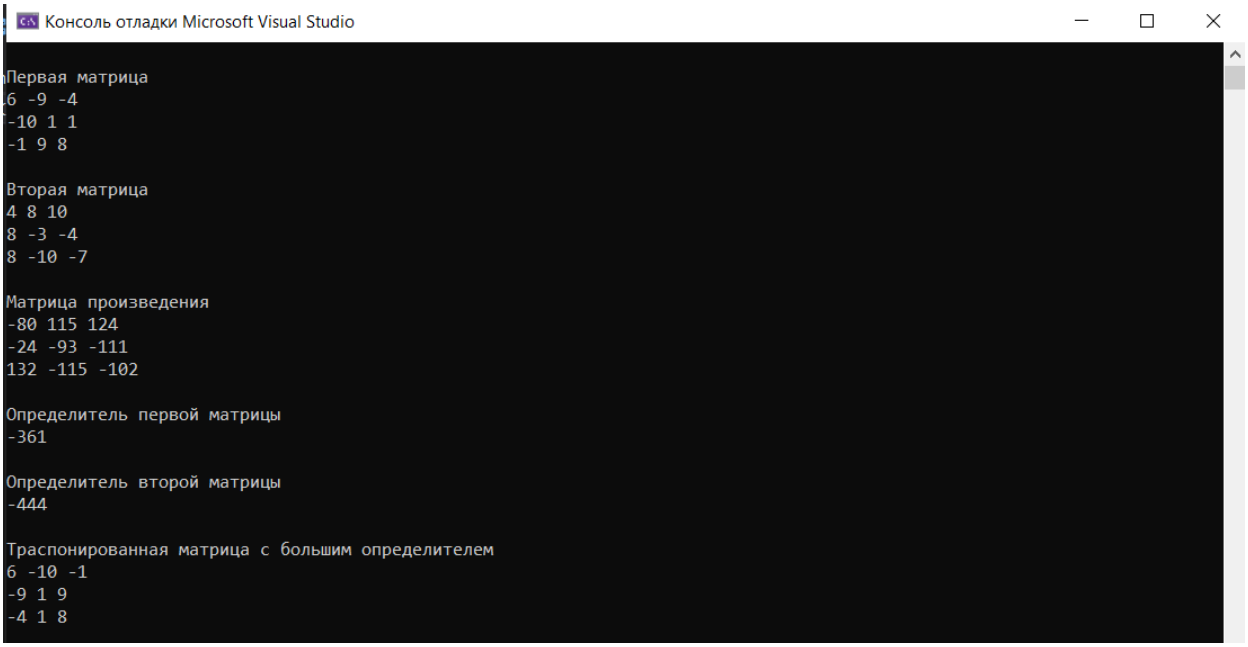


Рис.8 - Вывод результатов вычислений с сопроводительным текстом.