Министерство науки и высшего образования РФ
ФГАОУ ВПО
Национальный исследовательский технологический университет «МИСиС»

Институт Информационных технологий и компьютерных наук (ИТКН)

Кафедра Инфокоммуникационных технологий (ИКТ)

Отчет по лабораторной работе №4
по дисциплине «Программирование и Алгоритмизация»
на тему «Матрицы. Типовые алгоритмы обработки матриц»

Выполнил:
студент группы БИВТ-24-5

Черных Богдан

Проверил:
Стучилин В. В.

Москва, 2024

Цель работы: целью лабораторной работы является разработка программы для обработки матриц, которая позволяет отработать навыки работы с массивами и их алгоритмами. Программа должна обеспечивать выполнение операций с массивами произвольного размера, включая базовые алгоритмы и более сложные преобразования, такие как обнуление элементов по диагоналям и удаление строк с нулевыми элементами. Кроме того, программа должна быть реализована в двух вариантах представления матриц — как двумерный массив и как одномерная последовательность — и предоставлять удобный вывод результатов для проверки и анализа.

Код требуемых заданий (1 уровень 25 задание, 1 уровень 26 задание, 1 уровень 27 задание, 2 уровень 7 задание, 2 уровень 8 задание, 2 уровень 9 задание, 3 уровень 11 задание, 3 уровень 12 задание, 3 уровень 13 задание).

(В коде используется double checker только для удобства – из-за того, что я использую одни и те же название переменных, например a(array), если бы не базовое условие(if), мне бы пришлось каждый раз сбрасывать значение, или код бы у меня просто некорректно работал. Поэтому перед каждой новой задачей я проверяю checker).

Код:

```
//svg does precious
using System;                           // Аналог <iostream> для работы с консолью
и основными функциями
using System.Collections.Generic;       // Аналог <vector>, <list>, <map>, <set>,
<unordered_map>, <unordered_set>, <stack>, <queue>
using System.Text;                      // Аналог <string>, <cstring> (для работы
со строками и StringBuilder)
using System.Linq;                      // Аналог <algorithm> (для работы с LINQ,
сортировок, поиска и т.д.)
using System.IO;                        // Аналог <cstdio>, <fstream> (работа с
файлами)
using System.Globalization;             // Аналог <iomanip> (для форматирования)
using System.Collections;               // Работа с различными коллекциями
(например, ArrayList)
using System.Threading;                 // Потоки и многопоточность
using System.Runtime.Serialization;     // Аналог <stdexcept> (работа с
исключениями)
using System.Reflection;                // Аналог <typeinfo> (информация о типах,
рефлексия)
using System.Diagnostics;               // Аналог <utility>, <std::pair>
(вспомогательные функции и классы)
using System.ComponentModel;            // Дополнительные утилиты и атрибуты
using System.Numerics;                  // Работа с большими числами и
математическими операциями
using System.Globalization;
using System.Diagnostics;
using System.Net;
using System.Numerics;
// Для работы с потоками данных:
using System.Threading.Tasks;           // Асинхронные задачи

// Для работы с датами и временем:
```

```
using System.Timers;                    // Для работы с таймерами и временем
using System.Collections.Generic;
using System.Text;
using System.Linq;
using System.IO; //important
using System.Globalization;
using System.Collections;
using System.Threading;
using System.Runtime.Serialization;
using System.Reflection;
using System.Diagnostics;
using System.ComponentModel;
using System.Numerics;
using System.Globalization;
using System.Diagnostics;
using System.Net;
using System.Numerics;
using System.Threading.Tasks;
using System.Security.Cryptography;
using System.Data;
using System.Data.SqlClient;
using System.Xml;
using System.Xml.Linq;
using System.Runtime.InteropServices;
using System.Security;
using System.Web;
using System.Media;
using System.Drawing;
using System.Configuration;
using System.Timers;
using System.Runtime.Remoting;
using System.Runtime.CompilerServices;
using System.Runtime.Versioning;
using System.CodeDom;
using System.CodeDom.Compiler;
using System.Collections.Concurrent;
using System.Runtime;
using System.Windows;
using System.Windows.Input;
using System.Security.Principal;
using System.Security.Permissions;
using System.Resources;
using C = System.Console; //console
using dl = System.Decimal;//decimal
using str = System.String;//string
using l = System.Int64;   //long
using u = System.UInt64;  //Ulong
using db = System.Double; //Double

/*
Izzspot - 19 years
Boogie B - 20 years
SJ - 21 years
Bandokay - life sentence

Youngest in the charge
OFB, we don't window shop
Bro caught him an opp and tried turn him off (Bow, bow)
In this X3, man's swervin' off (Skrr, skrr)
Free Boogie Bando, he got birded off (Free my bro, free my bro)
Whenever we get a burner loss
We just cop a next one and go burst it off (Ay)
Lil bro's tellin' me he got his earnings wrong
We just took him OT, now his trapline's gone (Ring, ring)
Hashtag
Bro backed this ting and just started squeezin' (Clarted)
When it broad day, it was freezin'
```

```
Hashtag fuckery, hashtag screamin'
One on the hand ting woi, left man leanin', leanin' (Fucker)
Show us cause it's good to feel it
Shortie's cooze and she must be dreamin'
*/
/* ¡Adelante Barcelona, adelante Cataluña! Visca el Barça! Visca Catalunya!
   ¡Al diablo con todos los demás, porque lo más importante en la vida es el
fútbol!
*/
//-------------------------------------------------------------------------------
-----
//-------------------------------------------------------------------------------
-----
//-------------------------------------------------------------------------------
-----




class Program
{//Лабораторная работа 4
 //1,25 - Уровень 1 Задание 25; 1,26 - Уровень 1 Задание 26;
 //1,27 - Уровень 1 Задание 27; 2,7 - Уровень 2 Задание 7;
 //2,8 - Уровень 2 Задание 8; 2,9 - Уровень 2 Задание 9;
 //3,11 - Уровень 3 Задание 11; 3,12 - Уровень 3 Задание 12;
 //3,13 - Уровень 3 Задание 13

    static void Main()
    {
        // УРОВЕНЬ 1
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        Random rand = new Random();

        Console.Write("Введите номер уровня (1,25 , 2,8 ...): ");
        double checker = Convert.ToDouble(Console.ReadLine());

        if (checker == 1.25)
        {
            // Задание 25
            Console.WriteLine("Задание 1.25: Матрица X\n");
            Console.Write("Введите количество строк матрицы X:  \n");
            int rowsX = Convert.ToInt32(Console.ReadLine());
            Console.Write("Введите количество столбцов матрицы X: \n");
            int colsX = Convert.ToInt32(Console.ReadLine());
            int[,] X = new int[rowsX, colsX];

            FillMatrix(X, rand);
            Console.WriteLine("Исходная матрица X:\n");
            PrintMatrix(X);

            // Локальная функция для обмена строками с макс и мин отрицательными
элементами
            void SwapRowsWithMinMaxNegative(int[,] matrix)
            {
                int minNegativesRow = -1;
                int maxNegativesRow = -1;
                int minNegatives = int.MaxValue, maxNegatives = int.MinValue;

                for (int i = 0; i < matrix.GetLength(0); i++)
                {
                    int negativesCount = 0;
                    for (int j = 0; j < matrix.GetLength(1); j++)
                    {
                        if (matrix[i, j] < 0) negativesCount++;
                    }
```

```
                    if (negativesCount < minNegatives)
                    {
                        minNegatives = negativesCount;
                        minNegativesRow = i;
                    }

                    if (negativesCount > maxNegatives)
                    {
                        maxNegatives = negativesCount;
                        maxNegativesRow = i;
                    }
                }

                if (minNegativesRow != -1 && maxNegativesRow != -1 &&
minNegativesRow != maxNegativesRow)
                {
                    for (int j = 0; j < matrix.GetLength(1); j++)
                    {
                        int temp = matrix[minNegativesRow, j];
                        matrix[minNegativesRow, j] = matrix[maxNegativesRow, j];
                        matrix[maxNegativesRow, j] = temp;
                    }
                    Console.WriteLine("\nМатрица X после замены строк с мин и макс
отрицательных элементов:\n");
                    PrintMatrix(matrix);
                }
                else
                {
                    Console.WriteLine("Замена строк не требуется.");
                }
            }

            SwapRowsWithMinMaxNegative(X);
        }
        else if (checker == 1.26)
        {
            // Задание 26
            Console.WriteLine("\nЗадание 1.26: Матрица A\n");
            Console.Write("Введите количество строк матрицы A: \n");
            int rowsA = Convert.ToInt32(Console.ReadLine());
            Console.Write("Введите количество столбцов матрицы A: \n");
            int colsA = Convert.ToInt32(Console.ReadLine());
            int[,] A = new int[rowsA, colsA];

            FillMatrix(A, rand);
            Console.WriteLine("Исходная матрица A:\n");
            PrintMatrix(A);

            int[] B = new int[colsA];
            Console.WriteLine("Введите элементы вектора B(вводить новое число в
каждой строке):");
            for (int i = 0; i < colsA; i++)
            {
                B[i] = Convert.ToInt32(Console.ReadLine());
            }

            void ReplaceRowWithMaxInColumn(int[,] matrix, int columnIndex, int[]
vector)
            {
                // Проверка, достаточно ли столбцов для выполнения операции
                if (columnIndex >= matrix.GetLength(1))
                {
                    Console.WriteLine($"Ошибка: В матрице недостаточно столбцов
для работы с индексом {columnIndex + 1}.");
                    return;
                }
```

```
                int maxRowIndex = 0;
                int maxValue = int.MinValue;

                for (int i = 0; i < matrix.GetLength(0); i++)
                {
                    if (matrix[i, columnIndex] > maxValue)
                    {
                        maxValue = matrix[i, columnIndex];
                        maxRowIndex = i;
                    }
                }

                for (int j = 0; j < matrix.GetLength(1); j++)
                {
                    matrix[maxRowIndex, j] = vector[j];
                }

                Console.WriteLine("\nМатрица A после замены строки вектора B:\n");
                PrintMatrix(matrix);
            }

            // Проверка, достаточно ли столбцов для использования индекса 5
            ReplaceRowWithMaxInColumn(A, 5, B);
        }

        else if (checker == 1.27)
        {
            // Задание 27
            Console.WriteLine("\nЗадание 1.27: Матрица B\n");
            Console.Write("Введите количество строк матрицы B: \n");
            int rowsB = Convert.ToInt32(Console.ReadLine());
            Console.Write("Введите количество столбцов матрицы B: \n");
            int colsB = Convert.ToInt32(Console.ReadLine());
            int[,] BMatrix = new int[rowsB, colsB];

            FillMatrix(BMatrix, rand);
            Console.WriteLine("Исходная матрица B:\n");
            PrintMatrix(BMatrix);

            void ReplaceColumnWithMaxElementsInReverse(int[,] matrix, int
columnIndex)
            {
                int[] maxElements = new int[matrix.GetLength(0)];

                for (int i = 0; i < matrix.GetLength(0); i++)
                {
                    maxElements[i] = matrix[i, 0];
                    for (int j = 1; j < matrix.GetLength(1); j++)
                    {
                        if (matrix[i, j] > maxElements[i])
                        {
                            maxElements[i] = matrix[i, j];
                        }
                    }
                }

                Array.Reverse(maxElements);

                for (int i = 0; i < matrix.GetLength(0); i++)
                {
                    matrix[i, columnIndex] = maxElements[i];
                }

                Console.WriteLine("\nМатрица B после замены 4-го столбца:\n");
                PrintMatrix(matrix);
            }
```

```
            // первый элемент в 4столбце заменяется на максимальный элемент
последней строки(5),
            // второй элемент в 4столбце заменяется на максимальный элемент
предпоследней строки(4) и тд
            ReplaceColumnWithMaxElementsInReverse(BMatrix, 3);
        }
        //
        //
        //
        //
        //
        //
        //
        //
        //
        // УРОВЕНЬ 2

        else if (checker == 2.7)
        {
            // Задание 7: Найти максимальный элемент на главной диагонали и
заменить нулями элементы правее диагонали выше строки с этим элементом
            Console.WriteLine("\nЗадание 2.7: Матрица A (размер 6x6)");
            int size = 6;
            int[,] A = new int[size, size];

            FillMatrix(A, rand);
            Console.WriteLine("Исходная матрица A:\n");
            PrintMatrix(A);

            void ReplaceElementsAboveRowWithMaxOnDiagonal(int[,] matrix)
            {
                int maxDiagonalValue = int.MinValue;
                int maxDiagonalRow = 0;

                // Находим максимальный элемент на главной диагонали и его строку
                for (int i = 0; i < matrix.GetLength(0); i++)
                {
                    if (matrix[i, i] > maxDiagonalValue)
                    {
                        maxDiagonalValue = matrix[i, i];
                        maxDiagonalRow = i;
                    }
                }

                // Заменяем элементы правее главной диагонали нулями выше строки с
максимальным элементом на диагонали
                for (int i = 0; i < maxDiagonalRow; i++)
                {
                    for (int j = i + 1; j < matrix.GetLength(1); j++)
                    {
                        matrix[i, j] = 0;
                    }
                }

                Console.WriteLine("\nМатрица A после замены элементов правее
диагонали выше строки с макс. диагональным элементом:\n");
                PrintMatrix(matrix);
            }

            ReplaceElementsAboveRowWithMaxOnDiagonal(A);
        }


        else if (checker == 2.8)
        {
            // Задание 8: Поменять местами максимальные элементы 1-й и 2-й строк,
3-й и 4-й, 5-й и 6-й строк
```

```
            Console.WriteLine("\nЗадание 2.8: Матрица B (размер 6x6)");
            int size = 6;
            int[,] B = new int[size, size];

            FillMatrix(B, rand);
            Console.WriteLine("Исходная матрица B:\n");
            PrintMatrix(B);

            void SwapMaxElementsInPairs(int[,] matrix)
            {
                for (int i = 0; i < matrix.GetLength(0); i += 2)
                {
                    int maxIndex1 = 0;
                    int maxIndex2 = 0;
                    int max1 = matrix[i, 0];
                    int max2 = matrix[i + 1, 0];

                    // Поиск максимальных элементов в каждой паре строк
                    for (int j = 1; j < matrix.GetLength(1); j++)
                    {
                        if (matrix[i, j] > max1)
                        {
                            max1 = matrix[i, j];
                            maxIndex1 = j;
                        }
                        if (matrix[i + 1, j] > max2)
                        {
                            max2 = matrix[i + 1, j];
                            maxIndex2 = j;
                        }
                    }

                    // Замена максимальных элементов между строками
                    int temp = matrix[i, maxIndex1];
                    matrix[i, maxIndex1] = matrix[i + 1, maxIndex2];
                    matrix[i + 1, maxIndex2] = temp;
                }

                Console.WriteLine("\nМатрица B после замены максимальных элементов
пар строк:\n");
                PrintMatrix(matrix);
            }

            SwapMaxElementsInPairs(B);
        }

        else if (checker == 2.9)
        {
            // Задание 9: Расположить элементы строк матрицы A (размер 6x7) в
обратном порядке
            Console.WriteLine("\nЗадание 2.9: Матрица A (размер 6x7)");
            int rows = 6;
            int cols = 7;
            int[,] A = new int[rows, cols];

            FillMatrix(A, rand);
            Console.WriteLine("Исходная матрица A:\n");
            PrintMatrix(A);

            void ReverseElementsInRows(int[,] matrix)
            {
                for (int i = 0; i < matrix.GetLength(0); i++)
                {
                    int left = 0;
                    int right = matrix.GetLength(1) - 1;

                    // Переворачиваем элементы в строке
```

```
                    while (left < right)
                    {
                        int temp = matrix[i, left];
                        matrix[i, left] = matrix[i, right];
                        matrix[i, right] = temp;
                        left++;
                        right--;
                    }
                }

                Console.WriteLine("\nМатрица A после разворота элементов
строк:\n");
                PrintMatrix(matrix);
            }

            ReverseElementsInRows(A);
        }
        //
        //
        //
        //
        //
        //
        //
        //
        //
        //
        // УРОВЕНЬ 3

        if (checker == 3.11)
        {
            // Задание 11: Удалить все строки, содержащие нулевые элементы
            Console.WriteLine("\nЗадание 3.11: Удаление строк с нулевыми
элементами (двумерный массив)");
            int rows = 5, cols = 5; // Пример с фиксированным размером
            int[,] matrixA = new int[rows, cols];

            FillMatrix(matrixA, rand);
            Console.WriteLine("Исходная матрица:");
            PrintMatrix(matrixA);

            void RemoveRowsWithZeros(int[,] mat)
            {
                bool[] rowsToDelete = new bool[mat.GetLength(0)];

                // Отмечаем строки для удаления
                for (int i = 0; i < mat.GetLength(0); i++)
                {
                    for (int j = 0; j < mat.GetLength(1); j++)
                    {
                        if (mat[i, j] == 0)
                        {
                            rowsToDelete[i] = true;
                            break;
                        }
                    }
                }

                // Формируем новую матрицу без отмеченных строк
                int newRowCount = rowsToDelete.Count(r => !r);
                int[,] newMatrix = new int[newRowCount, mat.GetLength(1)];
                int newRow = 0;

                for (int i = 0; i < mat.GetLength(0); i++)
                {
                    if (!rowsToDelete[i])
                    {
```

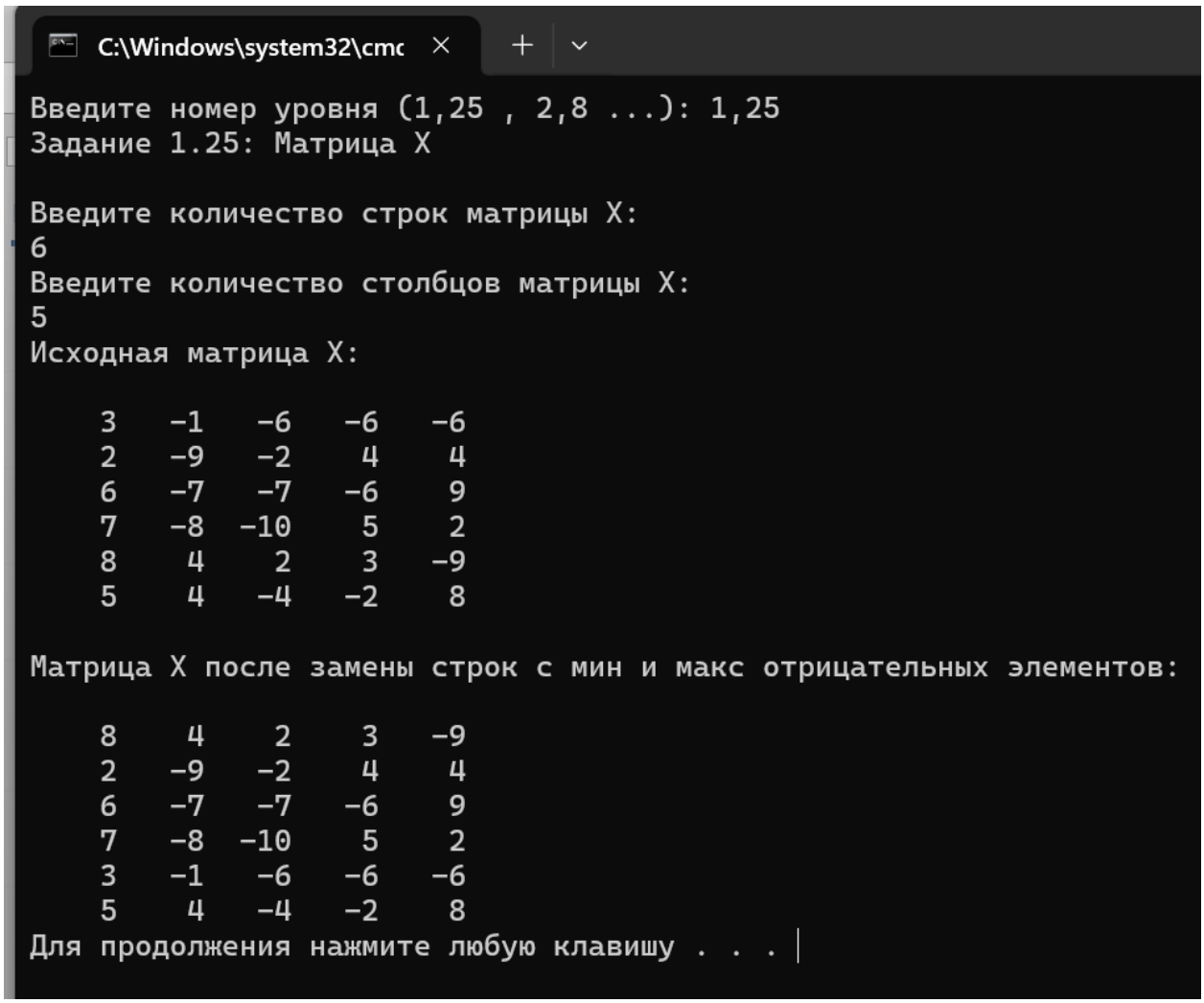
```
                    for (int j = 0; j < mat.GetLength(1); j++)
                    {
                        newMatrix[newRow, j] = mat[i, j];
                    }
                    newRow++;
                }
            }

            Console.WriteLine("\nМатрица после удаления строк с нулями:");
            PrintMatrix(newMatrix);
        }

        RemoveRowsWithZeros(matrixA);

        Console.WriteLine("\nВариант (б): Удаление строк с нулевыми элементами
в одномерном массиве");
        int[] oneDMatrixA = new int[rows * cols];
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                oneDMatrixA[i * cols + j] = matrixA[i, j];
            }
        }

        void RemoveRowsWithZerosInOneD(int[] mat, int rowCount, int colCount)
        {
            bool[] rowsToDelete = new bool[rowCount];

            for (int i = 0; i < rowCount; i++)
            {
                for (int j = 0; j < colCount; j++)
                {
                    if (mat[i * colCount + j] == 0)
                    {
                        rowsToDelete[i] = true;
                        break;
                    }
                }
            }

            Console.WriteLine("\nРезультат в одномерной последовательности:");
            for (int i = 0; i < rowCount; i++)
            {
                if (!rowsToDelete[i])
                {
                    for (int j = 0; j < colCount; j++)
                    {
                        Console.Write($"{mat[i * colCount + j],5}");
                    }
                    Console.WriteLine();
                }
            }
        }

        RemoveRowsWithZerosInOneD(oneDMatrixA, rows, cols);
    }

    else if (checker == 3.12)
    {
        // Задание 12: Привести матрицу к виду с нулями ниже главной диагонали
        Console.WriteLine("\nЗадание 3.12: Обнуление элементов ниже главной
диагонали (двумерный массив)");
        int matrixSize = 4; // Пример квадратной матрицы, можно менять
        int[,] matrixB = new int[matrixSize, matrixSize];

        FillMatrix(matrixB, rand);
```

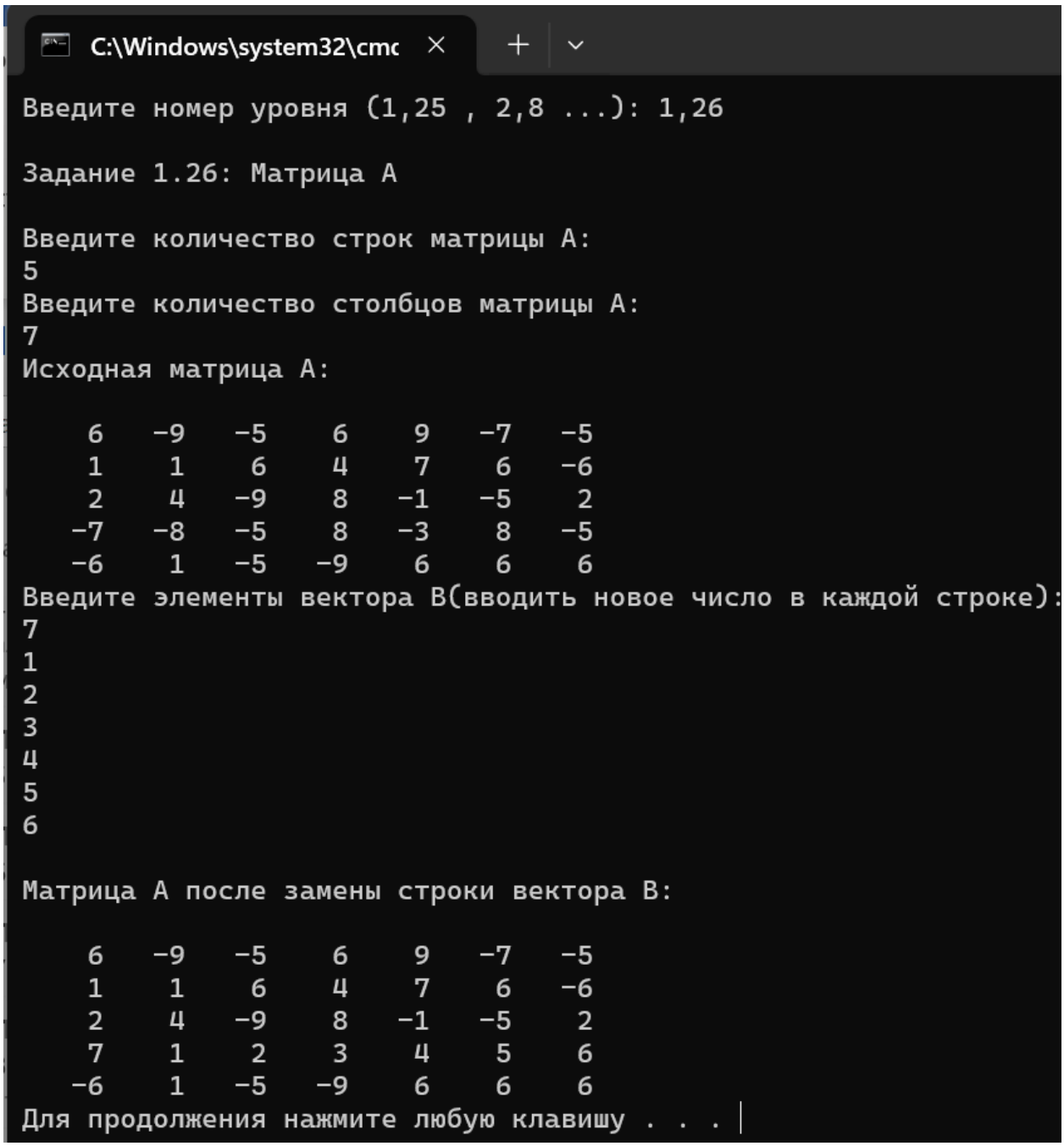
```
            Console.WriteLine("Исходная матрица:");
            PrintMatrix(matrixB);

            void ToUpperTriangularMatrix(int[,] mat)
            {
                for (int j = 0; j < mat.GetLength(1) - 1; j++)
                {
                    for (int k = j + 1; k < mat.GetLength(0); k++)
                    {
                        double p = (double)mat[k, j] / mat[j, j];
                        for (int m = j; m < mat.GetLength(1); m++)
                        {
                            mat[k, m] -= (int)(mat[j, m] * p);
                        }
                    }
                }

                Console.WriteLine("\nМатрица после обнуления элементов ниже
главной диагонали:");
                PrintMatrix(mat);
            }

            ToUpperTriangularMatrix(matrixB);

            Console.WriteLine("\nВариант (б): Обнуление элементов ниже главной
диагонали в одномерной последовательности");
            int[] oneDMatrixB = new int[matrixSize * matrixSize];
            for (int i = 0; i < matrixSize; i++)
            {
                for (int j = 0; j < matrixSize; j++)
                {
                    oneDMatrixB[i * matrixSize + j] = matrixB[i, j];
                }
            }

            void ToUpperTriangularInOneD(int[] mat, int size)
            {
                for (int j = 0; j < size - 1; j++)
                {
                    for (int k = j + 1; k < size; k++)
                    {
                        double p = (double)mat[k * size + j] / mat[j * size + j];
                        for (int m = j; m < size; m++)
                        {
                            mat[k * size + m] -= (int)(mat[j * size + m] * p);
                        }
                    }
                }

                Console.WriteLine("\nРезультат в одномерной последовательности:");
                for (int i = 0; i < size; i++)
                {
                    for (int j = 0; j < size; j++)
                    {
                        Console.Write($"{mat[i * size + j],5}");
                    }
                    Console.WriteLine();
                }
            }

            ToUpperTriangularInOneD(oneDMatrixB, matrixSize);
        }

        else if (checker == 3.13)
        {
            // Задание 13: Привести матрицу к виду с нулями выше главной диагонали
```

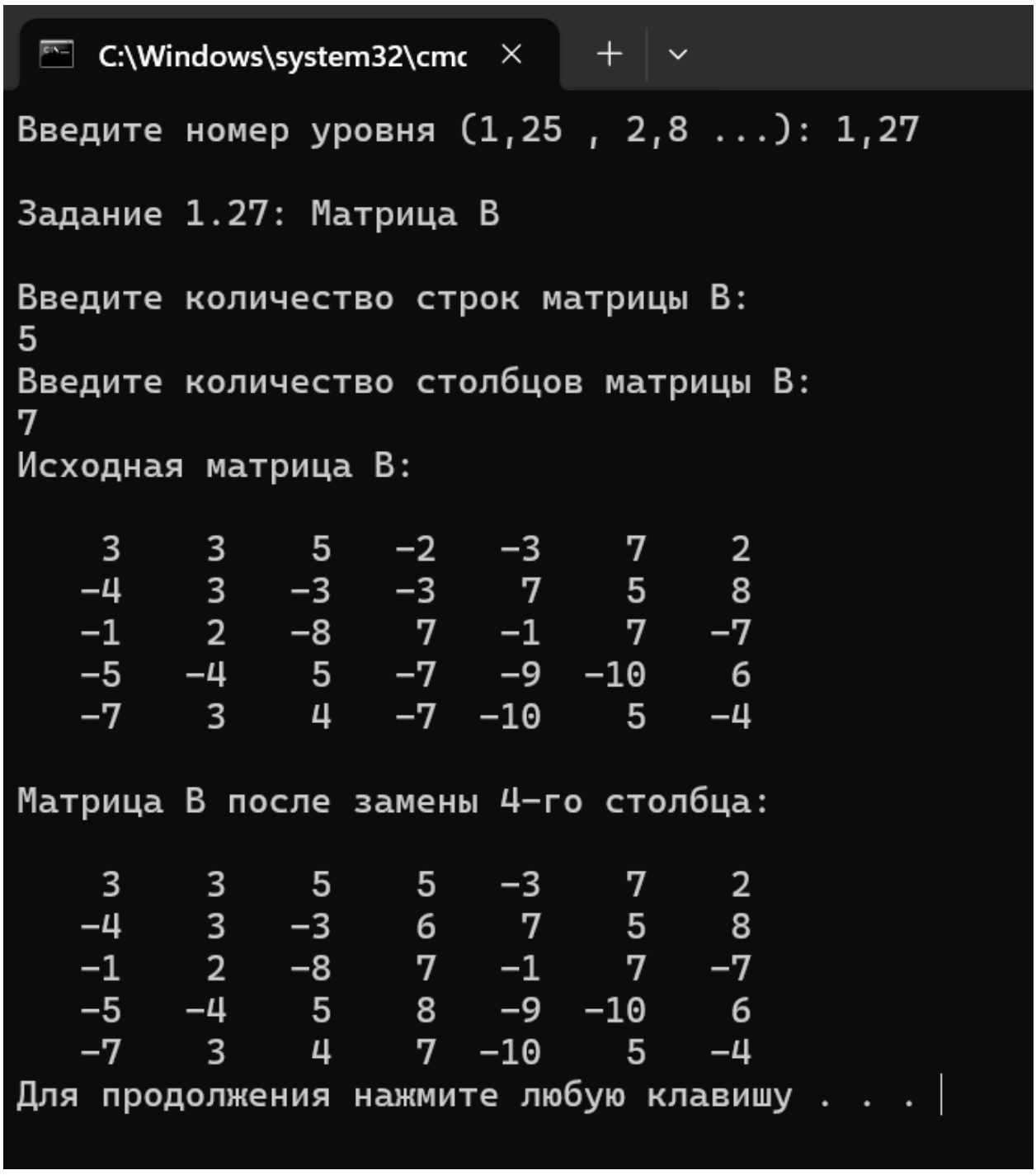
```
            Console.WriteLine("\nЗадание 3.13: Обнуление элементов выше главной
диагонали (двумерный массив)");
            int matrixSize = 4;
            int[,] matrixC = new int[matrixSize, matrixSize];

            FillMatrix(matrixC, rand);
            Console.WriteLine("Исходная матрица:");
            PrintMatrix(matrixC);

            void ToLowerTriangularMatrix(int[,] mat)
            {
                for (int j = mat.GetLength(1) - 1; j > 0; j--)
                {
                    for (int k = j - 1; k >= 0; k--)
                    {
                        double p = (double)mat[k, j] / mat[j, j];
                        for (int m = j; m >= 0; m--)
                        {
                            mat[k, m] -= (int)(mat[j, m] * p);
                        }
                    }
                }

                Console.WriteLine("\nМатрица после обнуления элементов выше
главной диагонали:");
                PrintMatrix(mat);
            }

            ToLowerTriangularMatrix(matrixC);

            Console.WriteLine("\nВариант (б): Обнуление элементов выше главной
диагонали в одномерной последовательности");
            int[] oneDMatrixC = new int[matrixSize * matrixSize];
            for (int i = 0; i < matrixSize; i++)
            {
                for (int j = 0; j < matrixSize; j++)
                {
                    oneDMatrixC[i * matrixSize + j] = matrixC[i, j];
                }
            }

            void ToLowerTriangularInOneD(int[] mat, int size)
            {
                for (int j = size - 1; j > 0; j--)
                {
                    for (int k = j - 1; k >= 0; k--)
                    {
                        double p = (double)mat[k * size + j] / mat[j * size + j];
                        for (int m = j; m >= 0; m--)
                        {
                            mat[k * size + m] -= (int)(mat[j * size + m] * p);
                        }
                    }
                }

                Console.WriteLine("\nРезультат в одномерной последовательности:");
                for (int i = 0; i < size; i++)
                {
                    for (int j = 0; j < size; j++)
                    {
                        Console.Write($"{mat[i * size + j],5}");
                    }
                    Console.WriteLine();
                }
            }
            ToLowerTriangularInOneD(oneDMatrixC, matrixSize);
        }
```

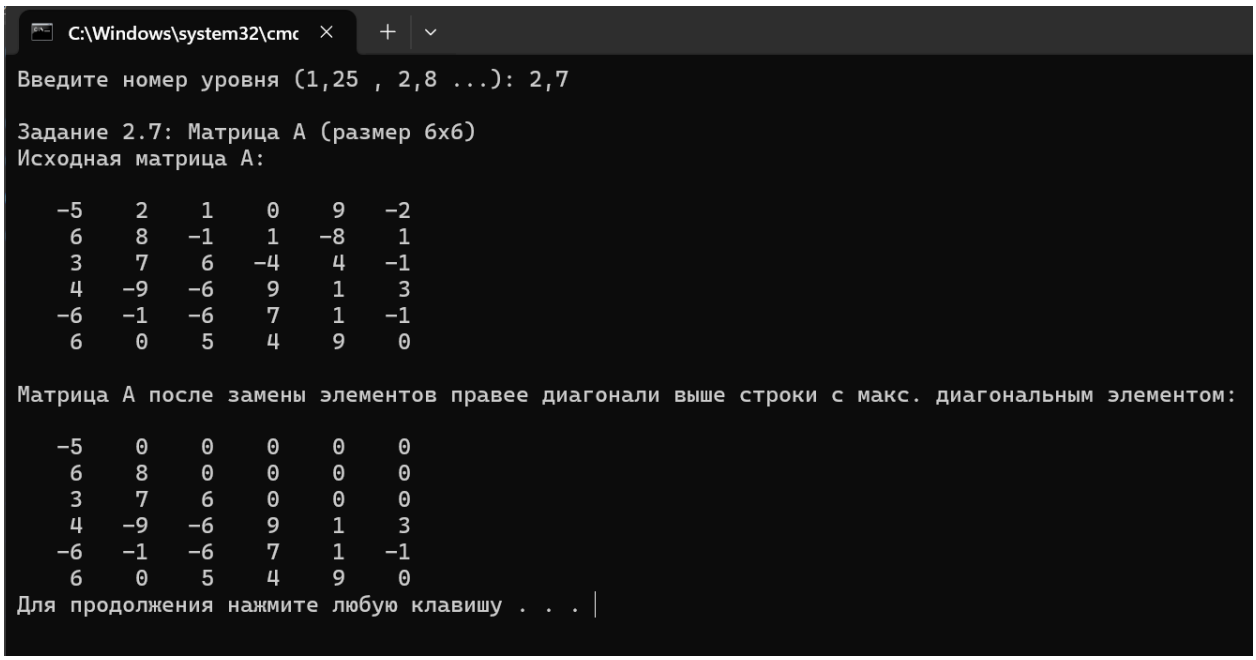
```
    }

    //voids for code
    static void FillMatrix(int[,] matrix, Random rand)
    {// Заполняет матрицы рандомными числами
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                matrix[i, j] = rand.Next(-10, 10); // Случайные числа от -10 до 10
            }
        }
    }

    static void PrintMatrix(int[,] matrix)
    {// Выводит саму матрицу
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                Console.Write($"{matrix[i, j],5}");
            }
            Console.WriteLine();
        }
    }
}
```

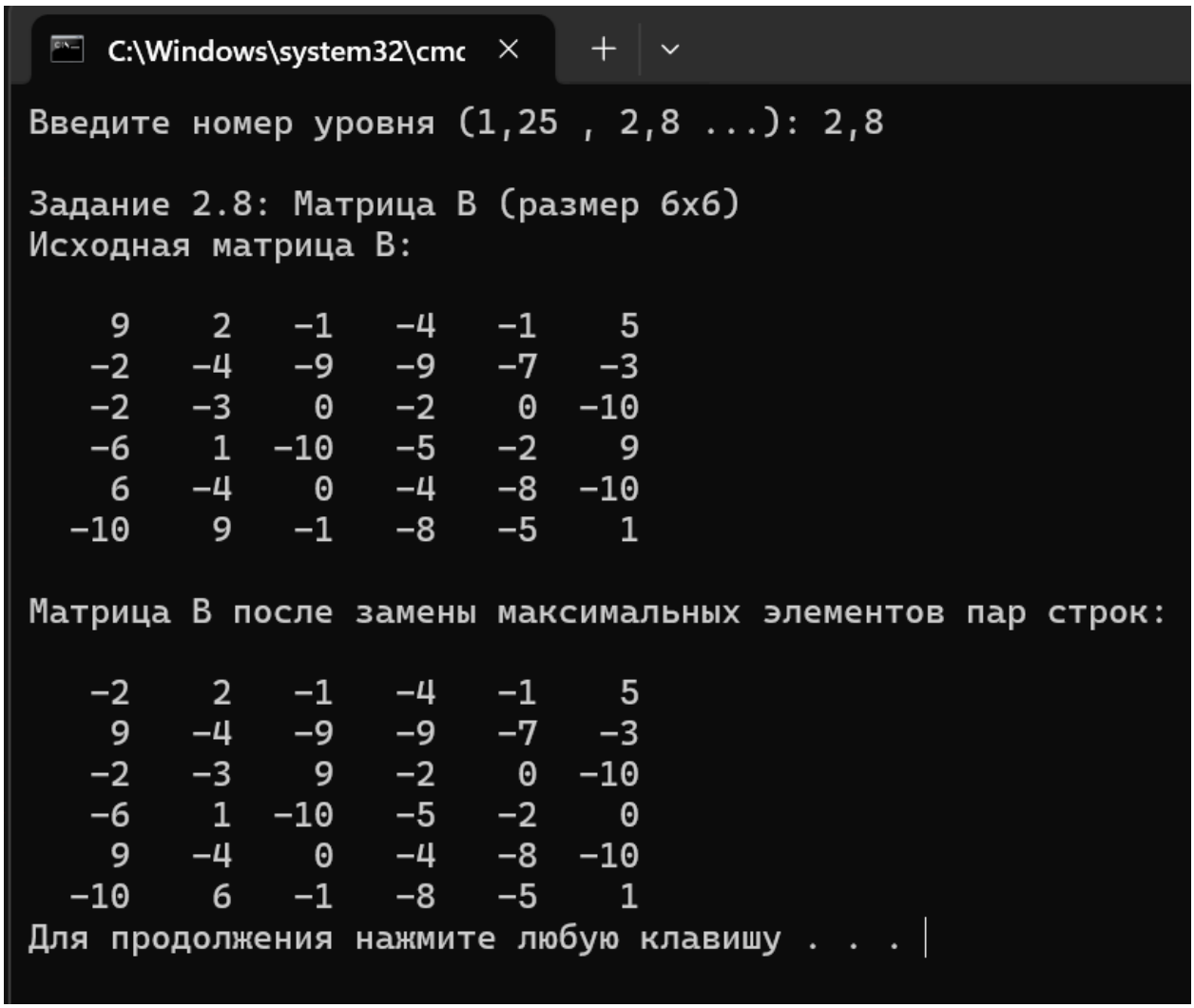
Выполнение всех заданий:

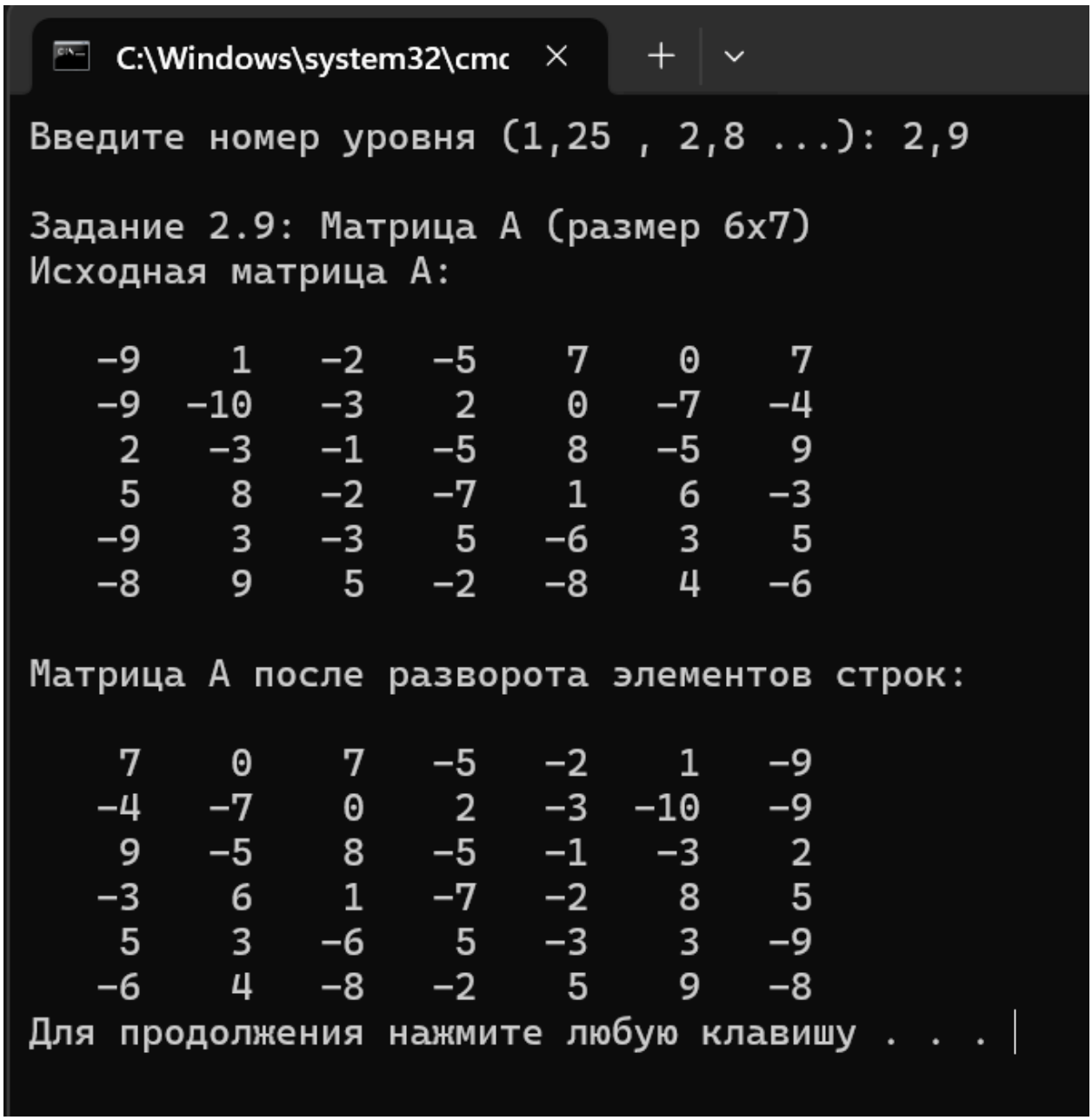

C:\Windows\system32\cmc
Введите номер уровня (1,25 , 2,8 ...): 1,25
Задание 1.25: Матрица X
Введите количество строк матрицы X:
6
Введите количество столбцов матрицы X:
5
Исходная матрица X:
3 -1 -6 -6 -6
2 -9 -2 4 4
6 -7 -7 -6 9
7 -8 -10 5 2
8 4 2 3 -9
5 4 -4 -2 8
Матрица X после замены строк с мин и макс отрицательных элементов:
8 4 2 3 -9
2 -9 -2 4 4
6 -7 -7 -6 9
7 -8 -10 5 2
3 -1 -6 -6 -6
5 4 -4 -2 8
Для продолжения нажмите любую клавишу . . .

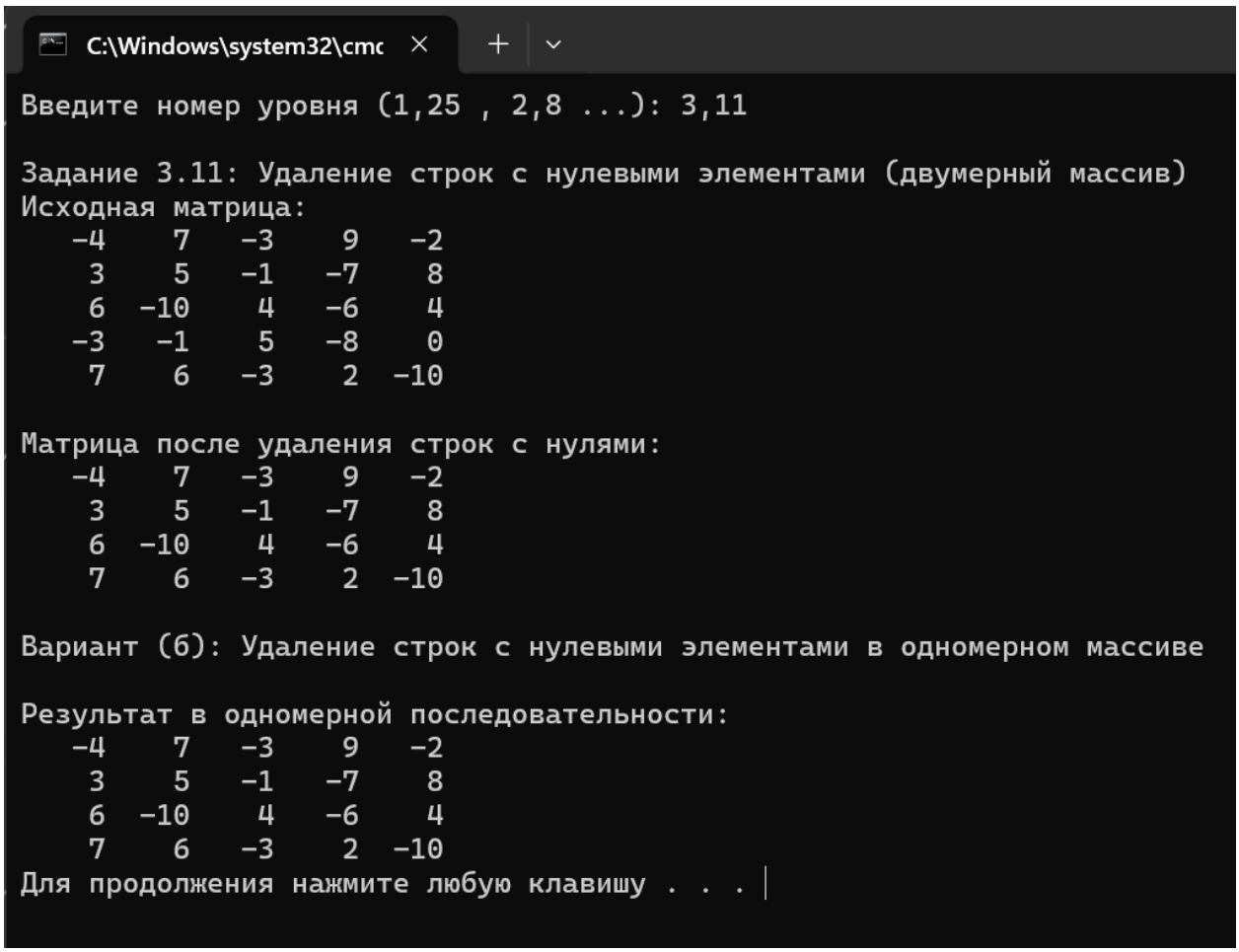

C:\Windows\system32\cmc
Введите номер уровня (1,25 , 2,8 ...): 1,26
Задание 1.26: Матрица A
Введите количество строк матрицы A:
5
Введите количество столбцов матрицы A:
7
Исходная матрица A:
6 -9 -5 6 9 -7 -5
1 1 6 4 7 6 -6
2 4 -9 8 -1 -5 2
-7 -8 -5 8 -3 8 -5
-6 1 -5 -9 6 6 6
Введите элементы вектора B(вводить новое число в каждой строке):
7
1
2
3
4
5
6
Матрица A после замены строки вектора B:
6 -9 -5 6 9 -7 -5
1 1 6 4 7 6 -6
2 4 -9 8 -1 -5 2
7 1 2 3 4 5 6
-6 1 -5 -9 6 6 6
Для продолжения нажмите любую клавишу . . .

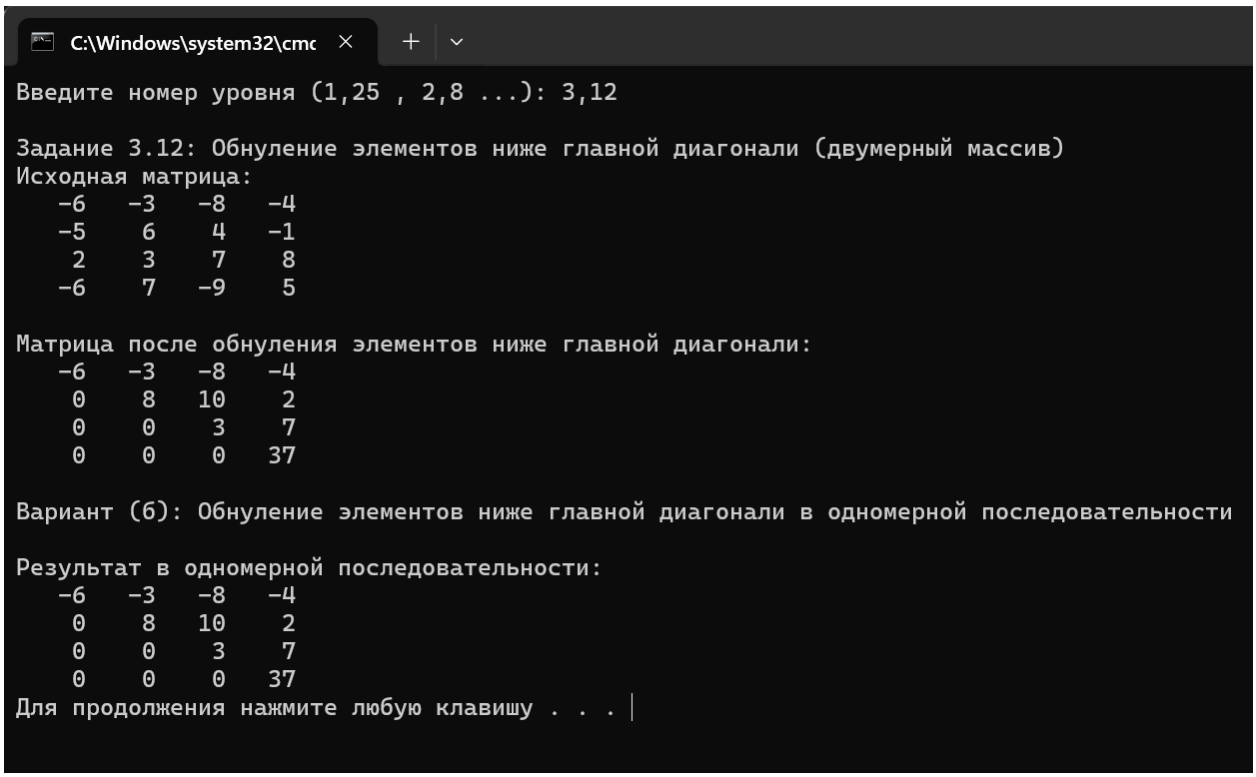

C:\Windows\system32\cmc
Введите номер уровня (1,25 , 2,8 ...): 1,27
Задание 1.27: Матрица B
Введите количество строк матрицы B:
5
Введите количество столбцов матрицы B:
7
Исходная матрица B:
3 3 5 -2 -3 7 2
-4 3 -3 -3 7 5 8
-1 2 -8 7 -1 7 -7
-5 -4 5 -7 -9 -10 6
-7 3 4 -7 -10 5 -4
Матрица B после замены 4-го столбца:
3 3 5 5 -3 7 2
-4 3 -3 6 7 5 8
-1 2 -8 7 -1 7 -7
-5 -4 5 8 -9 -10 6
-7 3 4 7 -10 5 -4
Для продолжения нажмите любую клавишу . . .

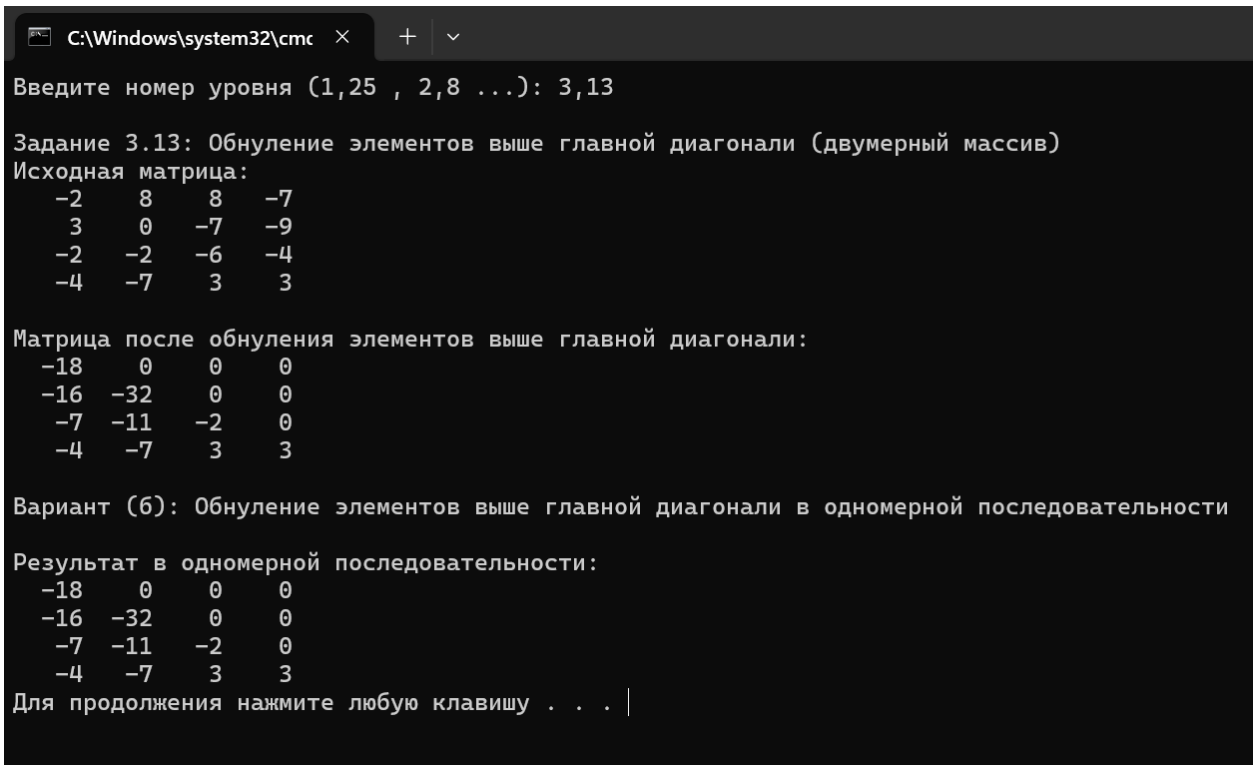

C:\Windows\system32\cmc
Введите номер уровня (1,25 , 2,8 ...): 2,7
Задание 2.7: Матрица A (размер 6x6)
Исходная матрица A:
-5 2 1 0 9 -2
6 8 -1 1 -8 1
3 7 6 -4 4 -1
4 -9 -6 9 1 3
-6 -1 -6 7 1 -1
6 0 5 4 9 0
Матрица A после замены элементов правее диагонали выше строки с макс. диагональным элементом:
-5 0 0 0 0 0
6 8 0 0 0 0
3 7 6 0 0 0
4 -9 -6 9 1 3
-6 -1 -6 7 1 -1
6 0 5 4 9 0
Для продолжения нажмите любую клавишу . . .


C:\Windows\system32\cmc
Введите номер уровня (1,25 , 2,8 ...): 2,8
Задание 2.8: Матрица B (размер 6x6)
Исходная матрица B:
9 2 -1 -4 -1 5
-2 -4 -9 -9 -7 -3
-2 -3 0 -2 0 -10
-6 1 -10 -5 -2 9
6 -4 0 -4 -8 -10
-10 9 -1 -8 -5 1
Матрица B после замены максимальных элементов пар строк:
-2 2 -1 -4 -1 5
9 -4 -9 -9 -7 -3
-2 -3 9 -2 0 -10
-6 1 -10 -5 -2 0
9 -4 0 -4 -8 -10
-10 6 -1 -8 -5 1
Для продолжения нажмите любую клавишу . . .

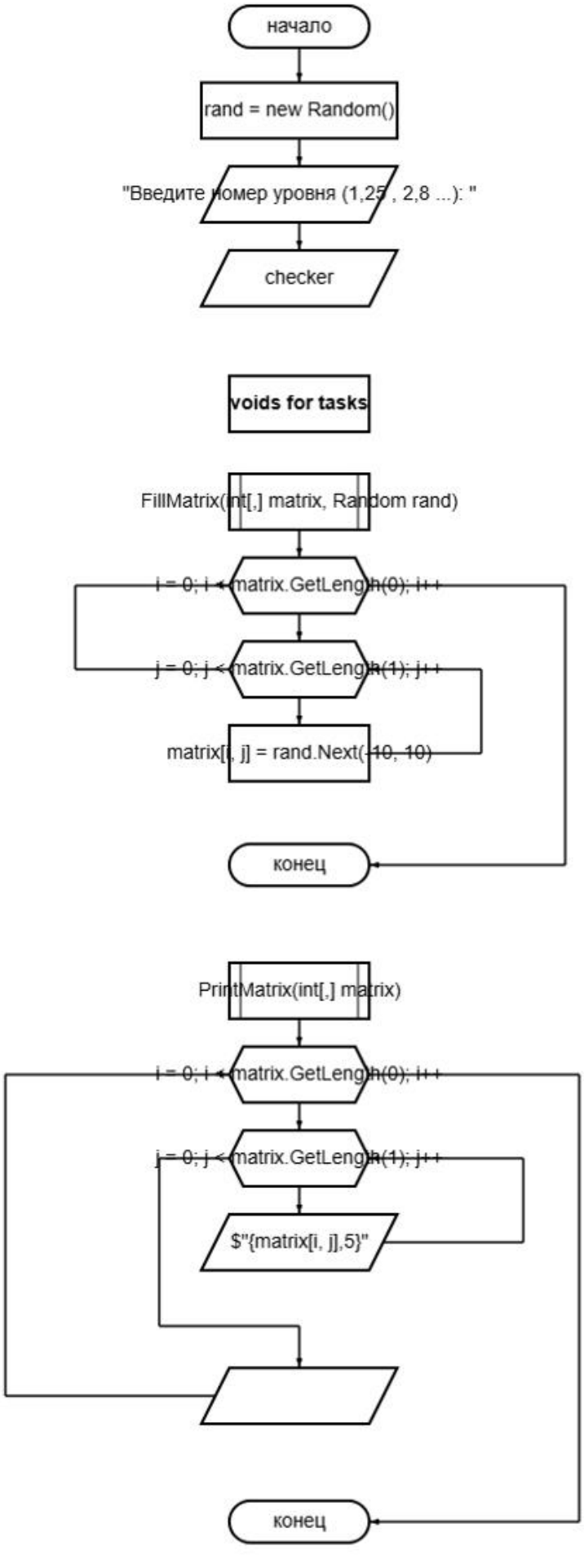

C:\Windows\system32\cmc
Введите номер уровня (1,25 , 2,8 ...): 2,9
Задание 2.9: Матрица A (размер 6x7)
Исходная матрица A:
-9 1 -2 -5 7 0 7
-9 -10 -3 2 0 -7 -4
2 -3 -1 -5 8 -5 9
5 8 -2 -7 1 6 -3
-9 3 -3 5 -6 3 5
-8 9 5 -2 -8 4 -6
Матрица A после разворота элементов строк:
7 0 7 -5 -2 1 -9
-4 -7 0 2 -3 -10 -9
9 -5 8 -5 -1 -3 2
-3 6 1 -7 -2 8 5
5 3 -6 5 -3 3 -9
-6 4 -8 -2 5 9 -8
Для продолжения нажмите любую клавишу . . .


C:\Windows\system32\cmc
Введите номер уровня (1,25 , 2,8 ...): 3,11
Задание 3.11: Удаление строк с нулевыми элементами (двумерный массив)
Исходная матрица:
-4 7 -3 9 -2
3 5 -1 -7 8
6 -10 4 -6 4
-3 -1 5 -8 0
7 6 -3 2 -10
Матрица после удаления строк с нулями:
-4 7 -3 9 -2
3 5 -1 -7 8
6 -10 4 -6 4
7 6 -3 2 -10
Вариант (б): Удаление строк с нулевыми элементами в одномерном массиве
Результат в одномерной последовательности:
-4 7 -3 9 -2
3 5 -1 -7 8
6 -10 4 -6 4
7 6 -3 2 -10
Для продолжения нажмите любую клавишу . . .

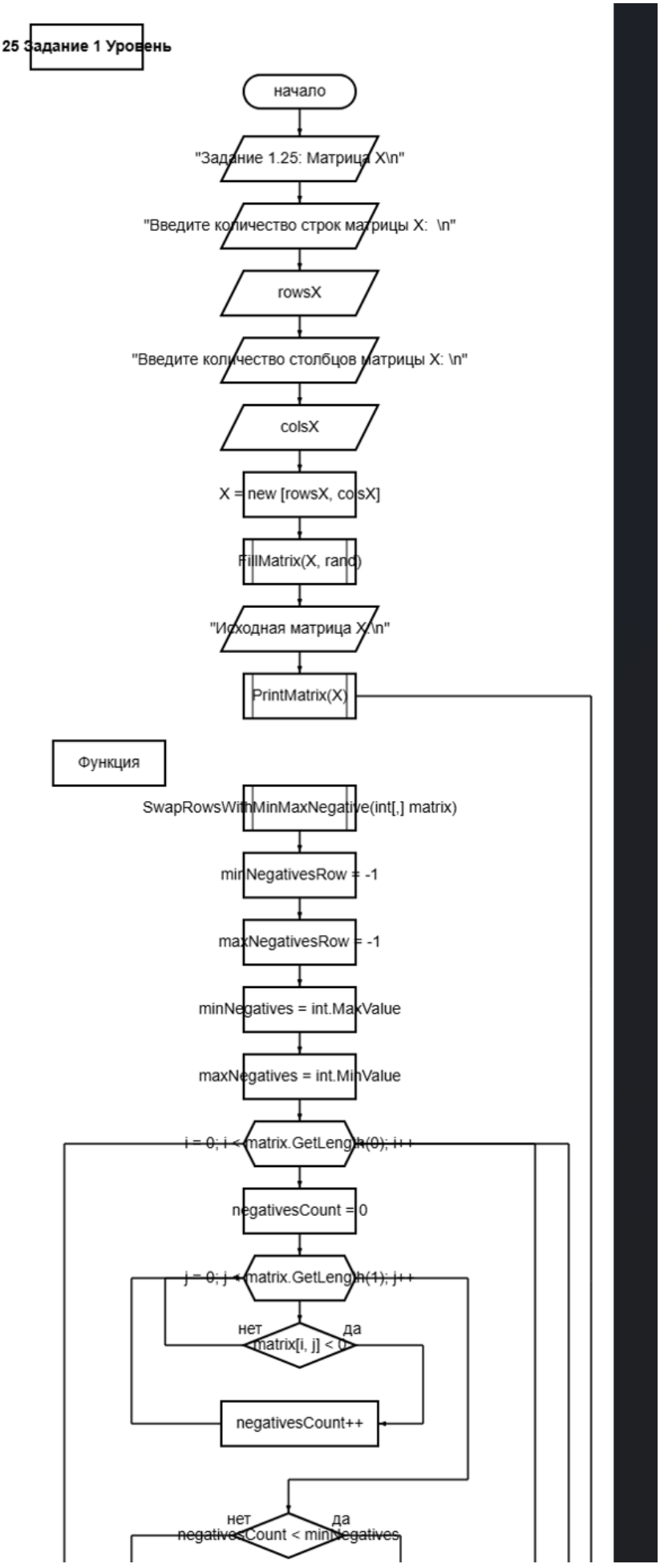

C:\Windows\system32\cmc
Введите номер уровня (1,25 , 2,8 ...): 3,12
Задание 3.12: Обнуление элементов ниже главной диагонали (двумерный массив)
Исходная матрица:
-6 -3 -8 -4
-5 6 4 -1
2 3 7 8
-6 7 -9 5
Матрица после обнуления элементов ниже главной диагонали:
-6 -3 -8 -4
0 8 10 2
0 0 3 7
0 0 0 37
Вариант (б): Обнуление элементов ниже главной диагонали в одномерной последовательности
Результат в одномерной последовательности:
-6 -3 -8 -4
0 8 10 2
0 0 3 7
0 0 0 37
Для продолжения нажмите любую клавишу . . .

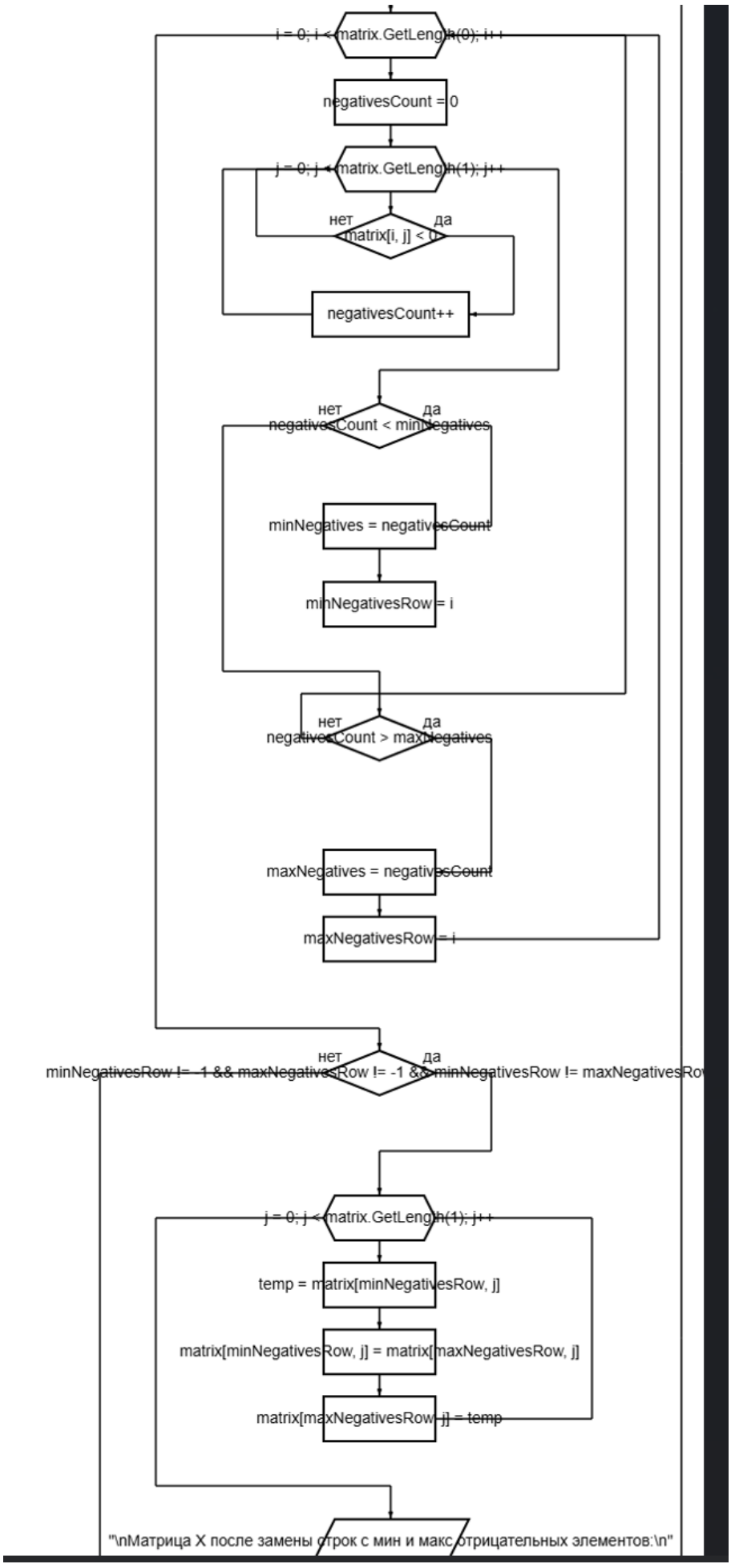


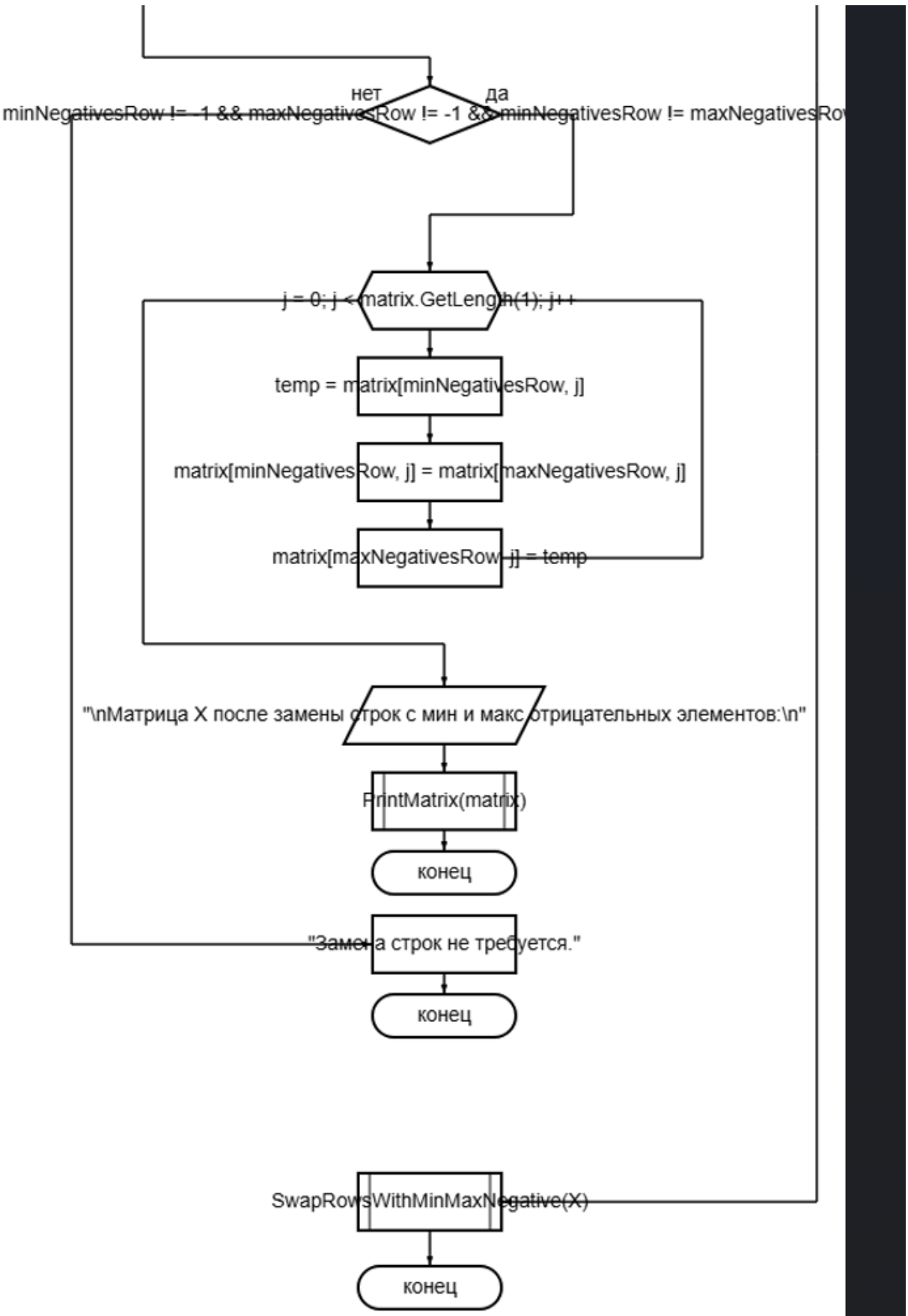
Блок-схемы для каждого задания:

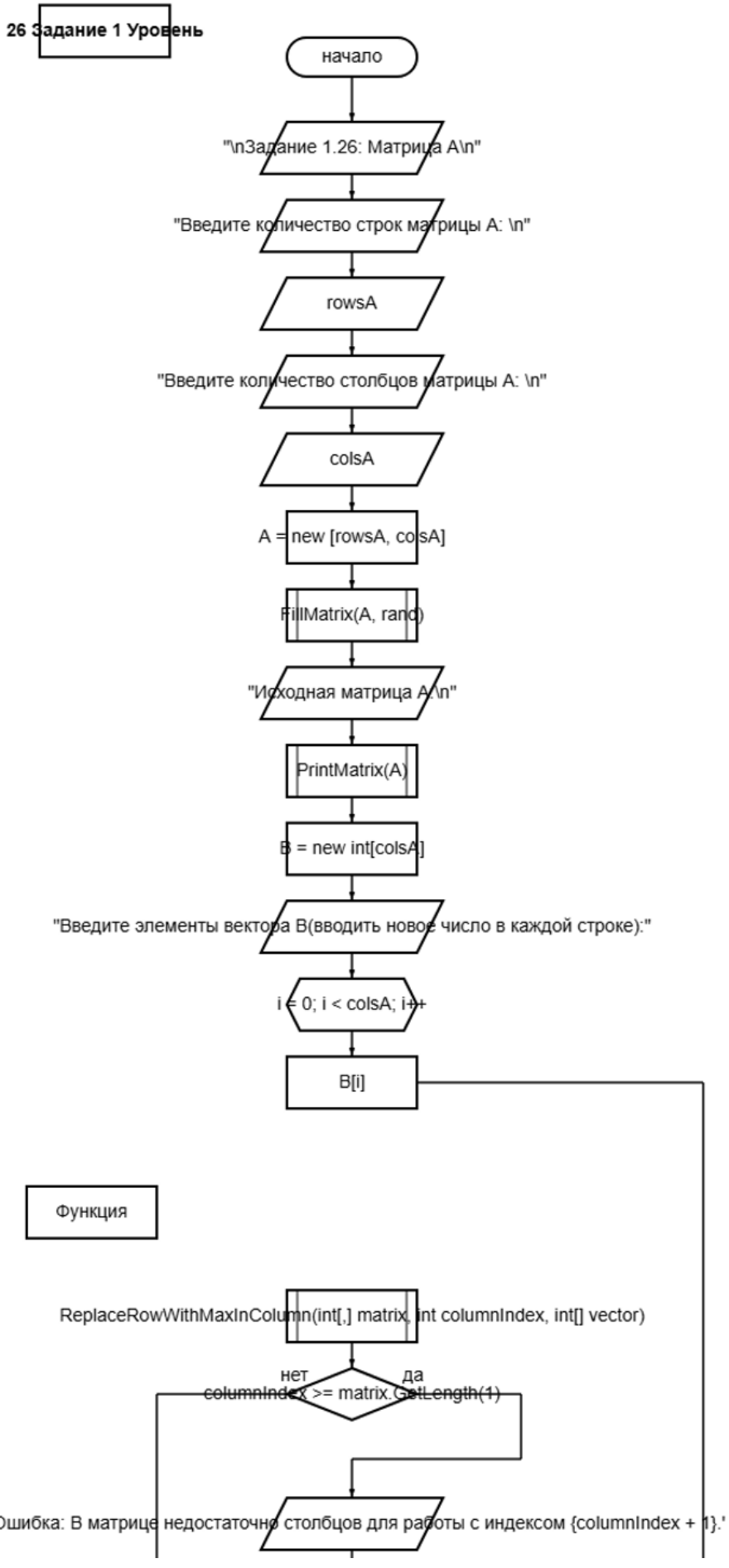

начало
rand = new Random()
"Введите номер уровня (1,25 , 2,8 ...): "
checker
voids for tasks
FillMatrix(int[,] matrix, Random rand)
i = 0; i < matrix.GetLength(0); i++
j = 0; j < matrix.GetLength(1); j++
matrix[i, j] = rand.Next(-10, 10)
конец
PrintMatrix(int[,] matrix)
i = 0; i < matrix.GetLength(0); i++
j = 0; j < matrix.GetLength(1); j++
$"{matrix[i, j],5}"
конец

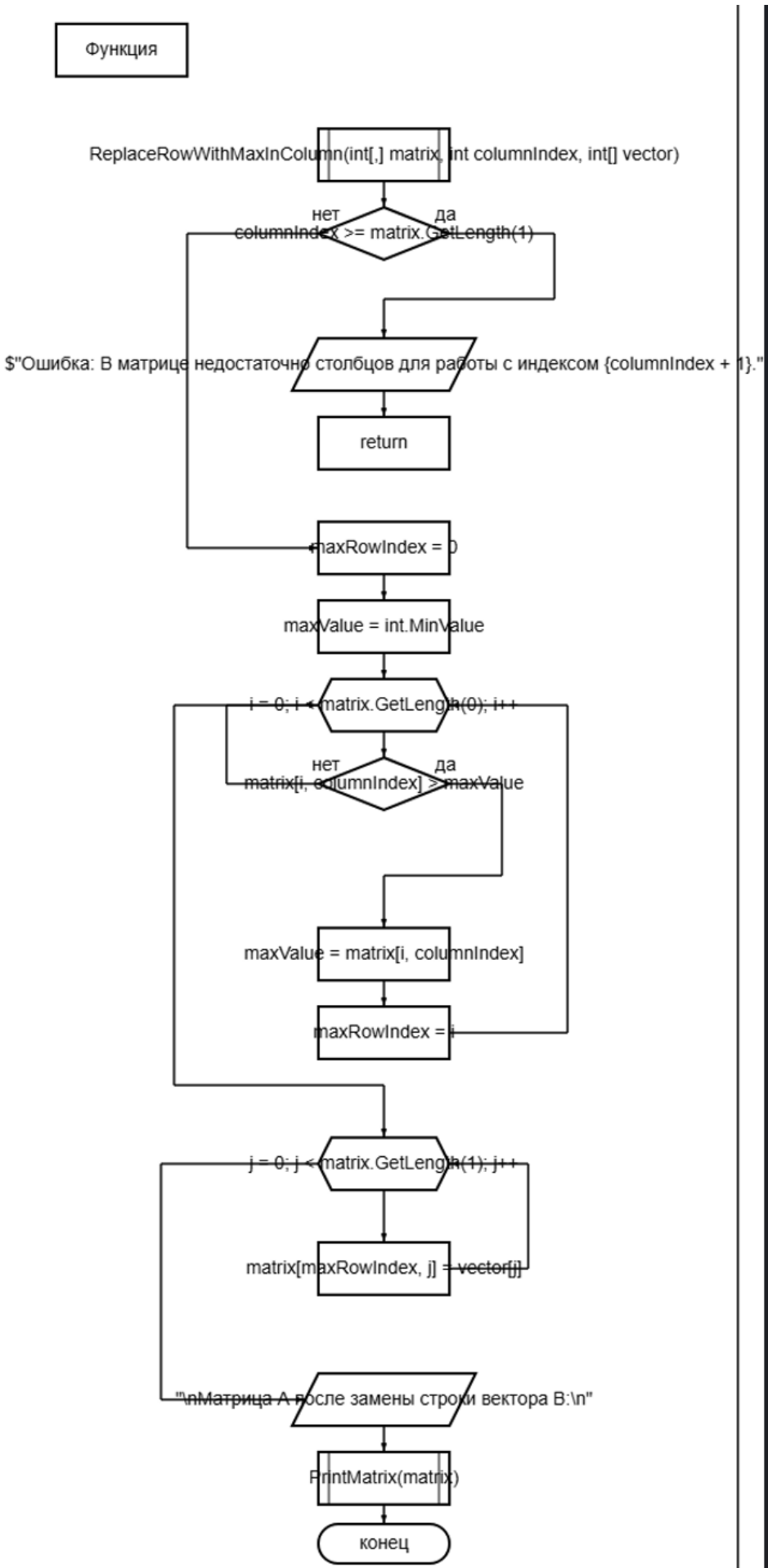
25 Задание 1 Уровень

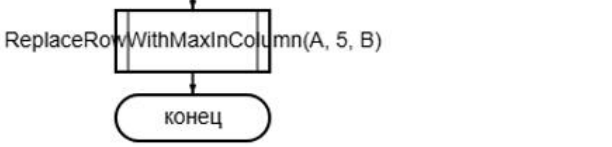



i = 0; i < matrix.GetLength(0); i++
negativesCount = 0
j = 0; j < matrix.GetLength(1); j++
нет
да
matrix[i, j] < 0
negativesCount++
нет
да
negativesCount < minNegatives
minNegatives = negativesCount
minNegativesRow = i
нет
да
negativesCount > maxNegatives
maxNegatives = negativesCount
maxNegativesRow = i
нет
да
minNegativesRow != -1 && maxNegativesRow != -1 && minNegativesRow != maxNegativesRow
j = 0; j < matrix.GetLength(1); j++
temp = matrix[minNegativesRow, j]
matrix[minNegativesRow, j] = matrix[maxNegativesRow, j]
matrix[maxNegativesRow, j] = temp
"\nМатрица X после замены строк с мин и макс отрицательных элементов:\n"

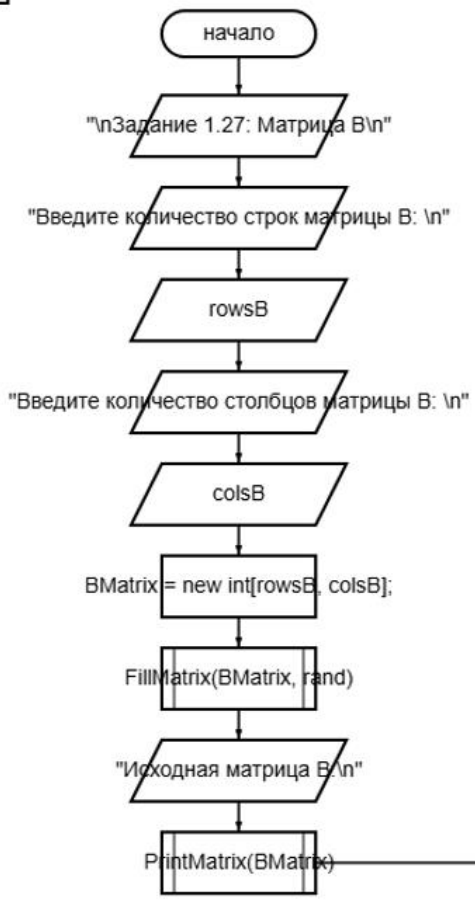

нет
да
minNegativesRow != -1 && maxNegativesRow != -1 && minNegativesRow != maxNegativesRow
j = 0; j < matrix.GetLength(1); j++
temp = matrix[minNegativesRow, j]
matrix[minNegativesRow, j] = matrix[maxNegativesRow, j]
matrix[maxNegativesRow, j] = temp
"\nМатрица X после замены строк с мин и макс отрицательных элементов:\n"
PrintMatrix(matrix)
конец
"Замена строк не требуется."
конец
SwapRowsWithMinMaxNegative(X)
конец


26 Задание 1 Уровень
начало
"\nЗадание 1.26: Матрица A\n"
"Введите количество строк матрицы A: \n"
rowsA
"Введите количество столбцов матрицы A: \n"
colsA
A = new [rowsA, colsA]
FillMatrix(A, rand)
"Исходная матрица A:\n"
PrintMatrix(A)
B = new int[colsA]
"Введите элементы вектора B(вводить новое число в каждой строке):"
i = 0; i < colsA; i++
B[i]
Функция
ReplaceRowWithMaxInColumn(int[,] matrix, int columnIndex, int[] vector)
нет
да
columnIndex >= matrix.GetLength(1)
$"Ошибка: В матрице недостаточно столбцов для работы с индексом {columnIndex + 1}."

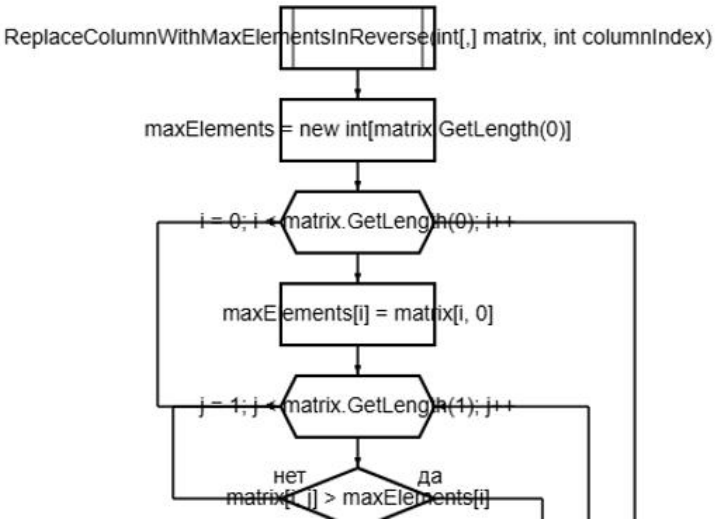

Функция
ReplaceRowWithMaxInColumn(int[,] matrix, int columnIndex, int[] vector)
нет
да
columnIndex >= matrix.GetLength(1)
$"Ошибка: В матрице недостаточно столбцов для работы с индексом {columnIndex + 1}."
return
maxRowIndex = 0
maxValue = int.MinValue
i = 0; i < matrix.GetLength(0); i++
нет
да
matrix[i, columnIndex] > maxValue
maxValue = matrix[i, columnIndex]
maxRowIndex = i
j = 0; j < matrix.GetLength(1); j++
matrix[maxRowIndex, j] = vector[j]
"\nМатрица A после замены строки вектора B:\n"
PrintMatrix(matrix)
конец

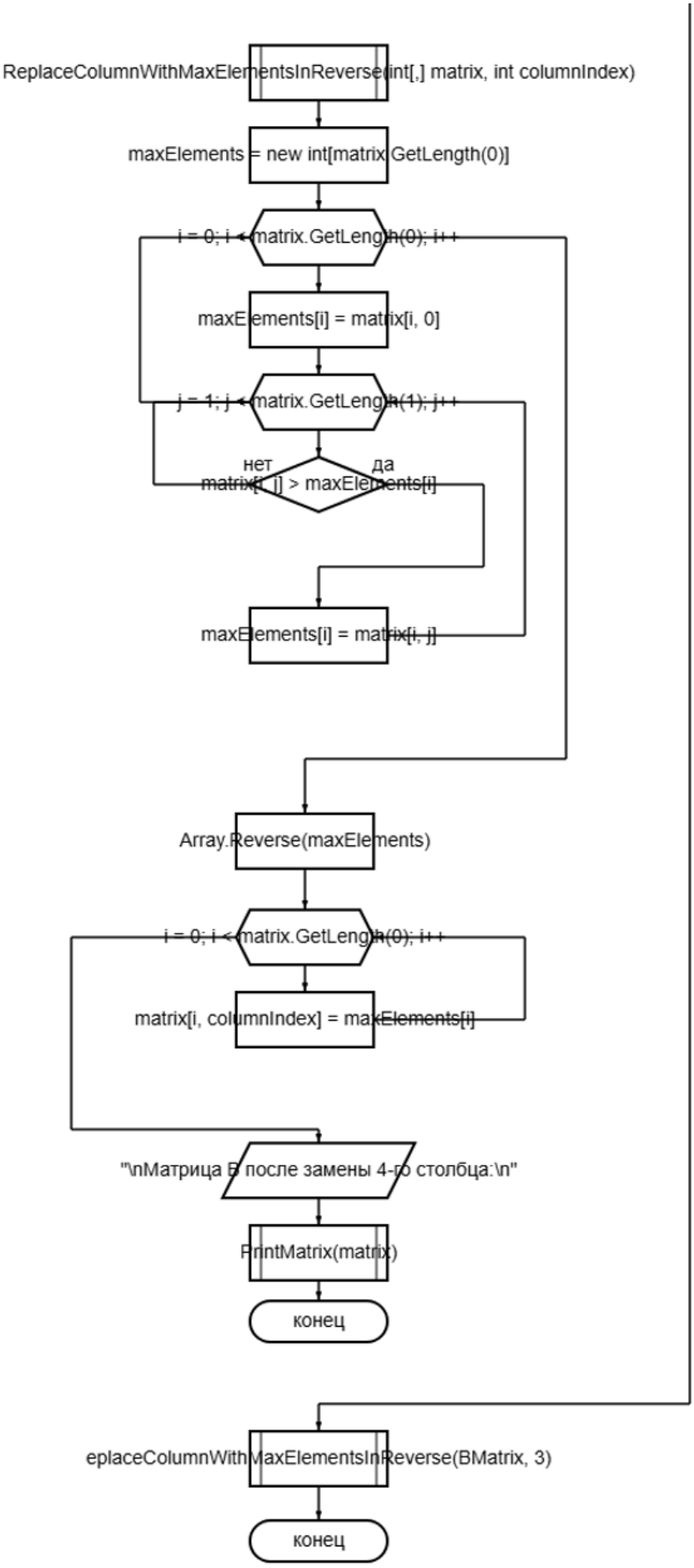


27 Задание 1 Уровень

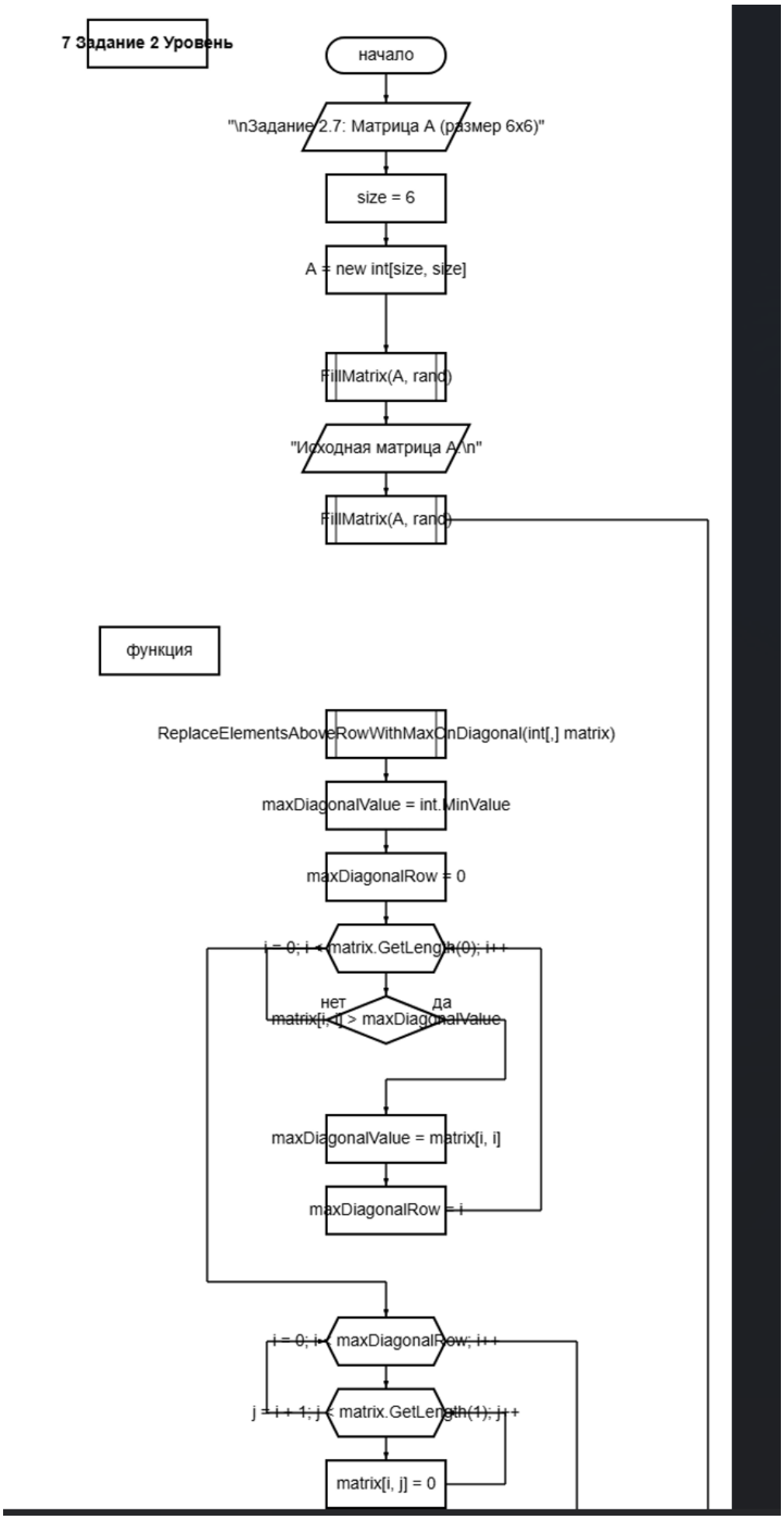


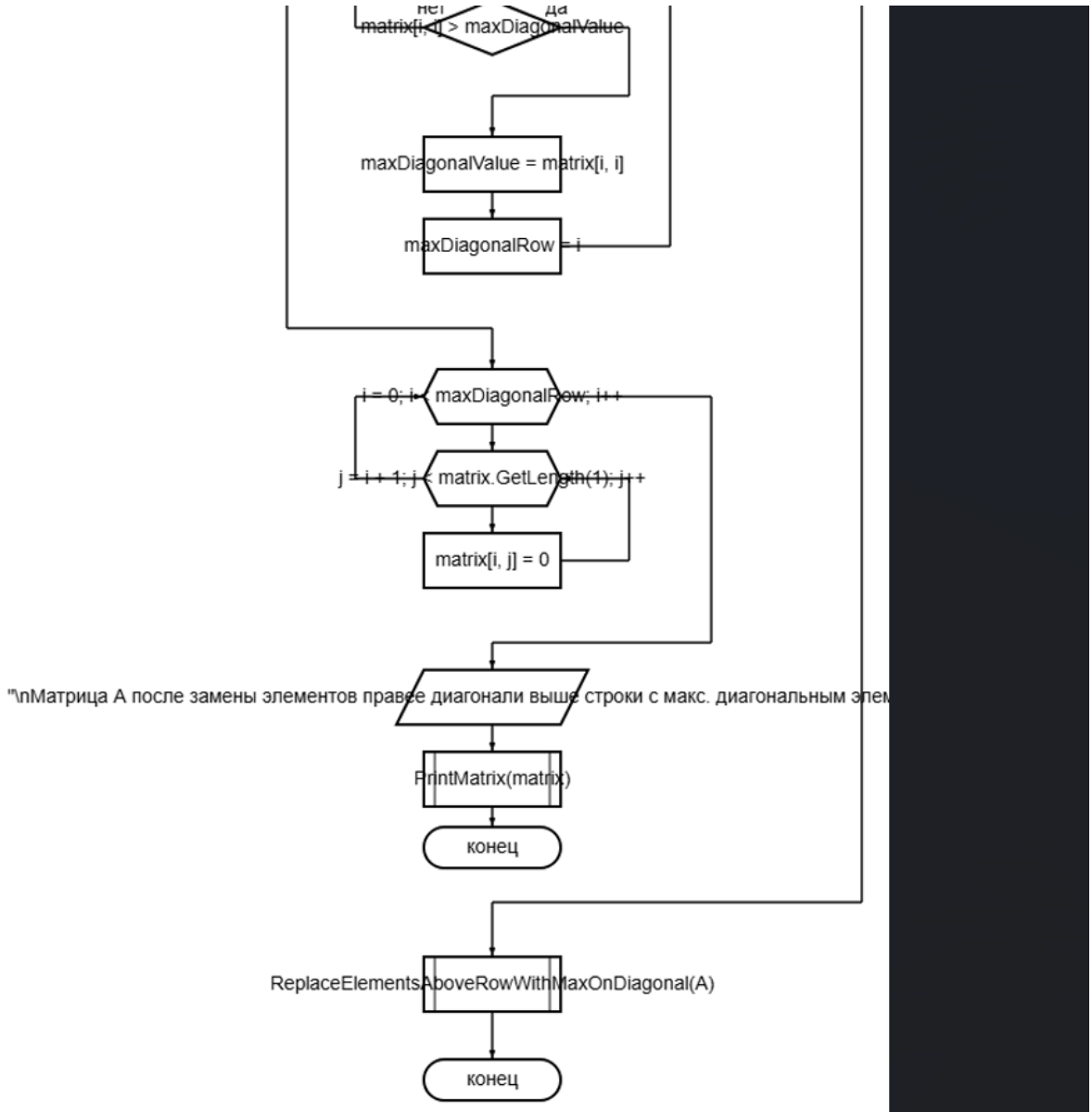
функция



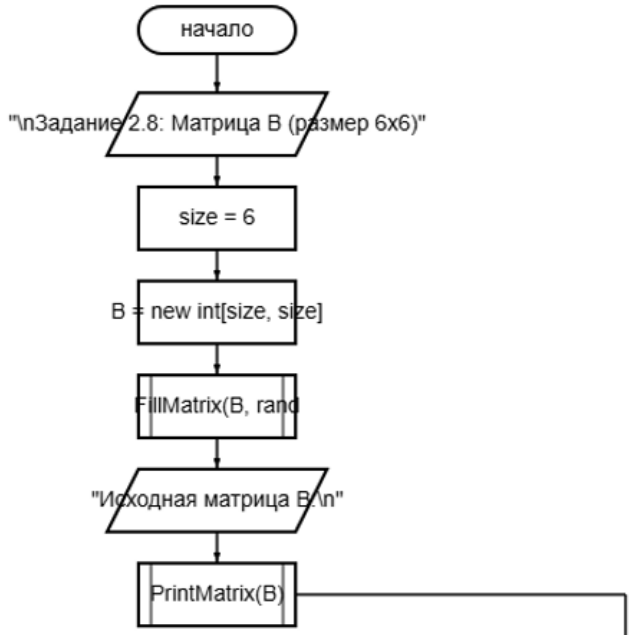

ReplaceColumnWithMaxElementsInReverse(int[,] matrix, int columnIndex)
maxElements = new int[matrix.GetLength(0)]
i = 0; i < matrix.GetLength(0); i++
maxElements[i] = matrix[i, 0]
j = 1; j < matrix.GetLength(1); j++
нет
да
matrix[i, j] > maxElements[i]
maxElements[i] = matrix[i, j]
Array.Reverse(maxElements)
i = 0; i < matrix.GetLength(0); i++
matrix[i, columnIndex] = maxElements[i]
"\nМатрица B после замены 4-го столбца:\n"
PrintMatrix(matrix)
конец
eplaceColumnWithMaxElementsInReverse(BMatrix, 3)
конец

7 Задание 2 Уровень

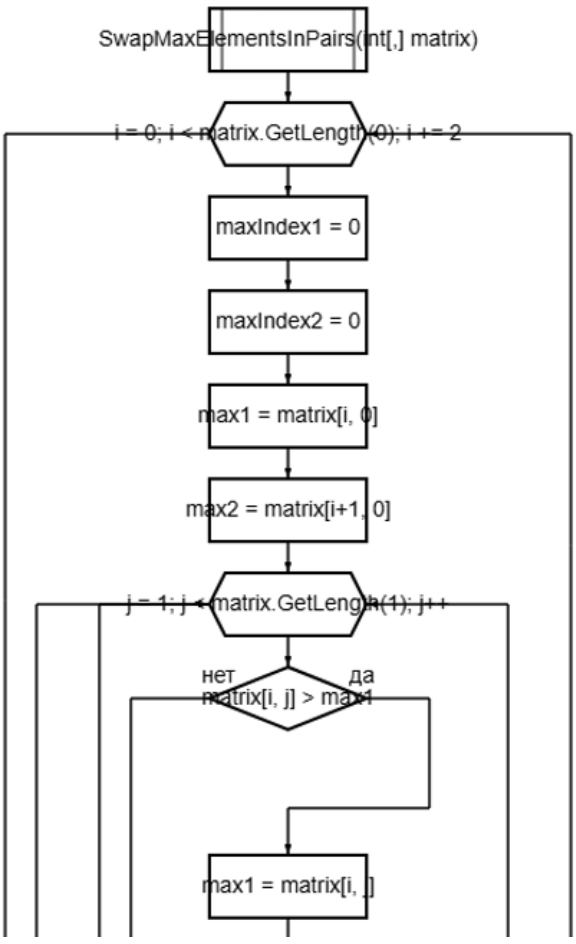


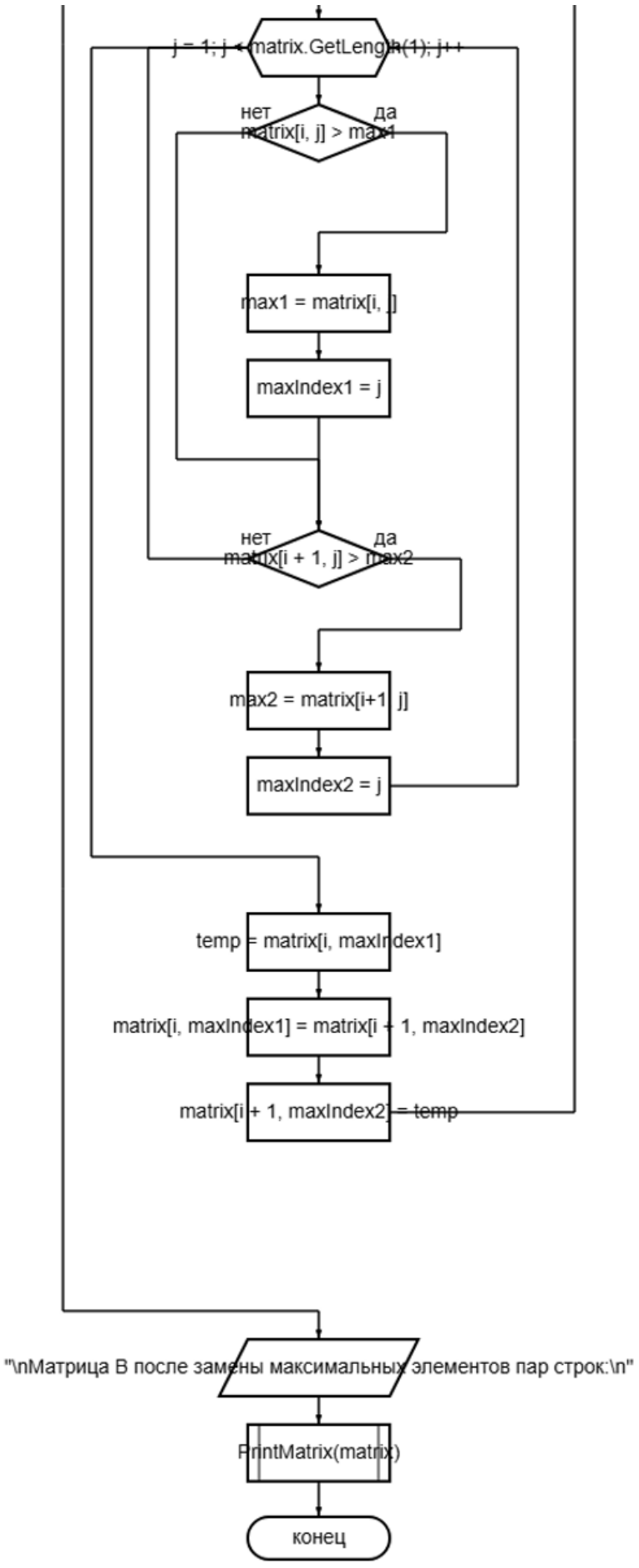

нет
да
matrix[i, i] > maxDiagonalValue
maxDiagonalValue = matrix[i, i]
maxDiagonalRow = i
i = 0; i < maxDiagonalRow; i++
j = i + 1; j < matrix.GetLength(1); j++
matrix[i, j] = 0
"\nМатрица A после замены элементов правее диагонали выше строки с макс. диагональным элем
PrintMatrix(matrix)
конец
ReplaceElementsAboveRowWithMaxOnDiagonal(A)
конец

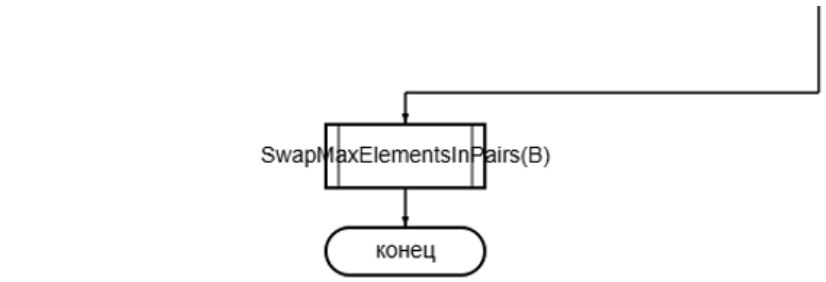
8 Задание 2 Уровень



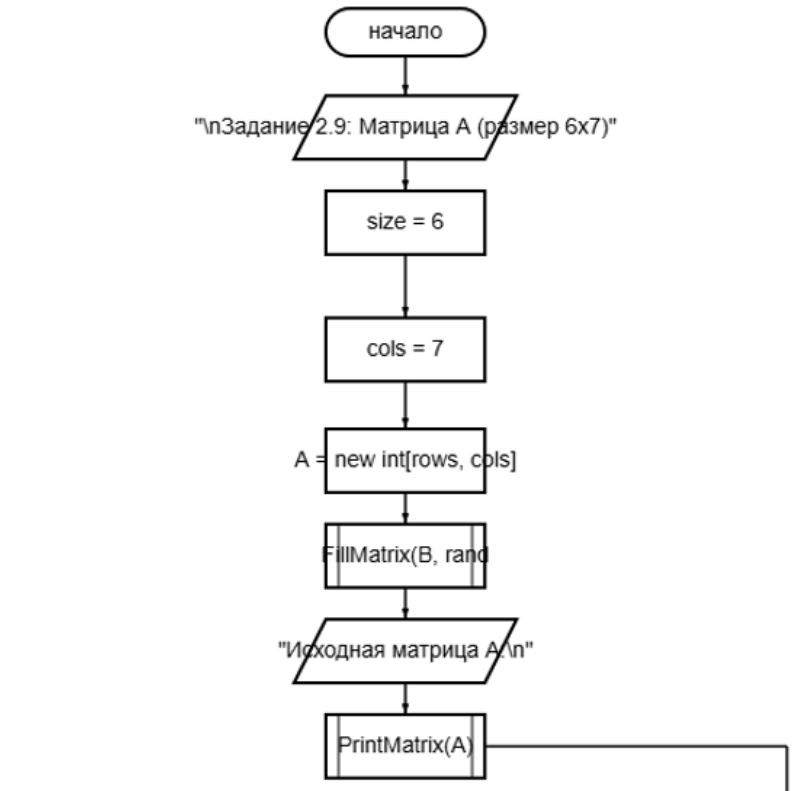
функция



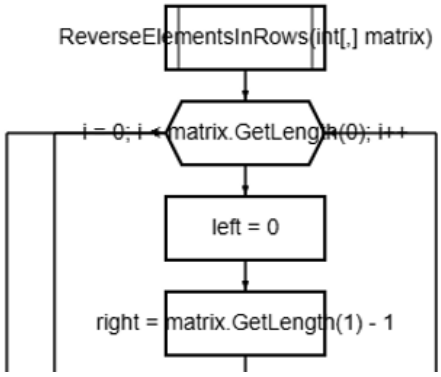

j = 1; j < matrix.GetLength(1); j++
нет
да
matrix[i, j] > max1
max1 = matrix[i, j]
maxIndex1 = j
нет
да
matrix[i + 1, j] > max2
max2 = matrix[i+1, j]
maxIndex2 = j
temp = matrix[i, maxIndex1]
matrix[i, maxIndex1] = matrix[i + 1, maxIndex2]
matrix[i + 1, maxIndex2] = temp
"\nМатрица B после замены максимальных элементов пар строк:\n"
PrintMatrix(matrix)
конец

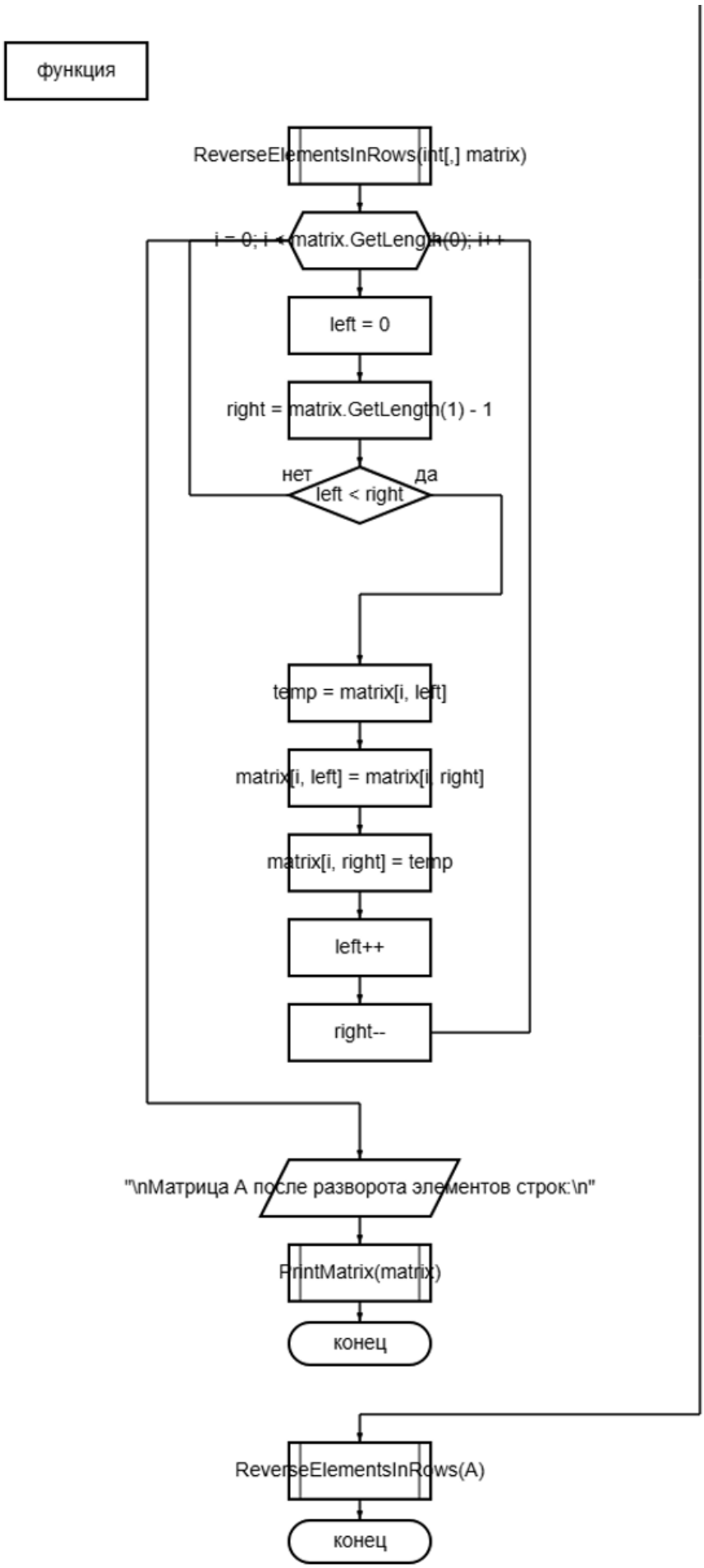


9 Задание 2 Уровень

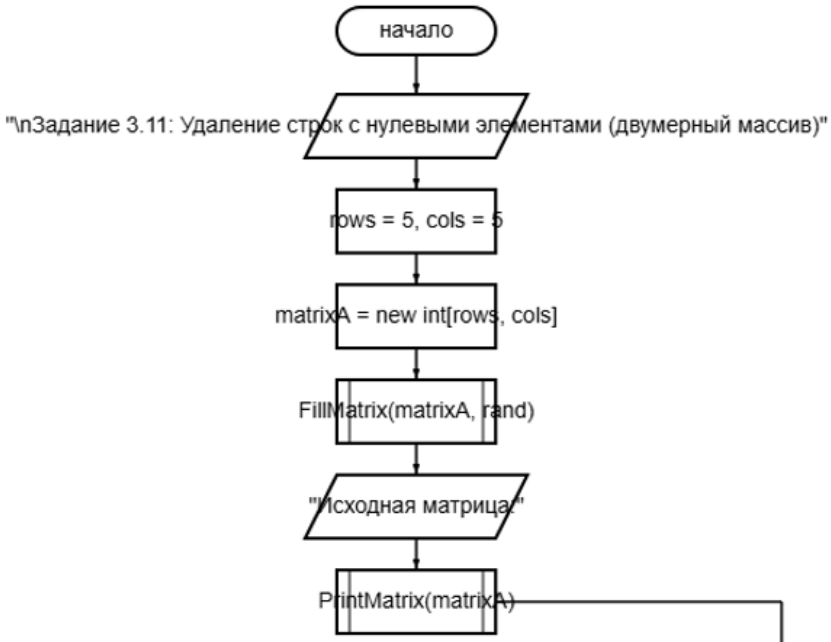


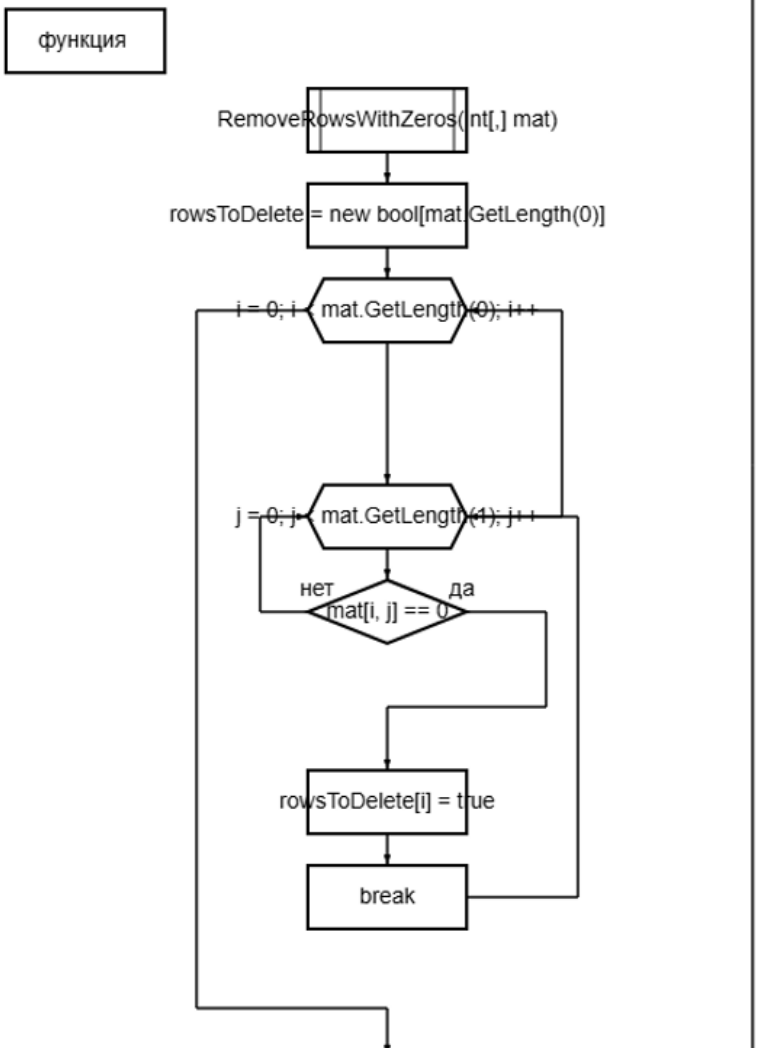
функция

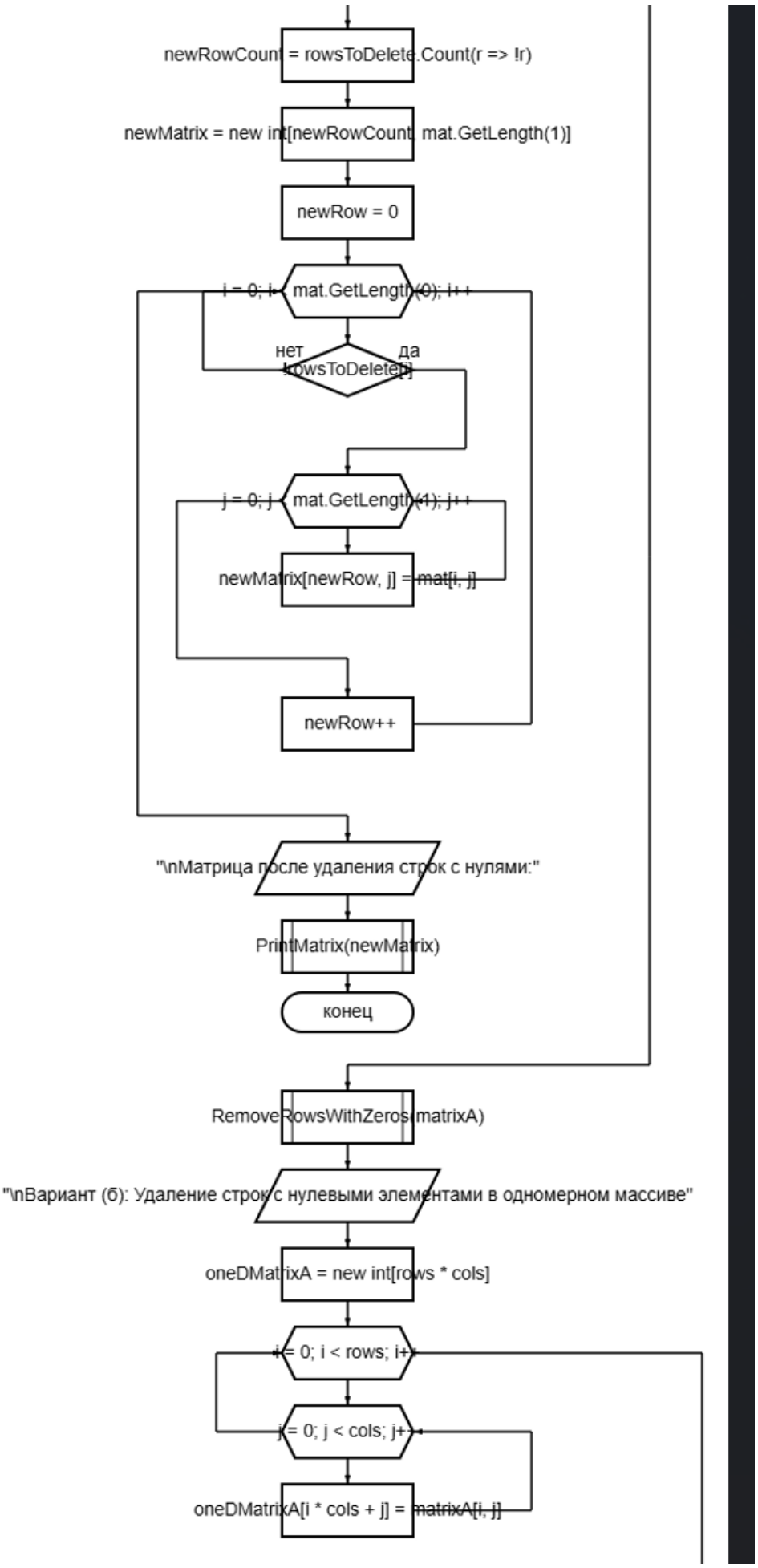


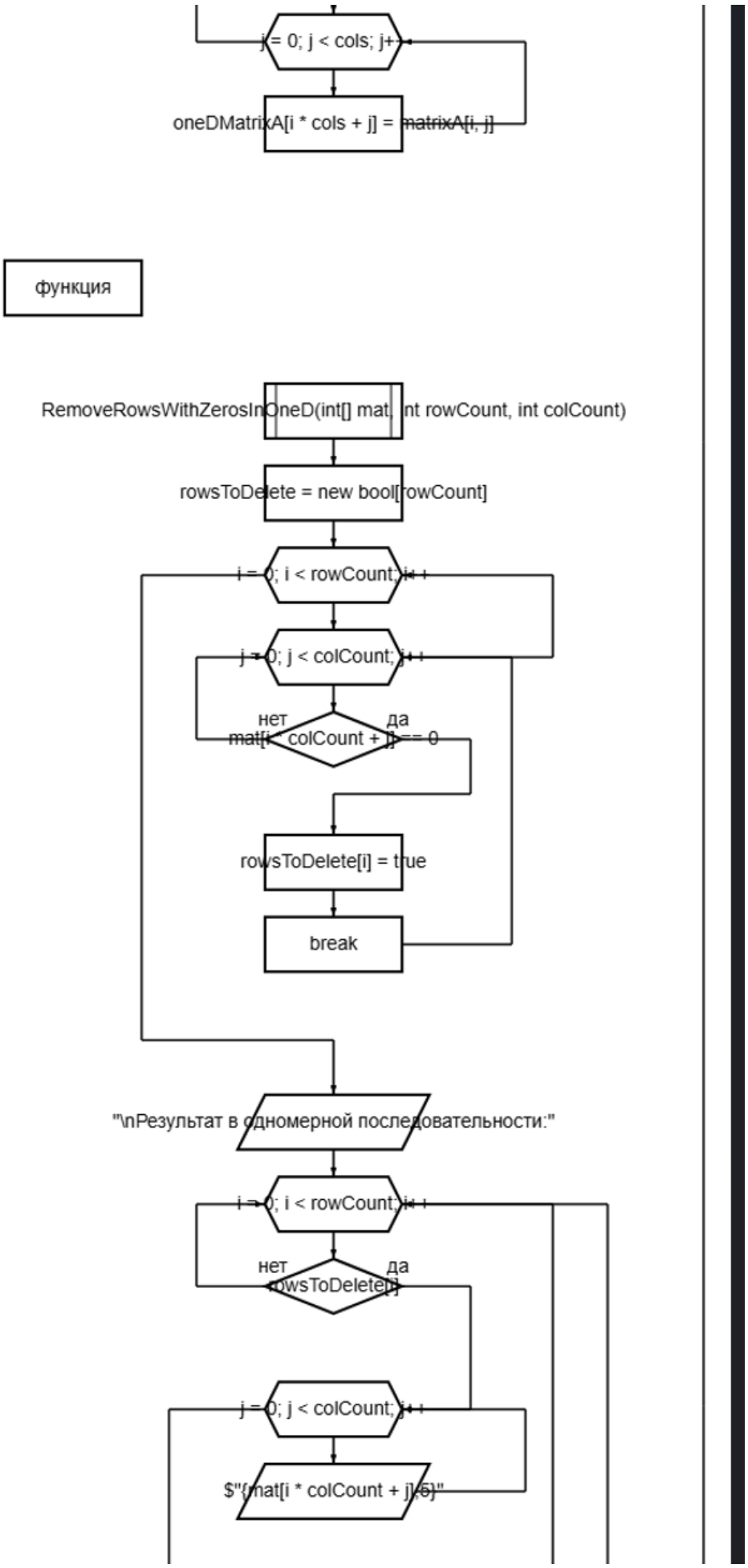

функция
ReverseElementsInRows(int[,] matrix)
i = 0; i < matrix.GetLength(0); i++
left = 0
right = matrix.GetLength(1) - 1
нет
да
left < right
temp = matrix[i, left]
matrix[i, left] = matrix[i, right]
matrix[i, right] = temp
left++
right--
"\nМатрица A после разворота элементов строк:\n"
PrintMatrix(matrix)
конец
ReverseElementsInRows(A)
конец

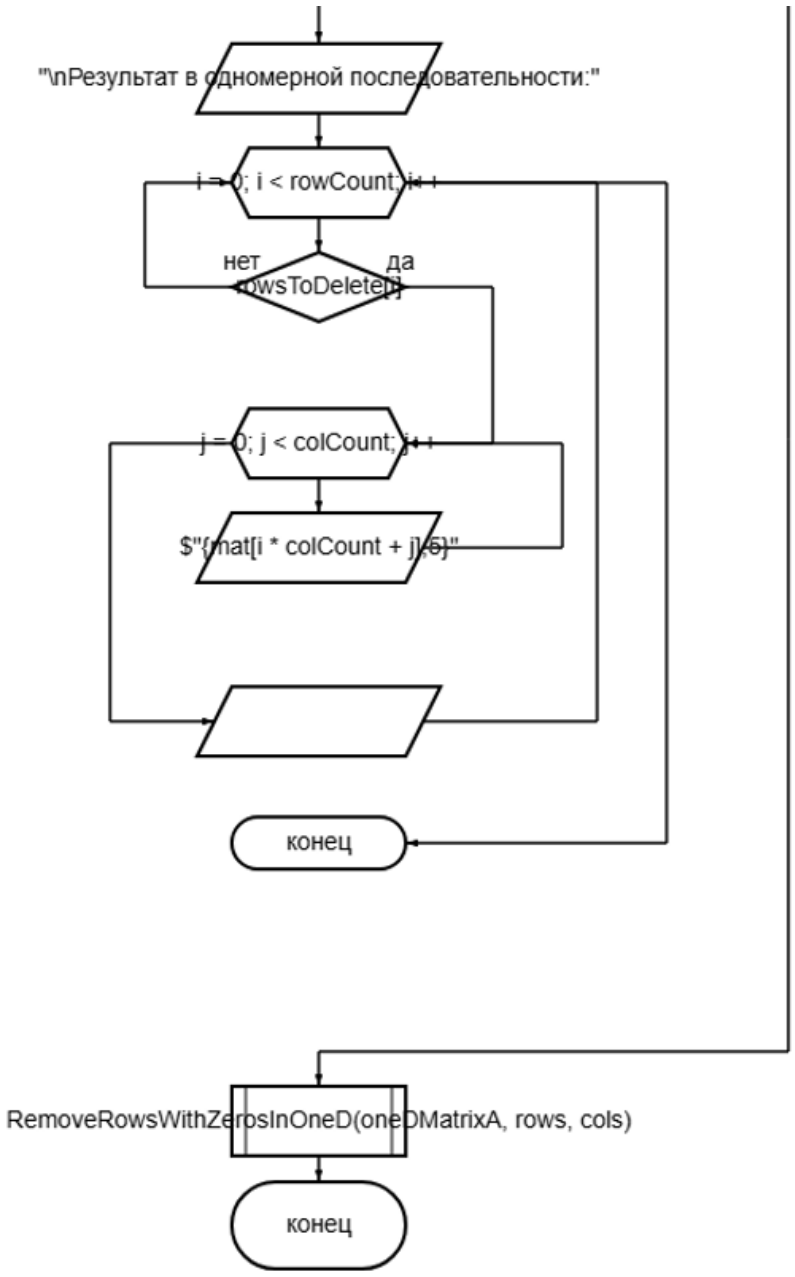
11 Задание 3 Уровень



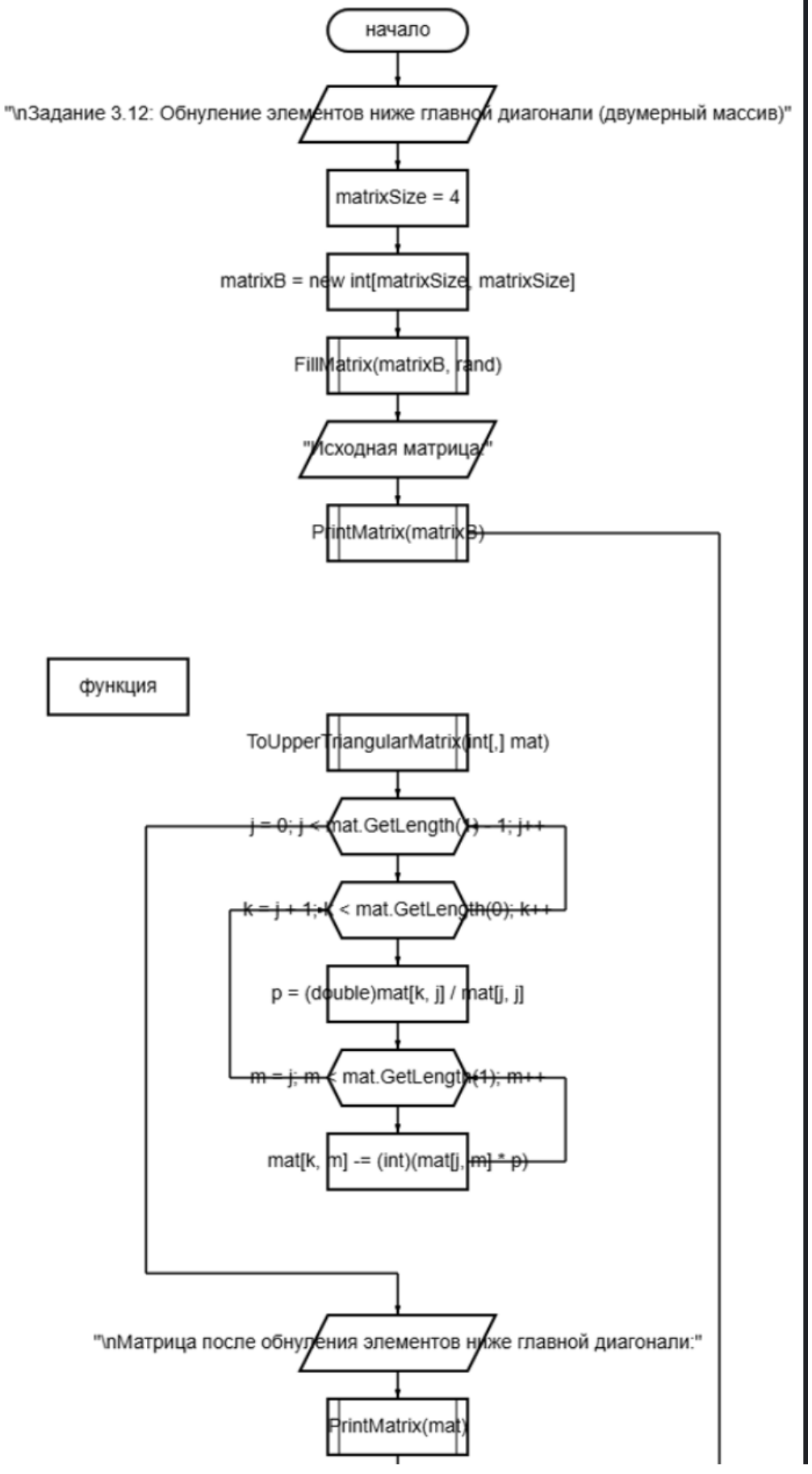


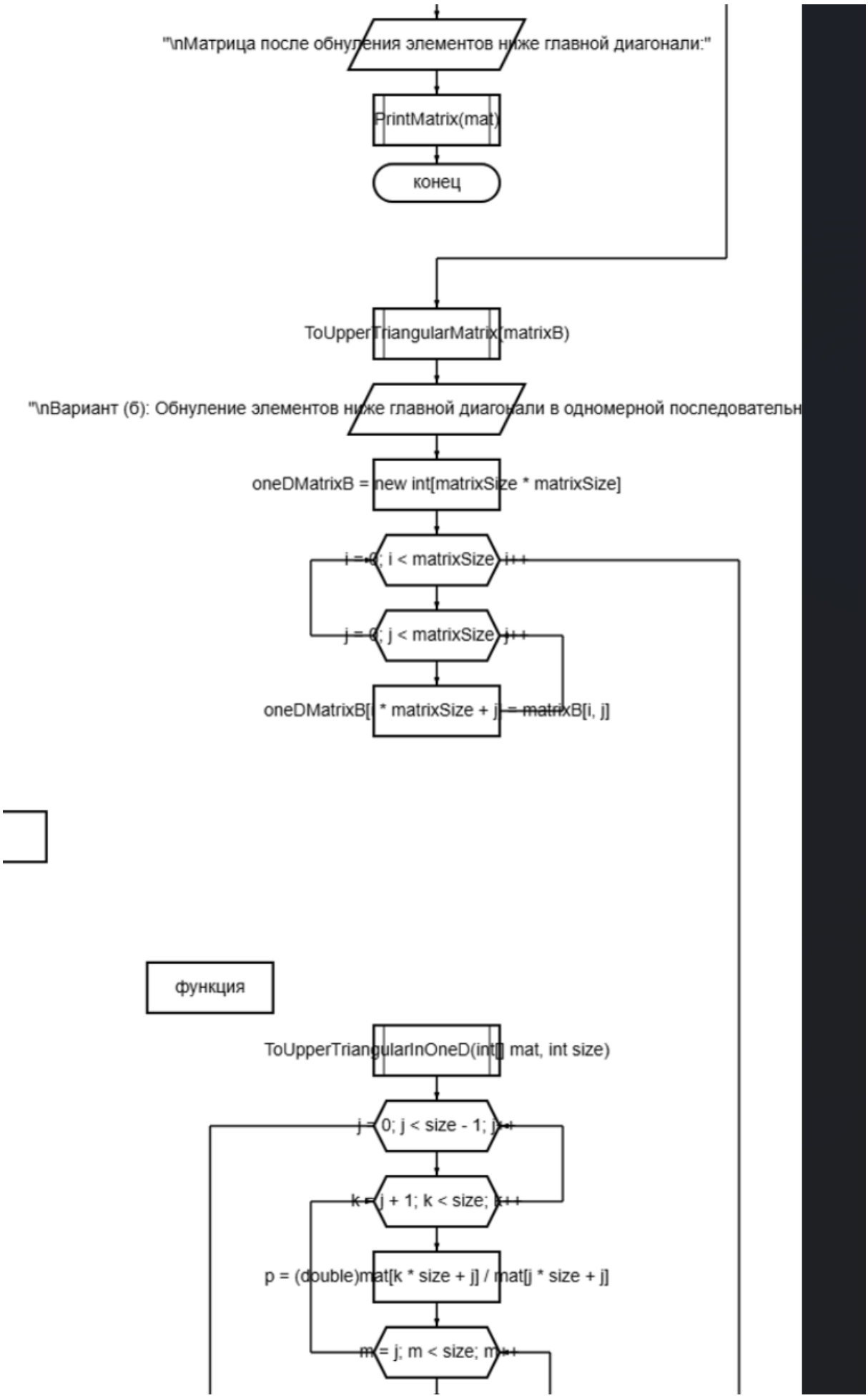

newRowCount = rowsToDelete.Count(r => !r)
newMatrix = new int[newRowCount, mat.GetLength(1)]
newRow = 0
i = 0; i < mat.GetLength(0); i++
нет
да
!rowsToDelete[i]
j = 0; j < mat.GetLength(1); j++
newMatrix[newRow, j] = mat[i, j]
newRow++
"\nМатрица после удаления строк с нулями:"
PrintMatrix(newMatrix)
конец
RemoveRowsWithZeros(matrixA)
"\nВариант (б): Удаление строк с нулевыми элементами в одномерном массиве"
oneDMatrixA = new int[rows * cols]
i = 0; i < rows; i++
j = 0; j < cols; j++
oneDMatrixA[i * cols + j] = matrixA[i, j]

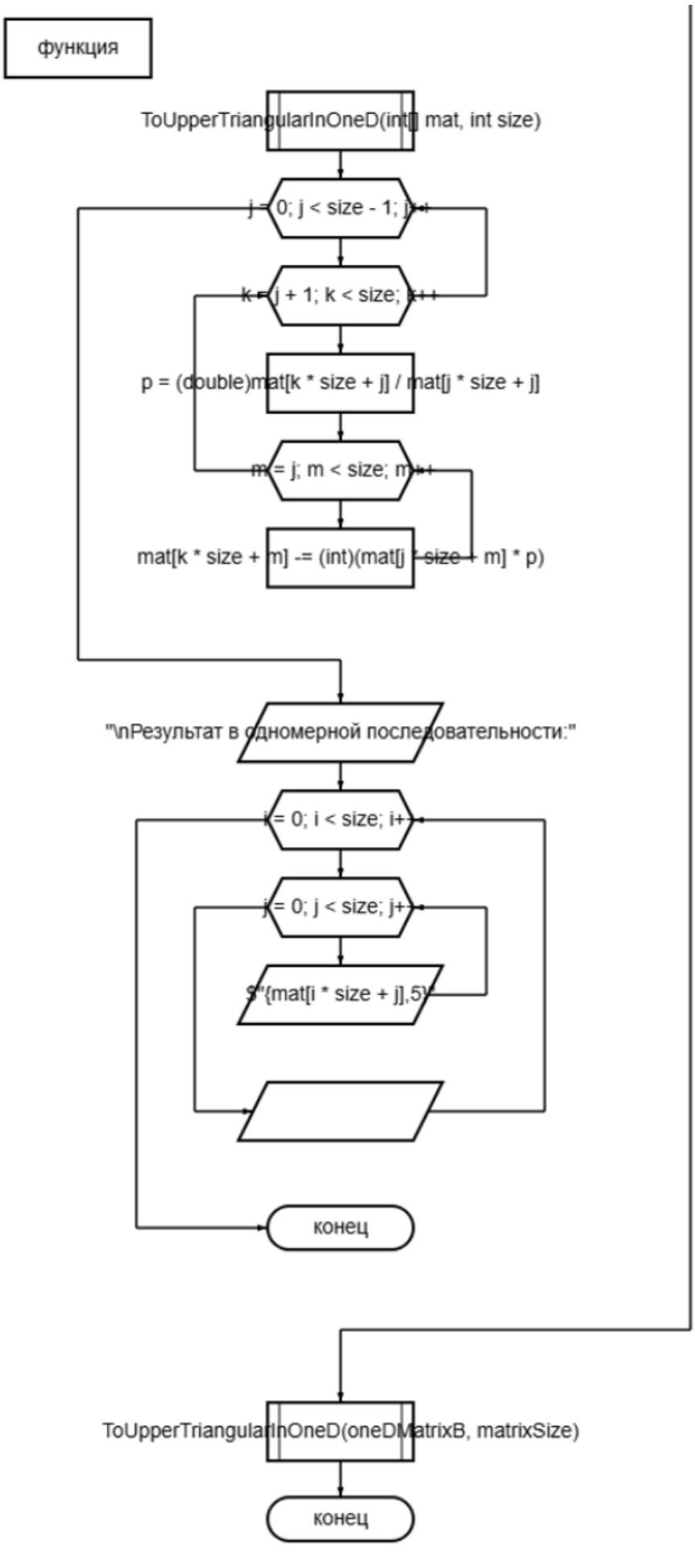

j = 0; j < cols; j++
oneDMatrixA[i * cols + j] = matrixA[i, j]
функция
RemoveRowsWithZerosInOneD(int[] mat, int rowCount, int colCount)
rowsToDelete = new bool[rowCount]
i = 0; i < rowCount; i++
j = 0; j < colCount; j++
нет
да
mat[i * colCount + j] == 0
rowsToDelete[i] = true
break
"\nРезультат в одномерной последовательности:"
i = 0; i < rowCount; i++
нет
да
rowsToDelete[i]
j = 0; j < colCount; j++
$"{mat[i * colCount + j],6}"


"\nРезультат в одномерной последовательности:"
i = 0; i < rowCount; i++
нет
да
rowsToDelete[i]
j = 0; j < colCount; j++
$"{mat[i * colCount + j],6}"
конец
RemoveRowsWithZerosInOneD(oneDMatrixA, rows, cols)
конец

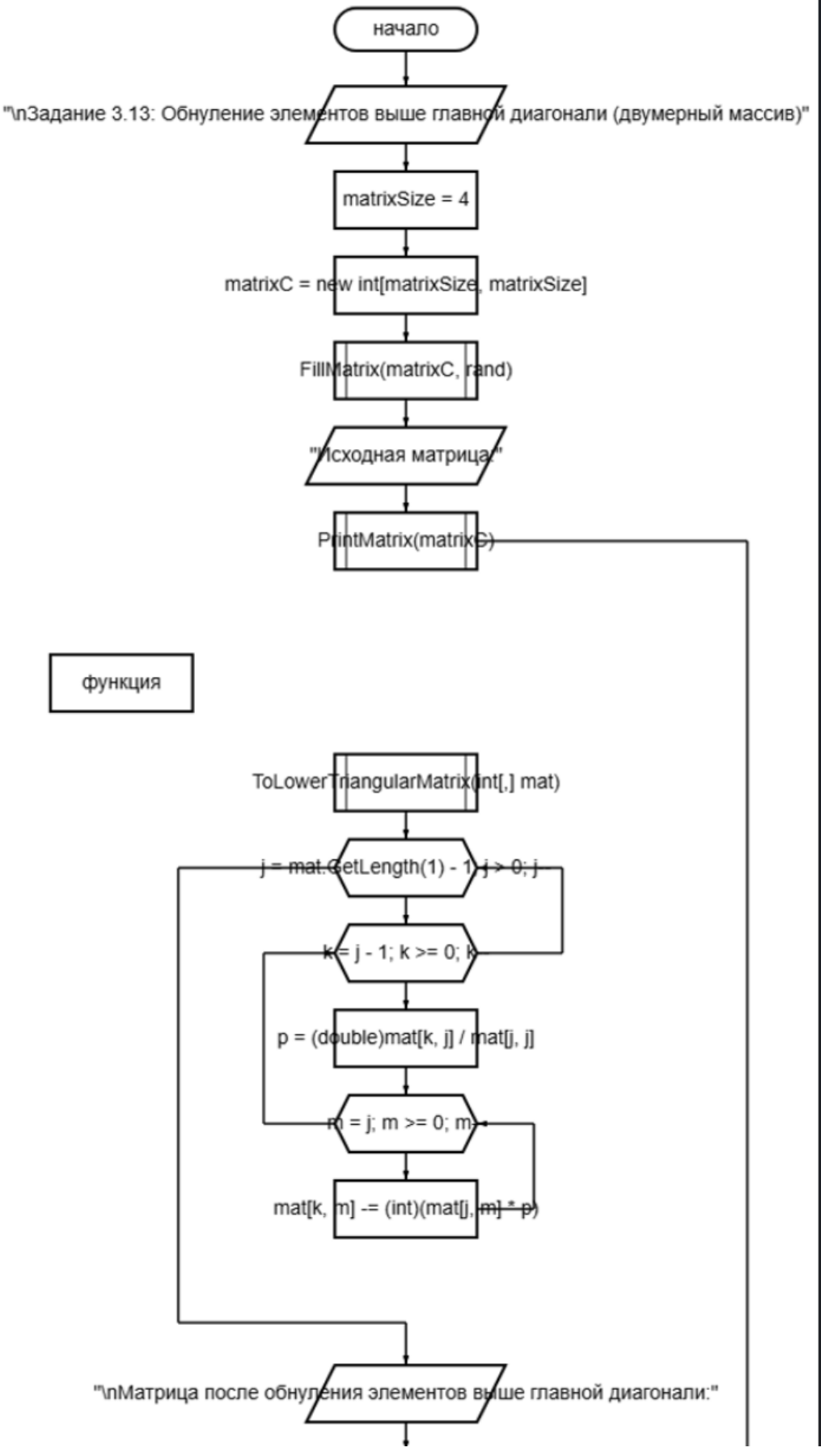
12 Задание 3 Уровень

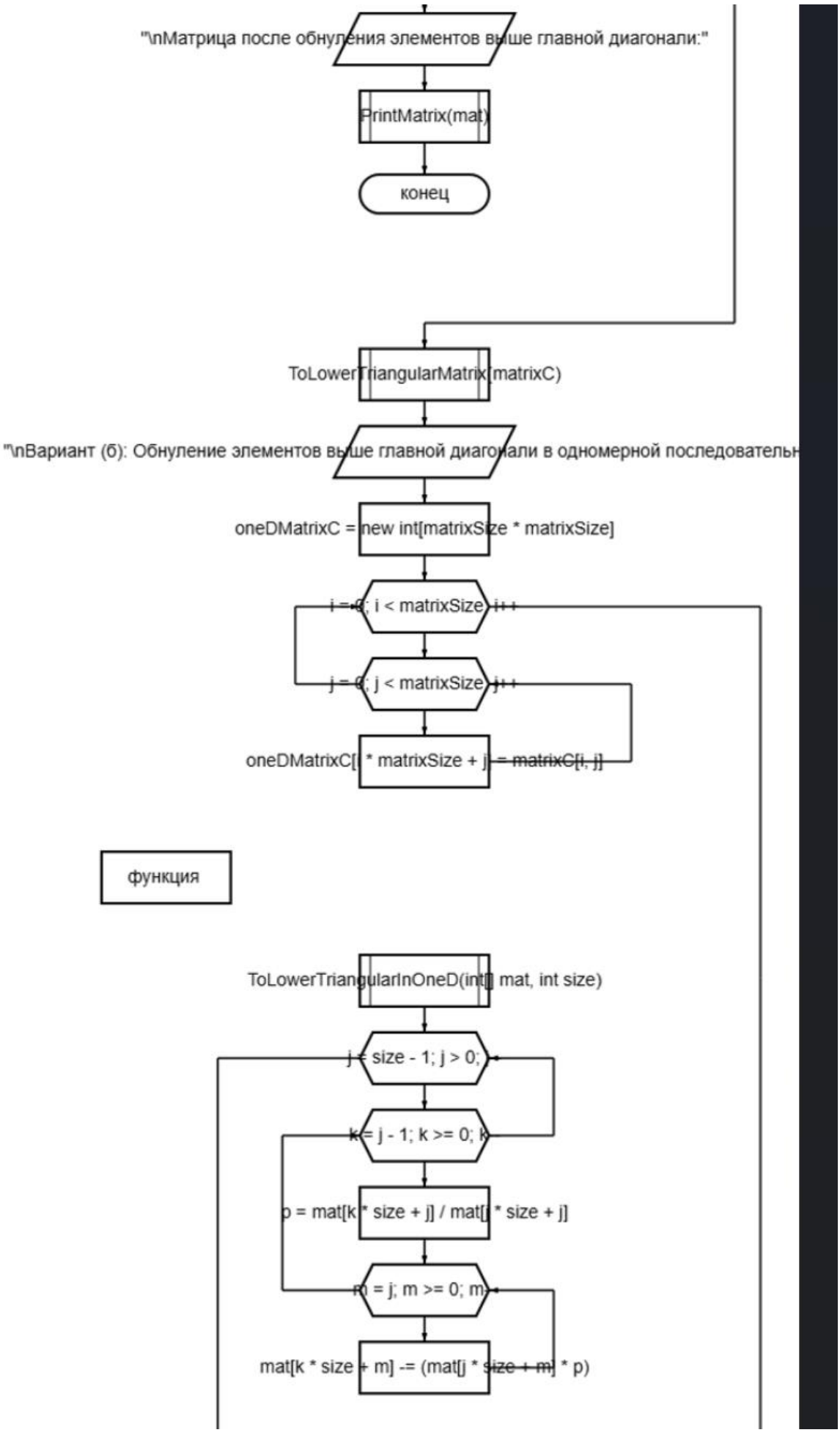


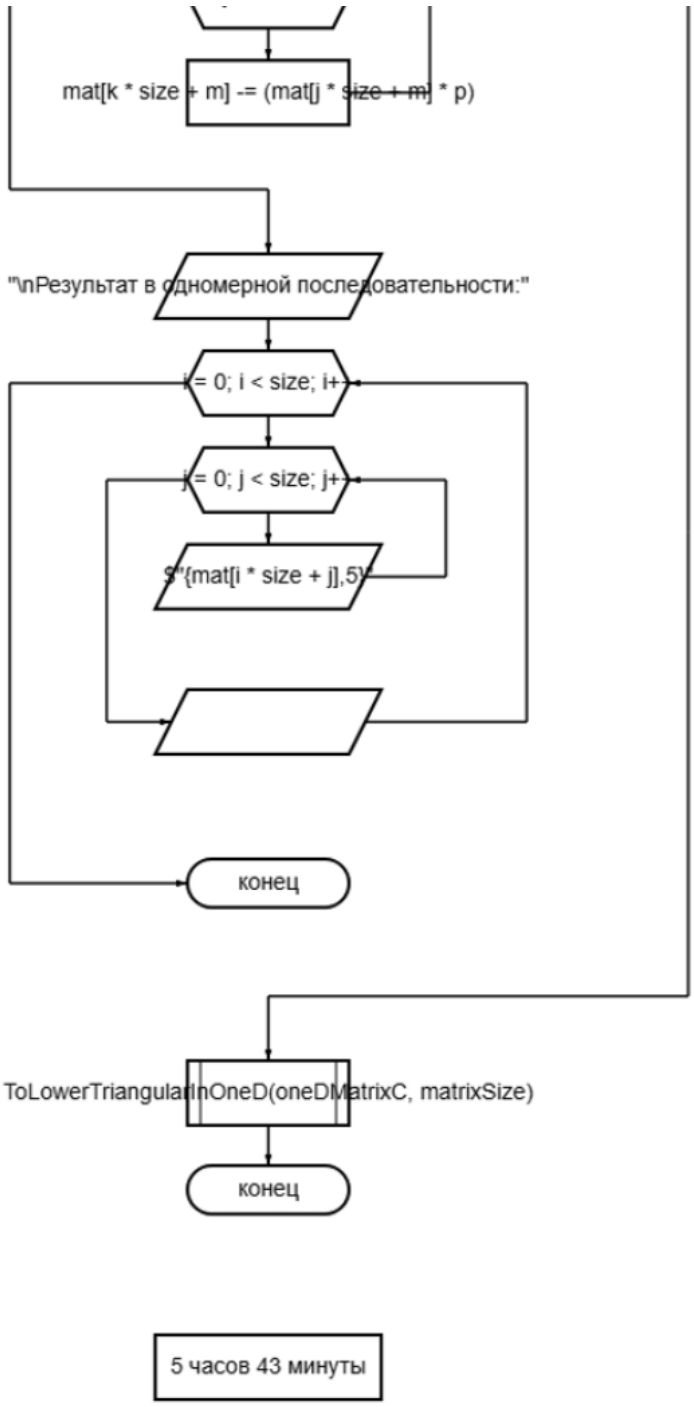

"\nМатрица после обнуления элементов ниже главной диагонали:"
PrintMatrix(mat)
конец
ToUpperTriangularMatrix(matrixB)
"\nВариант (б): Обнуление элементов ниже главной диагонали в одномерной последовательн
oneDMatrixB = new int[matrixSize * matrixSize]
i = 0; i < matrixSize; i++
j = 0; j < matrixSize; j++
oneDMatrixB[* matrixSize + j] = matrixB[i, j]
функция
ToUpperTriangularInOneD(int[] mat, int size)
j = 0; j < size - 1; j++
k = j + 1; k < size; k++
p = (double)mat[k * size + j] / mat[j * size + j]
m = j; m < size; m++


функция
ToUpperTriangularInOneD(int[] mat, int size)
j = 0; j < size - 1; j++
k = j + 1; k < size; k++
p = (double)mat[k * size + j] / mat[j * size + j]
m = j; m < size; m++
mat[k * size + m] -= (int)(mat[j * size + m] * p)
"\nРезультат в одномерной последовательности:"
i = 0; i < size; i++
j = 0; j < size; j++
$"{mat[i * size + j],5}"
конец
ToUpperTriangularInOneD(oneDMatrixB, matrixSize)
конец

13 Задание 3 Уровень




"\nМатрица после обнуления элементов выше главной диагонали:"
PrintMatrix(mat)
конец
ToLowerTriangularMatrix(matrixC)
"\nВариант (б): Обнуление элементов выше главной диагонали в одномерной последовательн
oneDMatrixC = new int[matrixSize * matrixSize]
i = 0; i < matrixSize; i++
j = 0; j < matrixSize; j++
oneDMatrixC[* matrixSize + j] = matrixC[i, j]
функция
ToLowerTriangularInOneD(int[] mat, int size)
j = size - 1; j > 0;
k = j - 1; k >= 0; k
p = mat[k * size + j] / mat[j * size + j]
m = j; m >= 0; m
mat[k * size + m] -= (mat[j * size + m] * p)



Вывод: в ходе выполнения лабораторной работы были приобретены практические навыки работы с массивами, включая их создание, обработку и вывод результатов в удобной форме. Программа успешно реализует типовые алгоритмы обработки данных в матрицах, позволяя проводить сложные преобразования, такие как обнуление элементов по диагоналям и удаление строк с определенными условиями. Кроме того, была реализована возможность представления матриц как в виде двумерного массива, так и в виде одномерной последовательности, что расширяет универсальность и гибкость работы с данными. Работа позволила закрепить понимание

принципов алгоритмической обработки массивов и научила применять творческий подход при решении сложных задач.