Дисциплина «Программирование корпоративных систем»
Рабочая тетрадь 2.2
Массивы

Теоретический материал

Массив представляет набор однотипных данных. Объявление массива похоже на объявление переменной за тем исключением, что после указания типа ставятся квадратные скобки:

тип_переменной[] название_массива;

Определим массив целых чисел:

int[] numbers;

После определения переменной массива можно присвоить ей определенное значение:

```
int[] nums = new int[4];
```

Также мы сразу можем указать значения для этих элементов:

```
int[] nums2 = new int[4] { 1, 2, 3, 5 };

int[] nums3 = new int[] { 1, 2, 3, 5 };

int[] nums4 = new[] { 1, 2, 3, 5 };

int[] nums5 = { 1, 2, 3, 5 };
```

Все перечисленные выше способы будут равноценны.

Начиная с версии C# 12 для определения массивов можно использовать выражения коллекций, которые предполагают заключение элементов массива в квадратные скобки:

```
int[] nums1 = [ 1, 2, 3, 5 ];

int[] nums2 = [];    // пустой массив
```

Для обращения к элементам массива используются индексы. Индекс представляет номер элемента в массиве, при этом нумерация начинается с нуля, поэтому индекс первого элемента будет равен 0, индекс четвертого элемента - 3.

Используя индексы, можно, как получить элементы массива:

```
int[] numbers = { 1, 2, 3, 5 };

Console.WriteLine(numbers[3]);

//получение эл-та массива 5
```

Так и изменить элемент массива по индексу:

```
numbers[1] = 505;

Console.WriteLine(numbers[1]);  // 505
```

Каждый массив имеет свойство Length, которое хранит длину массива. Например, получим длину массива numbers:

```
int[] numbers = { 1, 2, 3, 5 };

Console.WriteLine(numbers.Length);  // 4
```

Для перебора массивов можно использовать различные типы циклов. Например, цикл **foreach**:

```
int[] numbers = { 1, 2, 3, 4, 5 };

foreach (int i in numbers)
```

```
{
        Console.WriteLine(i);
}
```

Аналогично подобные действия можно сделать и с помощью цикла **for**:

```
int[] numbers = { 1, 2, 3, 4, 5 };
for (int i = 0; i < numbers.Length; i++)
{
        Console.WriteLine(numbers[i]);
}
```

Также можно использовать и другие виды циклов, например, **while**:

```
int[] numbers = { 1, 2, 3, 4, 5 };
int i = 0;
while(i < numbers.Length)
{
        Console.WriteLine(numbers[i]);
        i++;
}
```

Массивы, которые имеют два измерения (ранг равен 2) называют двухмерными. Например, создадим одномерный и двухмерный массивы, которые имеют одинаковые элементы:

```
int[] nums1 = new int[] { 0, 1, 2, 3, 4, 5 };

int[,] nums2 = { { 0, 1, 2 }, { 3, 4, 5 } };
```

Для генерации случайных чисел в программах, написанных на C#, предназначен класс «Random».

```
//Создание объекта для генерации чисел

Random rnd = new Random(245);

//Получить случайное число (в диапазоне от 0 до 10)

int value = rnd.Next(0, 10);

//Вывод числа в консоль

Console.WriteLine(value);
```

Задание 1

Задача:

Калькулятор матриц

Реализуйте программный продукт средствами языка C# со следующим функционалом:

1) Создание двух матриц размерности n*m (значения n и m вводятся с клавиатуры);
2) Заполнение матриц значениями с клавиатуры (по выбору пользователя, с последующим выводом результата на экран);
3) Заполнение матриц рандомными числами в диапазоне [a; b] (значения a и b вводятся с клавиатуры) (по выбору пользователя, с последующим выводом результата на экран);
4) Сложение матриц (предусмотреть проверку на возможность выполнения операции, с последующим выводом результата на экран);
5) Умножение матриц (предусмотреть проверку на возможность выполнения операции, с последующим выводом результата на экран);
6) Нахождение детерминанта (определителя) матрицы (предусмотреть проверку на возможность выполнения операции, с последующим выводом результата на экран);
7) Нахождение обратной матрицы (предусмотреть проверку на возможность выполнения операции, с последующим выводом результата на экран);
8) Транспонирование матриц (с последующим выводом результата на экран);
9) Нахождение корней системы уравнений, заданных матрицей (с последующим выводом результата на экран).

При тестировании продемонстрировать успешное выполнение всех пунктов (положительный сценарий), а также обработку следующих ситуаций (негативный сценарий):

1) Невозможность сложения матриц по причине несоответствия их размерностей;
2) Невозможность умножения матриц в связи с их несовместимостью;
3) Невозможность нахождения детерминанта у не квадратных матриц (n!=m);
4) Невозможность нахождения обратной матрицы в случае, если детерминант равен нулю (d=0);
5) Невозможность нахождения корней систему уравнений, если она не имеет решения или не имеет однозначного решения.

Весь функционал должен быть реализован вами, программы, разработанные с использованием сторонних решений (библиотеки, фреймворки и т.д.) реализующих функционал, приниматься не будут.

Решение:

```
using System;

class MatrixCalculator
{
   static void Main(string[] args)
   {
      Console.WriteLine("=== ДЕМОНСТРАЦИЯ ПОЛОЖИТЕЛЬНОГО
СЦЕНАРИЯ ===");
      RunPositiveScenario();

      Console.WriteLine("\n=== ДЕМОНСТРАЦИЯ НЕГАТИВНОГО
СЦЕНАРИЯ ===");
      RunNegativeScenario();

      Console.WriteLine("\n=== РАБОТА С ИНТЕРАКТИВНЫМ
РЕЖИМОМ ===");
      InteractiveMode();
```

```
    }

    static void RunPositiveScenario()
    {
        Console.WriteLine("\n--- 1. Создание и заполнение матриц ---");
        double[,] matrix1 = CreateTestMatrix(2, 2, new double[,] { { 2, 1 }, { 3,
4 } });
        double[,] matrix2 = CreateTestMatrix(2, 2, new double[,] { { 1, 0 }, { 2,
3 } });

        PrintMatrix(matrix1, "Матрица 1:");
        PrintMatrix(matrix2, "Матрица 2:");

        Console.WriteLine("\n--- 2. Сложение матриц ---");
        var sum = AddMatrices(matrix1, matrix2);
        PrintMatrix(sum, "Сумма:");

        Console.WriteLine("\n--- 3. Умножение матриц ---");
        var product = MultiplyMatrices(matrix1, matrix2);
        PrintMatrix(product, "Произведение:");

        Console.WriteLine("\n--- 4. Определитель матрицы 1 ---");
        double det = Determinant(matrix1);
        Console.WriteLine($"Определитель: {det:F6}");

        Console.WriteLine("\n--- 5. Обратная матрица ---");
        var inv = InverseMatrix(matrix1);
        PrintMatrix(inv, "Обратная матрица:");

        Console.WriteLine("\n--- 6. Транспонирование ---");
        var transposed = Transpose(matrix1);
        PrintMatrix(transposed, "Транспонированная матрица:");

        Console.WriteLine("\n--- 7. Решение системы уравнений ---");
        double[,] sys = { { 2, 1, 5 }, { 3, 4, 6 } };
        PrintMatrix(sys, "Матрица системы [A|b]:");
        double[] solution = SolveSystem(sys);
        if (solution != null)
        {
            for (int i = 0; i < solution.Length; i++)
                Console.WriteLine($"x{i + 1} = {solution[i]:F6}");
        }
    }

    static void RunNegativeScenario()
    {
        Console.WriteLine("\n--- 1. Невозможность сложения матриц с
разными размерами ---");
        double[,] m1 = CreateTestMatrix(2, 3, new double[,] { { 1, 2, 3 }, { 4, 5,
```

```
6 } });
        double[,] m2 = CreateTestMatrix(2, 2, new double[,] { { 1, 2 }, { 3, 4 }
});
        PrintMatrix(m1, "Матрица 1:");
        PrintMatrix(m2, "Матрица 2:");
        if (m1.GetLength(0) != m2.GetLength(0) || m1.GetLength(1) !=
m2.GetLength(1))
            Console.WriteLine("Ошибка: размерности не совпадают —
сложение невозможно.");

        Console.WriteLine("\n--- 2. Невозможность умножения (столбцы1 !=
строки2) ---");
        double[,] m3 = CreateTestMatrix(2, 3, new double[,] { { 1, 2, 3 }, { 4, 5,
6 } });
        double[,] m4 = CreateTestMatrix(2, 2, new double[,] { { 1, 2 }, { 3, 4 }
});
        PrintMatrix(m3, "Матрица 1:");
        PrintMatrix(m4, "Матрица 2:");
        if (m3.GetLength(1) != m4.GetLength(0))
            Console.WriteLine("Ошибка: умножение невозможно — столбцы1
!= строки2.");

        Console.WriteLine("\n--- 3. Определитель у не квадратной матрицы --
-");
        PrintMatrix(m3, "Матрица:");
        if (m3.GetLength(0) != m3.GetLength(1))
            Console.WriteLine("Ошибка: определитель можно вычислить
только для квадратной матрицы.");

        Console.WriteLine("\n--- 4. Обратная матрица при det = 0 ---");
        double[,] singular = CreateTestMatrix(2, 2, new double[,] { { 1, 2 }, { 2,
4 } });
        PrintMatrix(singular, "Матрица:");
        double detSingular = Determinant(singular);
        if (Math.Abs(detSingular) < 1e-10)
            Console.WriteLine("Ошибка: определитель = 0 → обратная
матрица не существует.");

        Console.WriteLine("\n--- 5. Решение системы без однозначного
решения ---");
        double[,] inconsistent = CreateTestMatrix(2, 3, new double[,] { { 1, 2, 3
}, { 2, 4, 7 } });
        PrintMatrix(inconsistent, "Матрица системы [A|b]:");
        double[] sol = SolveSystem(inconsistent);
        if (sol == null)
            Console.WriteLine("Ошибка: система не имеет решения или имеет
бесконечно много решений.");
    }
```

```
    static void InteractiveMode()
    {
        double[,] matrix1 = null;
        double[,] matrix2 = null;

        while (true)
        {
            Console.WriteLine("\n=== Интерактивный режим ===");
            Console.WriteLine("1. Создать матрицы");
            Console.WriteLine("2. Заполнить матрицу 1 вручную");
            Console.WriteLine("3. Заполнить матрицу 2 вручную");
            Console.WriteLine("4. Заполнить матрицу 1 случайными
числами");
            Console.WriteLine("5. Заполнить матрицу 2 случайными
числами");
            Console.WriteLine("6. Вывести матрицы");
            Console.WriteLine("7. Сложение матриц");
            Console.WriteLine("8. Умножение матриц");
            Console.WriteLine("9. Определитель матрицы");
            Console.WriteLine("10. Обратная матрица");
            Console.WriteLine("11. Транспонировать матрицу");
            Console.WriteLine("12. Решить систему уравнений");
            Console.WriteLine("0. Выход");
            Console.Write("Выберите действие: ");

            if (!int.TryParse(Console.ReadLine(), out int choice))
            {
                Console.WriteLine("Ошибка: введите число от 0 до 12.");
                continue;
            }

            switch (choice)
            {
                case 1:
                    matrix1 = CreateMatrix("1");
                    matrix2 = CreateMatrix("2");
                    if (matrix1 == null || matrix2 == null)
                    {
                        matrix1 = null;
                        matrix2 = null;
                        continue;
                    }
                    Console.WriteLine("Матрицы созданы.");
                    break;

                case 2:
                    if (matrix1 == null) { Console.WriteLine("Матрица 1 не
создана."); continue; }
                    FillMatrixManual(matrix1, "Матрица 1");
```

```
                break;

            case 3:
                if (matrix2 == null) { Console.WriteLine("Матрица 2 не
создана."); continue; }
                FillMatrixManual(matrix2, "Матрица 2");
                break;

            case 4:
                if (matrix1 == null) { Console.WriteLine("Матрица 1 не
создана."); continue; }
                if (!FillMatrixRandomSingle(ref matrix1)) continue;
                Console.WriteLine("Матрица 1 заполнена случайными
числами.");
                break;

            case 5:
                if (matrix2 == null) { Console.WriteLine("Матрица 2 не
создана."); continue; }
                if (!FillMatrixRandomSingle(ref matrix2)) continue;
                Console.WriteLine("Матрица 2 заполнена случайными
числами.");
                break;

            case 6:
                PrintMatrix(matrix1, "Матрица 1:");
                PrintMatrix(matrix2, "Матрица 2:");
                break;

            case 7:
                if (matrix1 == null || matrix2 == null)
                {
                    Console.WriteLine("Матрицы не созданы.");
                    continue;
                }
                if (matrix1.GetLength(0) == matrix2.GetLength(0) &&
matrix1.GetLength(1) == matrix2.GetLength(1))
                {
                    var result = AddMatrices(matrix1, matrix2);
                    PrintMatrix(result, "Результат сложения:");
                }
                else
                {
                    Console.WriteLine("Сложение невозможно: размерности не
совпадают.");
                }
                break;

            case 8:
```

```
                if (matrix1 == null || matrix2 == null)
                {
                    Console.WriteLine("Матрицы не созданы.");
                    continue;
                }
                if (matrix1.GetLength(1) == matrix2.GetLength(0))
                {
                    var result = MultiplyMatrices(matrix1, matrix2);
                    PrintMatrix(result, "Результат умножения:");
                }
                else
                {
                    Console.WriteLine("Умножение невозможно: число
столбцов первой матрицы не равно числу строк второй.");
                }
                break;

            case 9:
                if (matrix1 == null) { Console.WriteLine("Матрица 1 не
создана."); continue; }
                if (matrix1.GetLength(0) == matrix1.GetLength(1))
                {
                    try
                    {
                        double det = Determinant(matrix1);
                        Console.WriteLine($"Определитель матрицы 1:
{det:F6}");
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"Ошибка при вычислении
определителя: {ex.Message}");
                    }
                }
                else
                {
                    Console.WriteLine("Определитель можно вычислить только
для квадратной матрицы.");
                }
                break;

            case 10:
                if (matrix1 == null) { Console.WriteLine("Матрица 1 не
создана."); continue; }
                if (matrix1.GetLength(0) == matrix1.GetLength(1))
                {
                    try
                    {
                        double det = Determinant(matrix1);
```

```
                        if (Math.Abs(det) < 1e-10)
                        {
                            Console.WriteLine("Обратная матрица не существует
(определитель ≈ 0).");
                        }
                        else
                        {
                            var inverse = InverseMatrix(matrix1);
                            PrintMatrix(inverse, "Обратная матрица:");
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"Ошибка при вычислении обратной
матрицы: {ex.Message}");
                    }
                }
                else
                {
                    Console.WriteLine("Обратная матрица возможна только для
квадратной матрицы.");
                }
                break;

            case 11:
                if (matrix1 != null)
                {
                    var transposed1 = Transpose(matrix1);
                    PrintMatrix(transposed1, "Транспонированная матрица 1:");
                }
                if (matrix2 != null)
                {
                    var transposed2 = Transpose(matrix2);
                    PrintMatrix(transposed2, "Транспонированная матрица 2:");
                }
                if (matrix1 == null && matrix2 == null)
                {
                    Console.WriteLine("Нет созданных матриц.");
                }
                break;

            case 12:
                if (matrix1 == null) { Console.WriteLine("Матрица 1 не
создана."); continue; }
                int n = matrix1.GetLength(0);
                int cols = matrix1.GetLength(1);
                if (cols == n + 1)
                {
                    try
```

```
                    {
                        double[] result = SolveSystem(matrix1);
                        if (result != null)
                        {
                            Console.WriteLine("Решение системы:");
                            for (int i = 0; i < result.Length; i++)
                            {
                                Console.WriteLine($"x{i + 1} = {result[i]:F6}");
                            }
                        }
                        else
                        {
                            Console.WriteLine("Система не имеет решения или
имеет бесконечно много решений.");
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"Ошибка при решении системы:
{ex.Message}");
                    }
                }
                else
                {
                    Console.WriteLine("Для решения системы уравнений
матрица должна быть квадратной + столбец свободных членов (размер [n
x (n+1)]).");
                }
                break;

            case 0:
                Console.WriteLine("Выход из программы.");
                return;

            default:
                Console.WriteLine("Неверный выбор. Попробуйте снова.");
                break;
        }
    }
}

static double[,] CreateMatrix(string num)
{
    Console.Write($"Введите размерность матрицы {num} (n m): ");
    string[] input = Console.ReadLine()?.Split();
    if (input == null || input.Length != 2)
    {
        Console.WriteLine("Ошибка: введите 2 числа (n m).");
        return null;
```

```
        }

        if (!int.TryParse(input[0], out int rows) || !int.TryParse(input[1], out int
cols) || rows <= 0 || cols <= 0)
        {
            Console.WriteLine("Ошибка: введите положительные целые
числа.");
            return null;
        }

        return new double[rows, cols];
    }

    static bool FillMatrixRandomSingle(ref double[,] matrix)
    {
        Console.Write("Введите диапазон [a, b]: ");
        string[] range = Console.ReadLine()?.Split();
        if (range == null || range.Length != 2)
        {
            Console.WriteLine("Ошибка: введите 2 числа (a b).");
            return false;
        }

        if (!double.TryParse(range[0], out double a) || !double.TryParse(range[1],
out double b))
        {
            Console.WriteLine("Ошибка: введите числа для диапазона.");
            return false;
        }

        Random rand = new Random();
        FillWithRandom(matrix, a, b, rand);
        return true;
    }

    static double[,] CreateTestMatrix(int rows, int cols, double[,] data)
    {
        double[,] matrix = new double[rows, cols];
        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
                matrix[i, j] = data[i, j];
        return matrix;
    }

    static void FillMatrixManual(double[,] matrix, string name)
    {
        Console.WriteLine($"Заполнение {name}:");
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
```

```
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                while (true)
                {
                    Console.Write($"Введите элемент [{i + 1}, {j + 1}]: ");
                    if (double.TryParse(Console.ReadLine(), out double value))
                    {
                        matrix[i, j] = value;
                        break;
                    }
                    else
                    {
                        Console.WriteLine("Ошибка: введите число.");
                    }
                }
            }
        }
    }

    static void FillWithRandom(double[,] matrix, double min, double max,
Random rand)
    {
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                matrix[i, j] = rand.NextDouble() * (max - min) + min;
            }
        }
    }

    static void PrintMatrix(double[,] matrix, string label)
    {
        if (matrix == null)
        {
            Console.WriteLine(label + " матрица не создана.");
            return;
        }

        Console.WriteLine(label);
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                Console.Write($"{matrix[i, j],8:F2}");
            }
            Console.WriteLine();
        }
    }
```

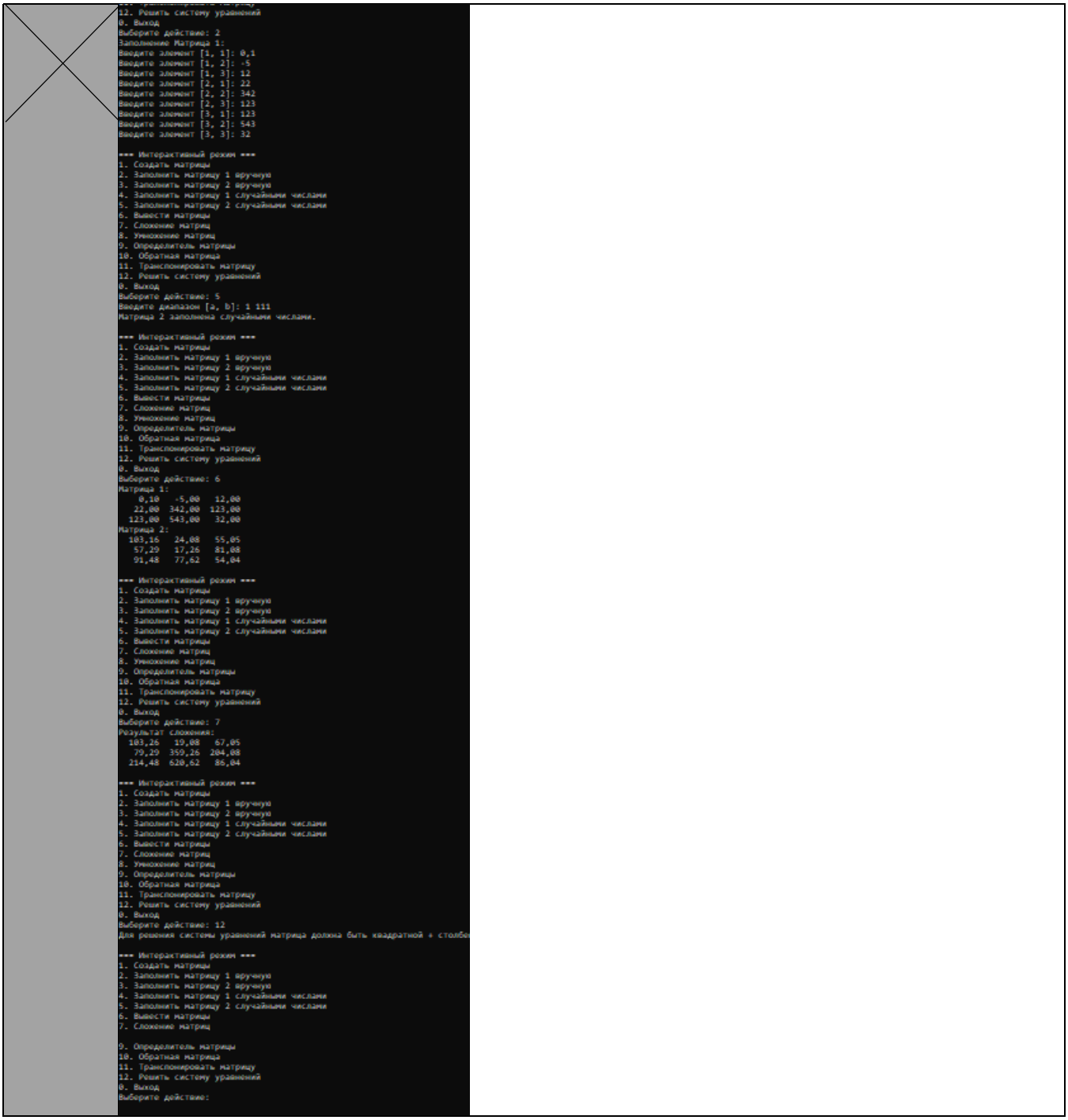
```
static double[,] AddMatrices(double[,] a, double[,] b)
{
   int rows = a.GetLength(0), cols = a.GetLength(1);
   double[,] result = new double[rows, cols];
   for (int i = 0; i < rows; i++)
   {
      for (int j = 0; j < cols; j++)
      {
         result[i, j] = a[i, j] + b[i, j];
      }
   }
   return result;
}

static double[,] MultiplyMatrices(double[,] a, double[,] b)
{
   int rows = a.GetLength(0);
   int cols = b.GetLength(1);
   int inner = a.GetLength(1);
   double[,] result = new double[rows, cols];

   for (int i = 0; i < rows; i++)
   {
      for (int j = 0; j < cols; j++)
      {
         result[i, j] = 0;
         for (int k = 0; k < inner; k++)
         {
            result[i, j] += a[i, k] * b[k, j];
         }
      }
   }
   return result;
}

static double Determinant(double[,] matrix)
{
   int n = matrix.GetLength(0);
   double[,] temp = (double[,])matrix.Clone();
   double det = 1;
   for (int i = 0; i < n; i++)
   {
      int maxRow = i;
      for (int k = i + 1; k < n; k++)
         if (Math.Abs(temp[k, i]) > Math.Abs(temp[maxRow, i]))
            maxRow = k;

      if (maxRow != i)
```

```
            {
                for (int j = 0; j < n; j++)
                {
                    double t = temp[i, j];
                    temp[i, j] = temp[maxRow, j];
                    temp[maxRow, j] = t;
                }
                det *= -1;
            }

            if (Math.Abs(temp[i, i]) < 1e-10)
                return 0;

            det *= temp[i, i];
            for (int k = i + 1; k < n; k++)
            {
                double factor = temp[k, i] / temp[i, i];
                for (int j = i; j < n; j++)
                    temp[k, j] -= factor * temp[i, j];
            }
        }
        return det;
    }

    static double[,] InverseMatrix(double[,] matrix)
    {
        int n = matrix.GetLength(0);
        double[,] augmented = new double[n, 2 * n];
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
                augmented[i, j] = matrix[i, j];
            augmented[i, i + n] = 1;
        }

        for (int i = 0; i < n; i++)
        {
            double pivot = augmented[i, i];
            for (int j = 0; j < 2 * n; j++)
                augmented[i, j] /= pivot;

            for (int k = 0; k < n; k++)
            {
                if (k != i)
                {
                    double factor = augmented[k, i];
                    for (int j = 0; j < 2 * n; j++)
                        augmented[k, j] -= factor * augmented[i, j];
                }
```

```
            }
        }

        double[,] inv = new double[n, n];
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
                inv[i, j] = augmented[i, j + n];
        }
        return inv;
    }

    static double[,] Transpose(double[,] matrix)
    {
        int rows = matrix.GetLength(0), cols = matrix.GetLength(1);
        double[,] result = new double[cols, rows];
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                result[j, i] = matrix[i, j];
            }
        }
        return result;
    }

    static double[] SolveSystem(double[,] augmentedMatrix)
    {
        int n = augmentedMatrix.GetLength(0);
        double[,] temp = (double[,])augmentedMatrix.Clone();

        for (int i = 0; i < n; i++)
        {
            int maxRow = i;
            for (int k = i + 1; k < n; k++)
                if (Math.Abs(temp[k, i]) > Math.Abs(temp[maxRow, i]))
                    maxRow = k;

            if (maxRow != i)
            {
                for (int j = 0; j <= n; j++)
                {
                    double t = temp[i, j];
                    temp[i, j] = temp[maxRow, j];
                    temp[maxRow, j] = t;
                }
            }

            if (Math.Abs(temp[i, i]) < 1e-10)
```

```
                return null;

            for (int k = i + 1; k < n; k++)
            {
                double factor = temp[k, i] / temp[i, i];
                for (int j = i; j <= n; j++)
                    temp[k, j] -= factor * temp[i, j];
            }
        }

        double[] x = new double[n];
        for (int i = n - 1; i >= 0; i--)
        {
            x[i] = temp[i, n];
            for (int j = i + 1; j < n; j++)
                x[i] -= temp[i, j] * x[j];
            x[i] /= temp[i, i];
        }
        return x;
    }
}
```

Ответ:


12. Решить систему уравнений
0. Выход
Выберите действие: 2
Заполнение Матрица 1:
Введите элемент [1, 1]: 0,1
Введите элемент [1, 2]: -5
Введите элемент [1, 3]: 12
Введите элемент [2, 1]: 22
Введите элемент [2, 2]: 342
Введите элемент [2, 3]: 123
Введите элемент [3, 1]: 123
Введите элемент [3, 2]: 543
Введите элемент [3, 3]: 32
=== Интерактивный режим ===
1. Создать матрицы
2. Заполнить матрицу 1 вручную
3. Заполнить матрицу 2 вручную
4. Заполнить матрицу 1 случайными числами
5. Заполнить матрицу 2 случайными числами
6. Вывести матрицы
7. Сложение матриц
8. Умножение матриц
9. Определитель матрицы
10. Обратная матрица
11. Транспонировать матрицу
12. Решить систему уравнений
0. Выход
Выберите действие: 5
Введите диапазон [a, b]: 1 111
Матрица 2 заполнена случайными числами.
=== Интерактивный режим ===
1. Создать матрицы
2. Заполнить матрицу 1 вручную
3. Заполнить матрицу 2 вручную
4. Заполнить матрицу 1 случайными числами
5. Заполнить матрицу 2 случайными числами
6. Вывести матрицы
7. Сложение матриц
8. Умножение матриц
9. Определитель матрицы
10. Обратная матрица
11. Транспонировать матрицу
12. Решить систему уравнений
0. Выход
Выберите действие: 6
Матрица 1:
0,10 -5,00 12,00
22,00 342,00 123,00
123,00 543,00 32,00
Матрица 2:
103,16 24,08 55,05
57,29 17,26 81,08
91,48 77,62 54,04
=== Интерактивный режим ===
1. Создать матрицы
2. Заполнить матрицу 1 вручную
3. Заполнить матрицу 2 вручную
4. Заполнить матрицу 1 случайными числами
5. Заполнить матрицу 2 случайными числами
6. Вывести матрицы
7. Сложение матриц
8. Умножение матриц
9. Определитель матрицы
10. Обратная матрица
11. Транспонировать матрицу
12. Решить систему уравнений
0. Выход
Выберите действие: 7
Результат сложения:
103,26 19,08 67,05
79,29 359,26 204,08
214,48 620,62 86,04
=== Интерактивный режим ===
1. Создать матрицы
2. Заполнить матрицу 1 вручную
3. Заполнить матрицу 2 вручную
4. Заполнить матрицу 1 случайными числами
5. Заполнить матрицу 2 случайными числами
6. Вывести матрицы
7. Сложение матриц
8. Умножение матриц
9. Определитель матрицы
10. Обратная матрица
11. Транспонировать матрицу
12. Решить систему уравнений
0. Выход
Выберите действие: 12
Для решения системы уравнений матрица должна быть квадратной + столбе
=== Интерактивный режим ===
1. Создать матрицы
2. Заполнить матрицу 1 вручную
3. Заполнить матрицу 2 вручную
4. Заполнить матрицу 1 случайными числами
5. Заполнить матрицу 2 случайными числами
6. Вывести матрицы
7. Сложение матриц
9. Определитель матрицы
10. Обратная матрица
11. Транспонировать матрицу
12. Решить систему уравнений
0. Выход
Выберите действие: