Минобрнауки РФ

ФГБОУ ВО Тверской Государственный Технический Университет

Кафедра  “Программное Обеспечение”

Лабораторная работа N2

По дисциплине “Структуры и алгоритмы обработки данных”

Выполнил: студент группы Б.ПИН.РИС-22.06

Иванов Алексей Михайлович

Проверил: Мальков Александр Анатольевич

г. Тверь, 2023

Оглавление

[**Часть 1. Постановка задачи.** 3](#_Toc145406338)

[**Часть 2. Алгоритм решения.** 3](#_Toc145406339)

[**Часть 3. Оценка временной эффективности в терминах О-функций.** 4](#_Toc145406340)

[**Часть 4. Программный код.** 5](#_Toc145406341)

[**Часть 5. Тестирование и результаты работы.** 9](#_Toc145406342)

# **Часть 1. Постановка задачи.**

Работа с разреженными массивами с математическим описанием местоположения нефоновых элементов:

преобразование индексов разреженного массива в индекс вектора;

чтение значения элемента массива из его упакованного представления по двум индексам (строка, столбец);

запись значения элемента массива в его упакованное представление по двум индексам.

# **Часть 2. Алгоритм решения.**

К данному типу матриц относятся матрицы, у которых местоположения элементов со значениями, отличными от фонового, могут быть математически описаны, то есть в их расположении есть какая-либо закономерность.

Элементы, значения которых являются фоновыми, называют нулевыми; а элементы, значения которых отличны от фонового, называют ненулевыми. Но необходимо помнить, что фоновое значение не всегда равно нулю.

Ненулевые значения хранятся, как правило, в одномерном массиве (векторе), а связь между местоположением в разреженной матрице и в новом, одномерном, описывается математически с помощью формулы, преобразующей индексы матрицы в индексы вектора.

Для работы с разреженной матрицей разрабатываются функции:

1. для преобразования индексов матрицы в индекс вектора;
2. для получения значения элемента матрицы из ее упакованного представления по двум индексам (строка, столбец);
3. для записи значения элемента матрицы в ее упакованное представление по двум индексам.

Например, в данной работе используется классический вариант, когда фоновые элементы задаются в шахматном порядке. Пример:

0 a 0 b

c 0 d 0

0 e 0 f

g 0 h 0

В таком случае очевидно, что матрицу полностью хранить необязательно – половина элементов пустые, и известно, где они находятся. Их можно поместить в вектор матрицы, и когда пользователю будет необходимо получить значение из матрицы, достаточно будет лишь найти номер элемента в векторе и работать с этим элементом.

В таком случае, для получения доступа к элементам, нужно разработать функцию, принимающую в качестве аргументов x и y координаты числа, а возвращающую индекс в векторе.

Заметим, что для четных строк столбцы с элементами имеют нечетный номер, а для нечетных строк – наоборот четный, если считать с нуля.

Матрицу можно разбить на строки, причем оставить в них только нефоновые элементы. В итоговом массиве будем хранить последовательности таких строк, его длина тогда будет len\*len/2. Пусть номер строки будет т.н. шагом для номера столбца, то есть к номеру столбца, обработанному неким образом, мы будем добавлять половину длины массива – тогда в векторе как раз будут последовательно упакованы строки необходимым образом. Теперь лишь остается определить, как нужно действовать с номером столбца.

Возвращаясь к вышенаписанному замечанию, можно также вспомнить, что и четные числа вида 2n, и нечетные числа вида 2n+1, дают в результате целочисленного деления на 2 одно и то же число – n. То есть целочисленное деление номера столбца на 2 поможет избавиться от необходимости определять номер строки в векторе. И тогда для получения индекса, необходимо будет сложить x\*len/2 и y/2.

Для получения элемента еще необходимо будет определить, находится ли элемент в векторе или он фоновый. Вернемся к замечанию. Из него можно также сделать вывод, что если на четной строке номер столбца элемента четный, то он фоновый, и что если на нечетной строке номер столбца элемента нечетный, то он также фоновый. Это и ляжет в основу алгоритма получения элемента.

# **Часть 3. Оценка временной эффективности в терминах О-функций.**

Составим сводную таблицу эффективности для матрицы в обоих типах представлений.

|  |  |  |
| --- | --- | --- |
|  | Классический вид | Представление в векторе |
| Вставка по индексу | O(1) | O(1) |
| Замена по индексу | О(1) | O(1) |
| Удаление по индексу | O(1) | O(1) |
| Получение по индексу | O(1) | O(1) |
| Память | O() | O() |

Классический вид матрицы предполагает мгновенный доступ ко всем элементам с любой целью при известном индексе. Однако это требует хранения абсолютно всех элементов, даже фоновых.

При представлении в векторе все то же самое: вычисление индексов это операции, которые всегда выполняются 1 раз и за константное время.

Однако в случае, если фоновые элементы описываются математически, представление матрицы в векторе дает выигрыш в памяти в 2 раза, что может быть существенно при больших объемах данных.

# **Часть 4. Программный код.**

Итоговая библиотека классов имеет следующий вид:

/// <summary>

/// Класс, отобрающий матрицу, описываемую

/// математически как шахматную с нулями на белых клетках

/// </summary>

public class ChessMatrix

{

/// <summary>

/// Длина матрицы

/// </summary>

public int len;

/// <summary>

/// Вектор матрицы

/// </summary>

public int[] vector;

/// <summary>

/// Конструктор, создающий шахматную матрицу данной длины

/// </summary>

/// <param name="size">Длина шахматной матрицы</param>

public ChessMatrix(int size)

{

len = size;

vector = new int[(len \* len) / 2];

}

private int VectorIndex(int x, int y)

{

return x \* len / 2 + y / 2;

}

/// <summary>

/// Назначает удаляет элемент на данной позиции

/// </summary>

/// <param name="x">x- координата элемента</param>

/// <param name="y">y- координата элемента</param>

/// <returns>True: назначение возможно

/// False: во всех иных случаях </returns>

public void DeleteValue(int x, int y)

{

this.SetValue(x, y, default);

}

/// <summary>

/// Назначает новый элемент на данную позицию

/// </summary>

/// <param name="x">x- координата нового элемента</param>

/// <param name="y">y- координата нового элемента</param>

/// <param name="value">Назначаемый элемент</param>

/// <returns>True: назначение возможно

/// False: во всех иных случаях </returns>

public bool SetValue(int x, int y, int value)

{

if (!(

(y % 2 == 0 && x % 2 == 0) ||

((y + 1) % 2 == 0 && (x + 1) % 2 == 0)

))

{

vector[VectorIndex(x, y)] = value;

return true;

}

else return false;

}

/// <summary>

/// Получает элемент по данной позиции

/// </summary>

/// <param name="x">x- координата элемента</param>

/// <param name="y">y- координата элемента</param>

/// <returns>Элемент на данной позиции</returns>

public int GetValue(int x, int y)

{

if (

(y % 2 == 0 && x % 2 == 0) ||

((y + 1) % 2 == 0 && (x + 1) % 2 == 0)

)

return 0;

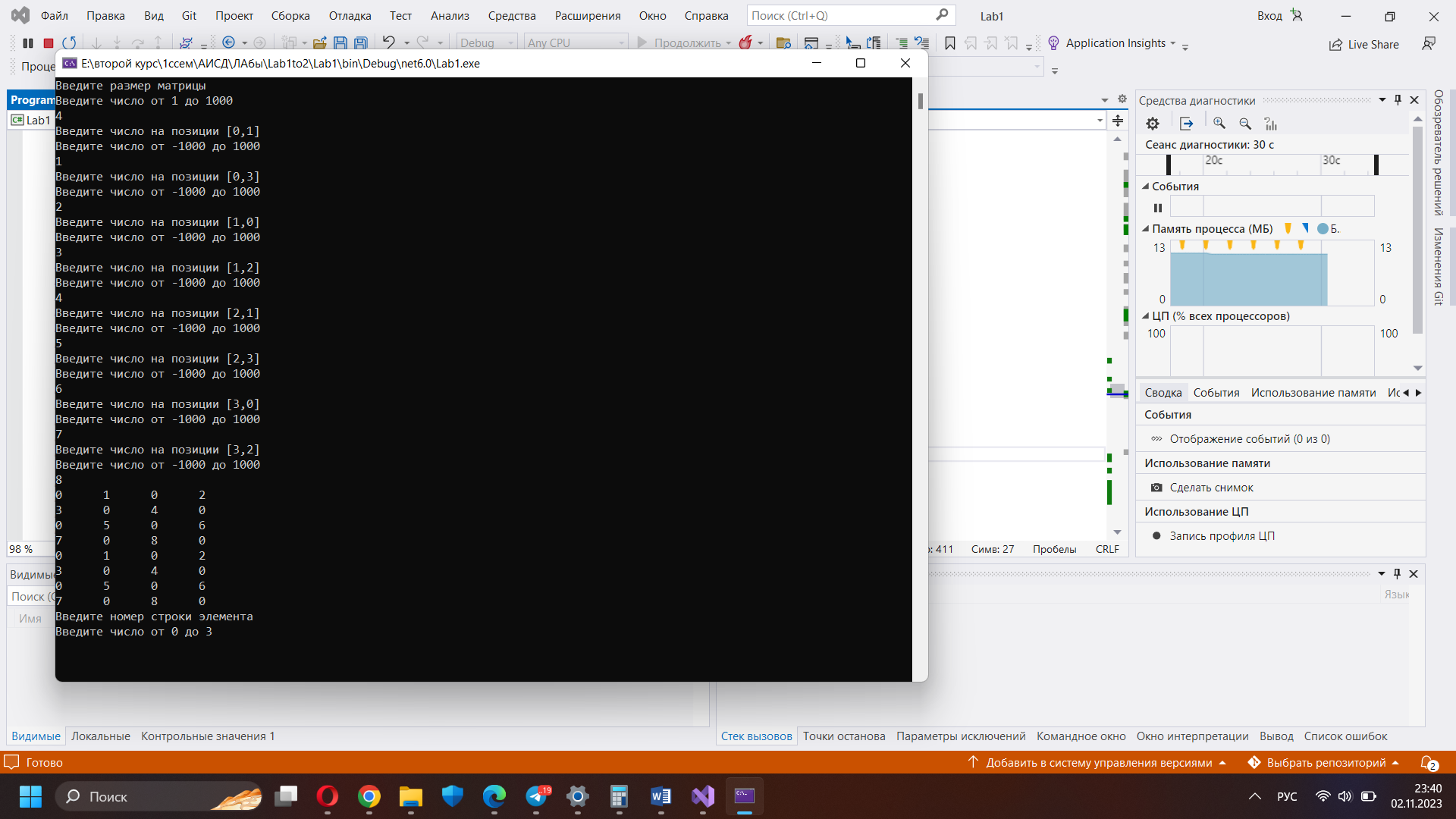
else

return vector[VectorIndex(x, y)];

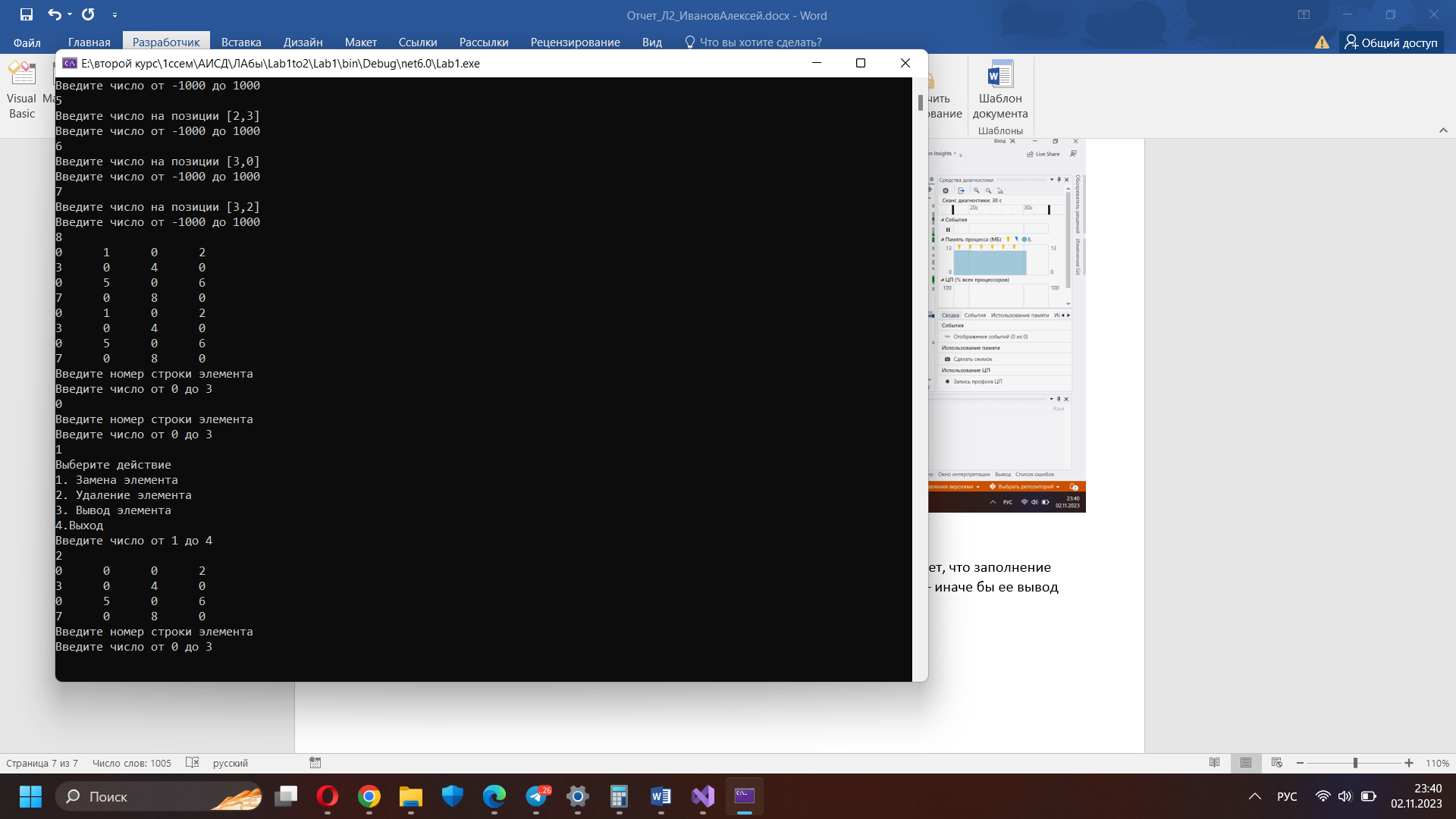
}

# }

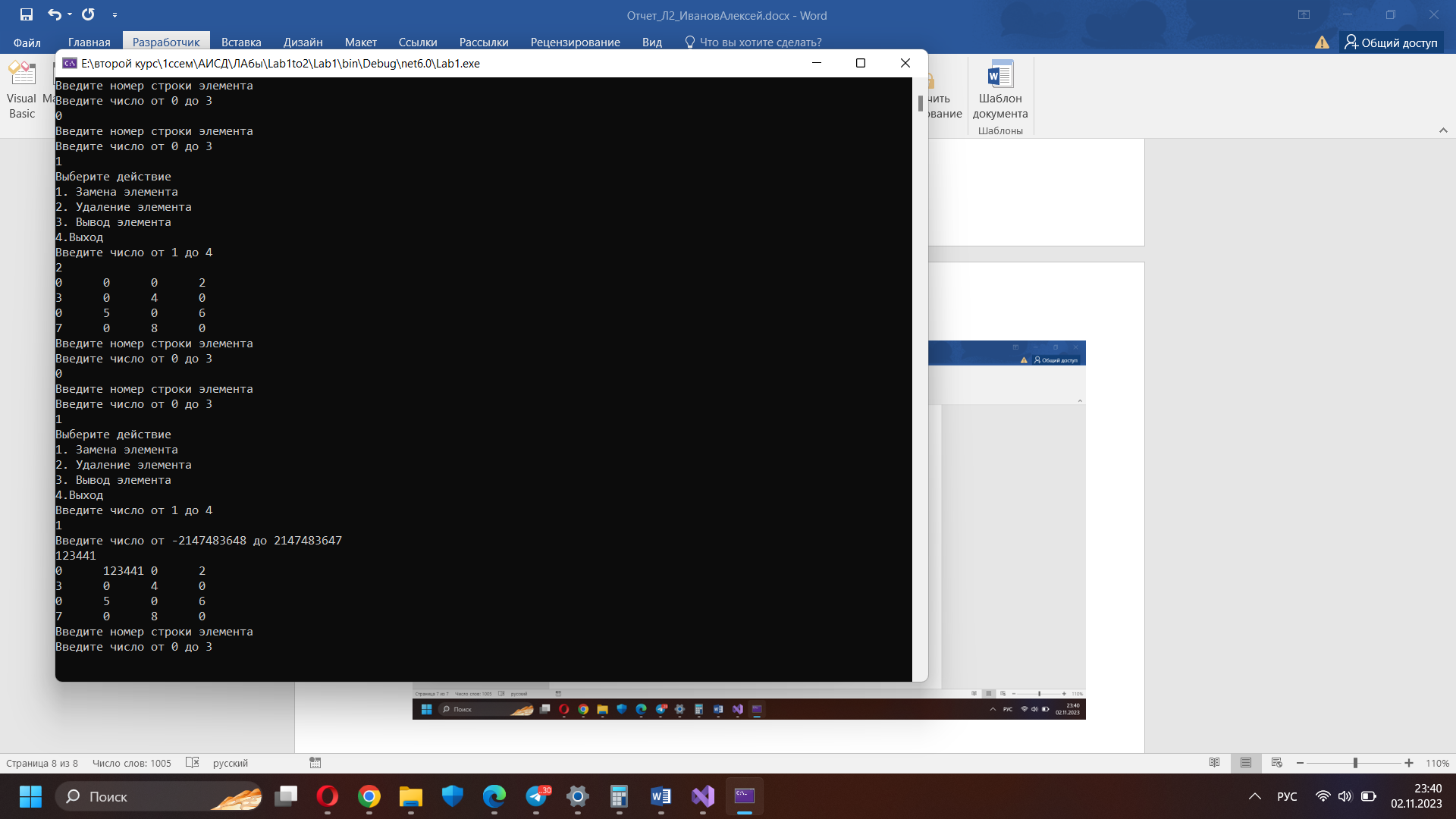
# **Часть 5. Тестирование и результаты работы.**



Для тестирования было построено консольное приложение. Оно демонстрирует, что заполнение матрицы идет корректно, а также получение элементов работает как следует – иначе бы ее вывод был невозможен.



Удаление элемента также работает корректно.



Замена также происходит правильно.