Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 11

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Методы»

Выполнил: ст. гр. ТУУ-111

Михалин. А. В. Вариант №2

27.05.2025

(дата выполнения)

Проверил: к.т.н., доц. Сафронов А.И.

11.06.2025

(дата приёмки)

Москва – 2025 г.

**1. Цель**

Необходимо разработать программное обеспечение для вывода двумерного массива змейкой в формате метода на языке программирования *C#*.

**2. Формулировка задачи**

Составить типизированный метод для вывода элементов двумерного массива змейкой, начиная с позиции, обусловленной наличием логического признака (флага). Для печати из правого нижнего угла влево использовать значение *true*, для печати из левого верхнего угла – значение *false*. Отвечать логическим признаком наличия/отсутствия не размещённых на экране элементов массива. При наименовании метода использовать префикс «*\_b*».

3**. Блок-схема алгоритма**

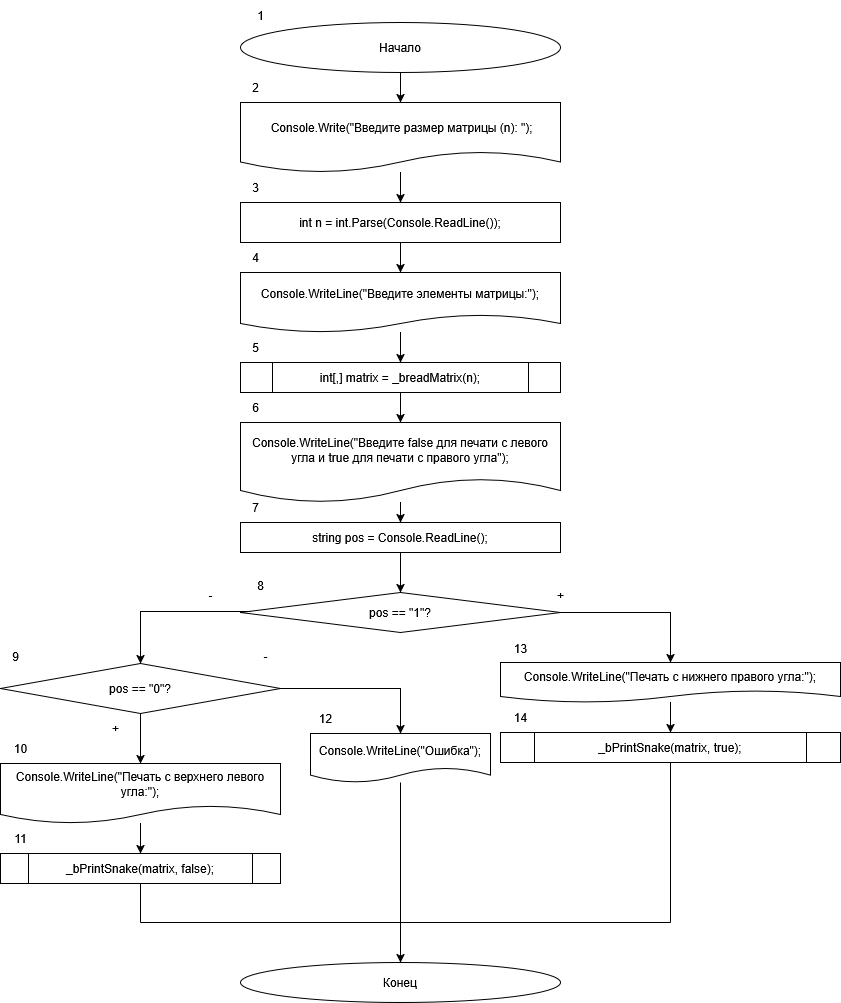
****

Рисунок 3.1 – Блок-схема алгоритма

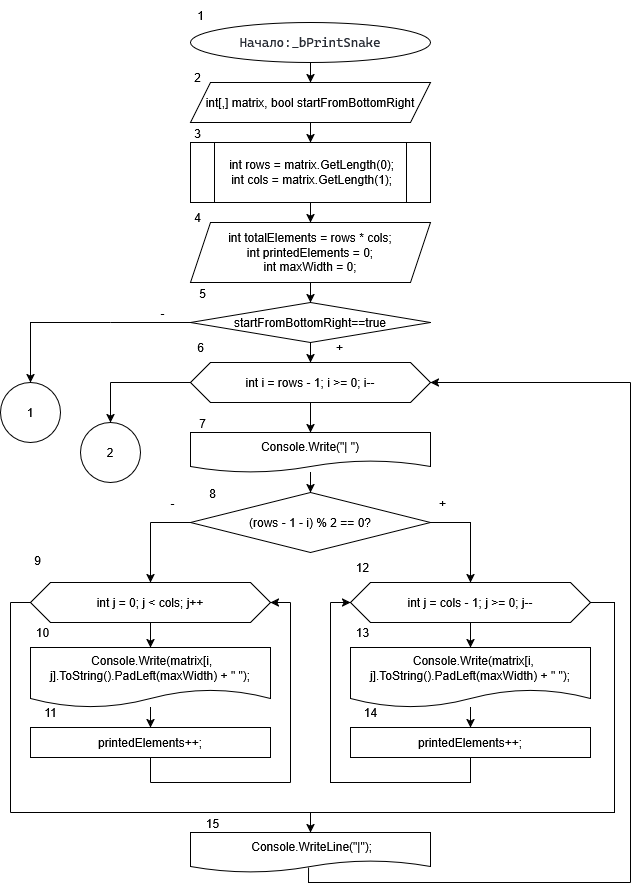
****

Рисунок 3.2 – Блок-схема алгоритма

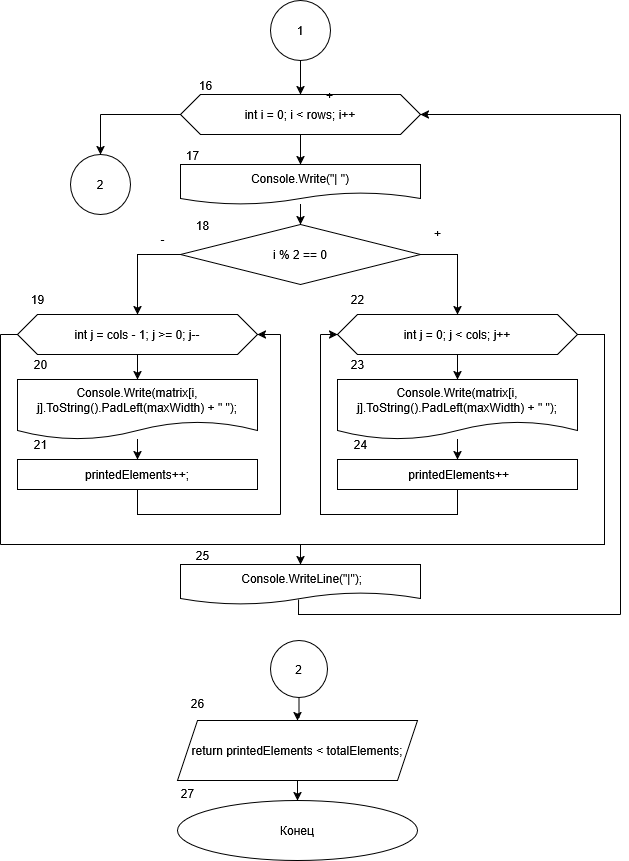
****

Рисунок 3.3 – Блок-схема алгоритма

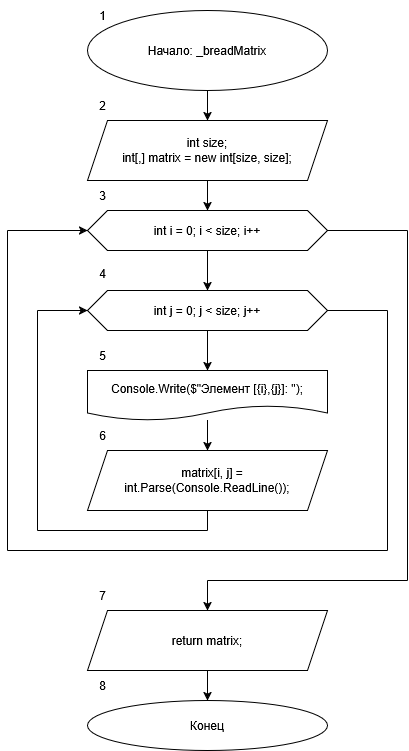
****

Рисунок 3.4 – Блок-схема алгоритма

**4. Подбор тестовых примеров**

В качестве тестового примера будет использоваться введенный двухмерный массив:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| 35 | 36 | 37 | 38 | 39 | 40 | 41 |
| 42 | 43 | 44 | 45 | 46 | 47 | 48 |

А по результатам работы программы могут получится 2 :

При введении true:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 13 | 12 | 11 | 10 | 9 | 8 | 7 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 27 | 26 | 25 | 24 | 23 | 22 | 21 |
| 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| 41 | 40 | 39 | 38 | 37 | 36 | 35 |
| 42 | 43 | 44 | 45 | 46 | 47 | 48 |

При введении false:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 48 | 47 | 46 | 45 | 44 | 43 | 42 |
| 35 | 36 | 37 | 38 | 39 | 40 | 41 |
| 34 | 33 | 32 | 31 | 30 | 29 | 28 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 20 | 19 | 18 | 17 | 16 | 15 | 14 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**5.Листинг**

using System;

class Program

{

// Функция для печати матрицы "змейкой"

// startFromBottomRight = true - начать с нижнего правого угла

// startFromBottomRight = false - начать с верхнего левого угла

static void \_bPrintSnake(int[,] matrix, bool startFromBottomRight)

{

int rows = matrix.GetLength(0); // количество строк

int cols = matrix.GetLength(1); // количество столбцов

// Определяем максимальную длину строки среди всех элементов для выравнивания вывода

int maxWidth = 0;

foreach (var item in matrix)

{

int len = item.ToString().Length;

if (len > maxWidth) maxWidth = len;

}

if (startFromBottomRight)

{

// Обход строк с нижней вверх

for (int i = rows - 1; i >= 0; i--)

{

Console.Write("| ");

// Чередуем направление обхода столбцов для "змейки"

if ((rows - 1 - i) % 2 == 0)

{

for (int j = cols - 1; j >= 0; j--)

Console.Write(matrix[i, j].ToString().PadLeft(maxWidth) + " ");

}

else

{

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

Console.Write(matrix[i, j].ToString().PadLeft(maxWidth) + " ");

}

Console.WriteLine("|");

}

}

else

{

// Обход строк сверху вниз

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

{

Console.Write("| ");

// Чередуем направление обхода столбцов для "змейки"

if (i % 2 == 0)

{

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

Console.Write(matrix[i, j].ToString().PadLeft(maxWidth) + " ");

}

else

{

for (int j = cols - 1; j >= 0; j--)

Console.Write(matrix[i, j].ToString().PadLeft(maxWidth) + " ");

}

Console.WriteLine("|");

}

}

}

// Функция для ввода элементов квадратной матрицы заданного размера

static int[,] \_breadMatrix(int size)

{

int vvod = 0;

int[,] matrix = new int[size, size];

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

{

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

{

Console.Write($"Элемент [{i},{j}]: ");

// Проверка корректности ввода (целое число)

while (!int.TryParse(Console.ReadLine(), out vvod))

Console.WriteLine("Ошибка! Введите целое положительное число:");

matrix[i, j] = vvod;

}

}

return matrix;

}

static void Main()

{

Console.Write("Введите размер двухмерного массива (n): ");

int n = 0;

// Проверка корректности ввода размера массива

while (!int.TryParse(Console.ReadLine(), out n) || n <= 0)

Console.WriteLine("Ошибка! Введите целое положительное число:");

Console.WriteLine("Введите элементы матрицы:");

int[,] matrix = \_breadMatrix(n);

Console.WriteLine("Начать змейку с верхнего левого угла ? (true / false) : ");

string pos = Console.ReadLine();

if (pos == "true")

{

// Старт с верхнего левого угла

Console.WriteLine("Печать с верхнего левого угла:");

\_bPrintSnake(matrix, false);

}

else if (pos == "false")

{

// Старт с нижнего правого угла

Console.WriteLine("Печать с нижнего правого угла:");

\_bPrintSnake(matrix, true);

}

else

{

Console.WriteLine("Ошибка: введите 'true' или 'false'.");

}

Console.ReadKey();

}

}

**6. Расчёт тестовых примеров на ПК**

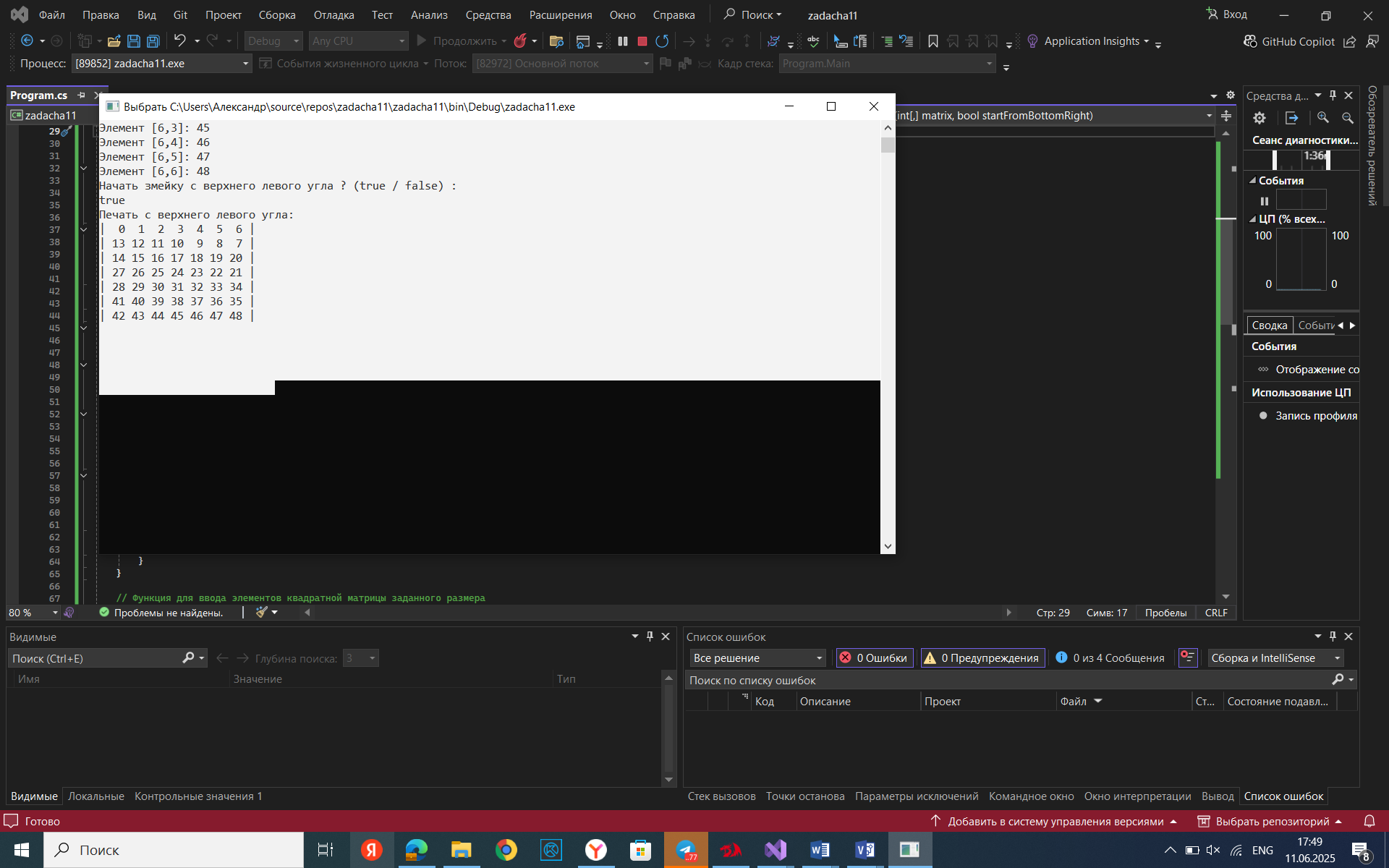
****

Рисунок 6.1 – Тестовый пример с выводом матрицы из левого верхнего угла

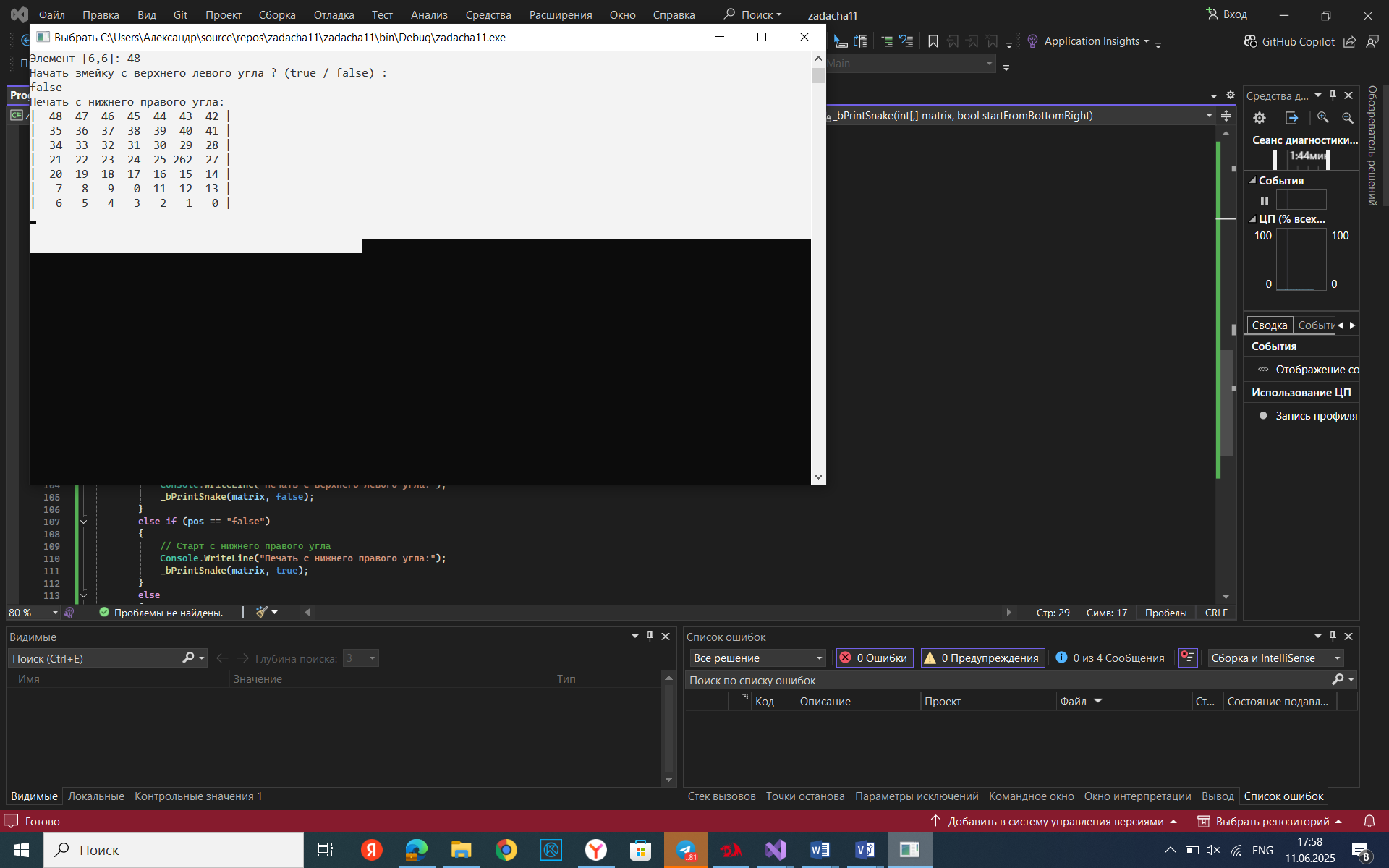
****

Рисунок 6.2 – Тестовый пример с выводом матрицы из правого нижнего угла

**7. Вывод по работе**

Разработано программное обеспечение, удовлетворяющее требованиям задания. Существующий метод грамотно выводит в консоль двумерный массив змейкой от двух противоположных углов в зависимости от выбранного флага.