ГУАП КАФЕДРА № 43

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ ПРЕПОДАВАТЕЛЬ

|  |  |  |
| --- | --- | --- |
| ассистент |  | М.А. Мурашова |
| должность, уч. степень, звание | подпись, дата | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3 |
| ОСНОВЫ JAVASCRIPT |
| по дисциплине: WEB-ТЕХНОЛОГИИ |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |
| --- | --- | --- |
| СТУДЕНТ ГР. | 4232 | Г. П. Матюшков |
|  | подпись, дата | инициалы, фамилия |

Санкт-Петербург

2025

**Содержание**

[1. Цель работы 3](#_Toc192092890)

[2. Задание 3](#_Toc192092891)

[3. Вариант задания 3](#_Toc192092892)

[4. Названия средств, использованных при выполнении работы 3](#_Toc192092893)

[5. Выполнение задания 3](#_Toc192092894)

[6. Листинг программного кода 11](#_Toc192092895)

[7. Выводы 12](#_Toc192092896)

# 1. Цель работы

Знакомство с языком JavaScript.

# 2. Задание

**Базовое задание**

Используя редактор javascript (см. листинг 1) написать программу, которая выполняет задание по варианту (базовая часть) для матрицы произвольного размера (n\*m) и использует массив. Обязательно использовать методы массива. Вывести на страницу матрицу в форме прямоугольника. Размеры матрицы получать через диалог с пользователем.

**Расширенное задание**

Нарисовать заданную вариантом фигуру, используя объект canvas и образец скрипта из листинга 2. Повторяющие фрагменты формировать с помощью циклов.

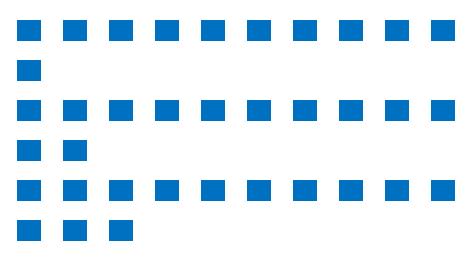
Размер фигуры должен задаваться пользователем через диалог. Под размером понимается количество элементов по вертикали и горизонтали. На пример, для варианта 1 задание можно сформулировать так: построить фигуру, состоящую из четного количества (n) вертикальных синий линий, которые пересекаются с нечетным количеством (n-1) горизонтальных линий, при чем меньшая половина горизонтальных линий синего цвета, а большая половина красного цвета. Пересечения линий должны образовывать квадраты.

При желании до рисования фигуры можно сформировать матрицу, отражающую заполнение прямоугольника, в который вписана фигура, элементами, а потом визуализировать эту матрицу.

# 3. Вариант задания

Базовое задание: 13) На основе исходной матрицы сформировать новую, в которой строка матрицы с номером n, умноженная на число z, прибавлена к строке с номером m.

Расширенное задание: Вариант 13



# 4. Названия средств, использованных при выполнении работы

1. Редактор кода Visual Studio Code.

2. Браузер Yandex.

# 5. Выполнение задания

Ниже представлены таблицы (1-4) описывающие все переменные, методы и функции.

Таблица 1 – описание всех переменных программ

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| rows | Number | Хранит количество строк матрицы, введенное пользователем. |
| cols | Number | Хранит количество столбцов матрицы, введенное пользователем. |
| n | Number | Номер строки для умножения на z (в базовом задании). |
| m | Number | Номер строки, к которой прибавляется результат (в базовом задании). |
| z | Number | Число, на которое умножается строка n (в базовом задании). |
| matrix | Array | Хранит исходную матрицу, заполненную случайными числами. |
| resultMatrix | Array | Хранит результирующую матрицу после выполнения операции. |
| myrezult | String | Хранит строковое представление матриц для вывода в поле результата. |
| canvas | HTMLElement | Ссылка на элемент <canvas> для рисования (в расширенном задании). |
| ctx | CanvasRenderingContext2D | Контекст рисования канваса, используемый для рисования на нем. |
| size | Number | Размер одного прямоугольника в пикселях (в расширенном задании). |
| blue | String | Цвет (HEX) для рисования синих квадратов. |
| green | String | Цвет (строка в формате HEX) для рисования зеленого квадрата. |
| i | Number | Индекс строки при обходе двумерного массива pattern. |
| j | Number | Индекс столбца при обходе двумерного массива pattern. |

Таблица 2 – описание использованных методов массива

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Параметры** | **Назначение** |
| Array.from() | arrayLike, mapFn (опционально) | Создает новый массив из объекта-подобного массива или итерируемого объекта. В данном случае используется для создания массива заданной длины и заполнения его значениями, вычисляемыми через функцию. |
| map() | callback(item, index, array) | Создает новый массив, применяя функцию к каждому элементу. Используется для копирования матрицы и преобразования строк в текст. |
| join() | separator (по умолчанию ',') | Объединяет элементы массива в строку с указанным разделителем. Используется для форматирования матрицы с табуляцией и переносами строк. |

Таблица 3 – описание использованных методов других стандартных объектов

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Параметры** | **Назначение** |
| parseInt() | string, radix (опционально) | Преобразует строку в целое число. Здесь используется для преобразования введенных данных в числа. |
| isNaN() | value | Проверяет, является ли переданное значение NaN. В данном коде проверяется корректность введенных данных для строк и столбцов матрицы. |
| prompt() | message, default (опционально) | Запрашивает ввод данных у пользователя. |
| alert() | message | Выводит модальное окно с сообщением для пользователя. В данном случае, используется для уведомления о некорректных данных. |
| Math.floor() | Number | Округляет число вниз до целого. Используется для генерации случайных чисел. |
| Math.random() | - | Генерирует случайное число от 0 до 1. Используется для заполнения матрицы. |
| getElementById() | id | Возвращает элемент по его ID. Используется для доступа к <canvas>. |
| getContext() | contextType | Возвращает объект контекста рисования для указанного канваса. Здесь используется для получения контекста рисования в 2D. |
| fillRect() | x, y, width, height | Рисует прямоугольник на канвасе, начиная с координат (x, y) и с размерами width и height. Используется для рисования клеток на канвасе. |
| clearRect() | x, y, width, height | Очищает указанную область на <canvas>. Используется для очистки перед рисованием. |

Таблица 4 – описание использованных стрелочных функций

|  |  |  |
| --- | --- | --- |
| **Стрелочная функция** | **Параметры** | **Назначение** |
| (\_, i) => Array.from() | (\_, i) | Эта стрелочная функция используется в Array.from() для генерации строк матрицы. Параметры \_ (не используется) и i — индекс текущей строки. |
| () => Math.floor() | - | Генерирует случайное число для заполнения матрицы (базовое задание). |
| (value, j) => value + ... | value, j | Выполняет операцию сложения элементов строк (базовое задание). |
| row => row.join('\t') | row | Эта стрелочная функция используется для объединения элементов каждой строки матрицы с разделителем \t (табуляция). |

На рисунках 1-7 представлена работа программы базового задания. Программа успешно формирует матрицу произвольного размера, заполненную случайными числами. Операция умножения строки n на z и прибавления к строке m выполняется корректно благодаря использованию метода map. Вывод в виде прямоугольника с табуляцией и переносами строк обеспечивает читаемость. Проверка ввода предотвращает ошибки, информируя пользователя через alert. Программа демонстрирует работу с двумерными массивами и методами Array.from, map, join.

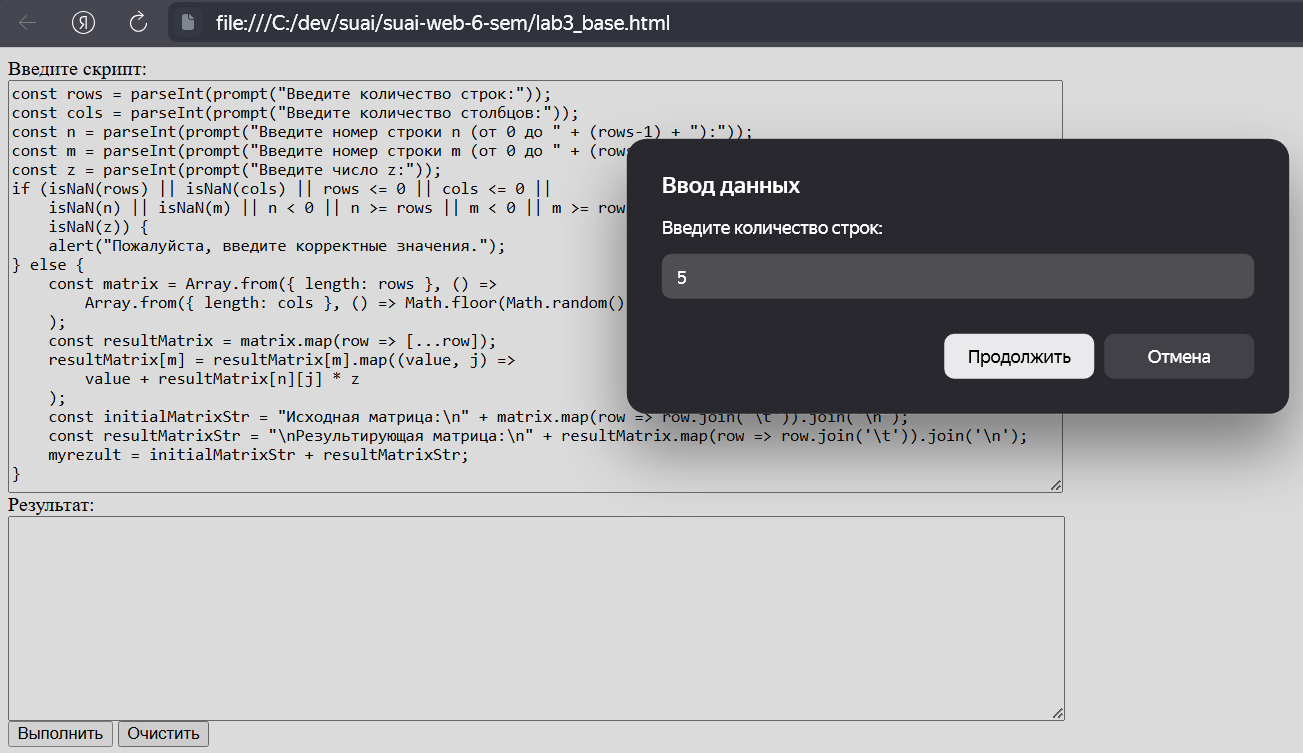


Рисунок 1 – запрос у пользователя количества строк

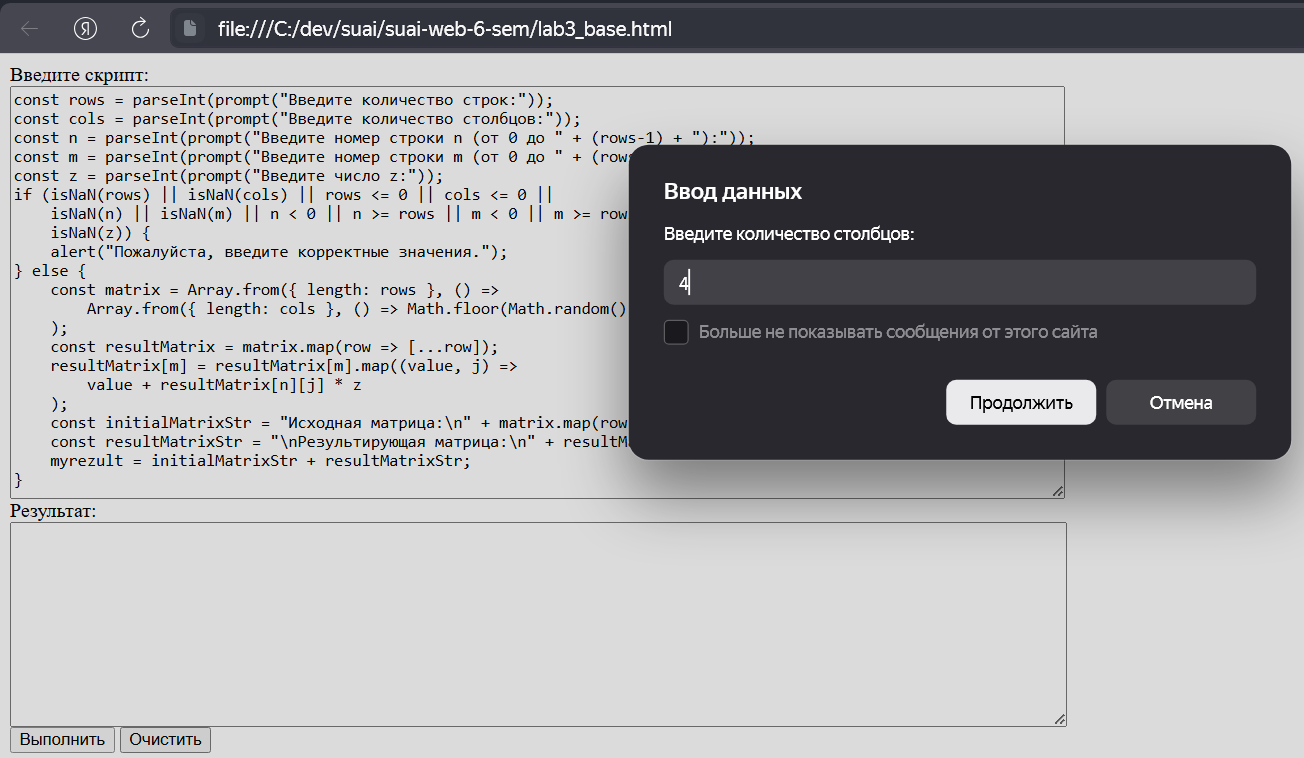


Рисунок 2 – запрос у пользователя количества столбцов

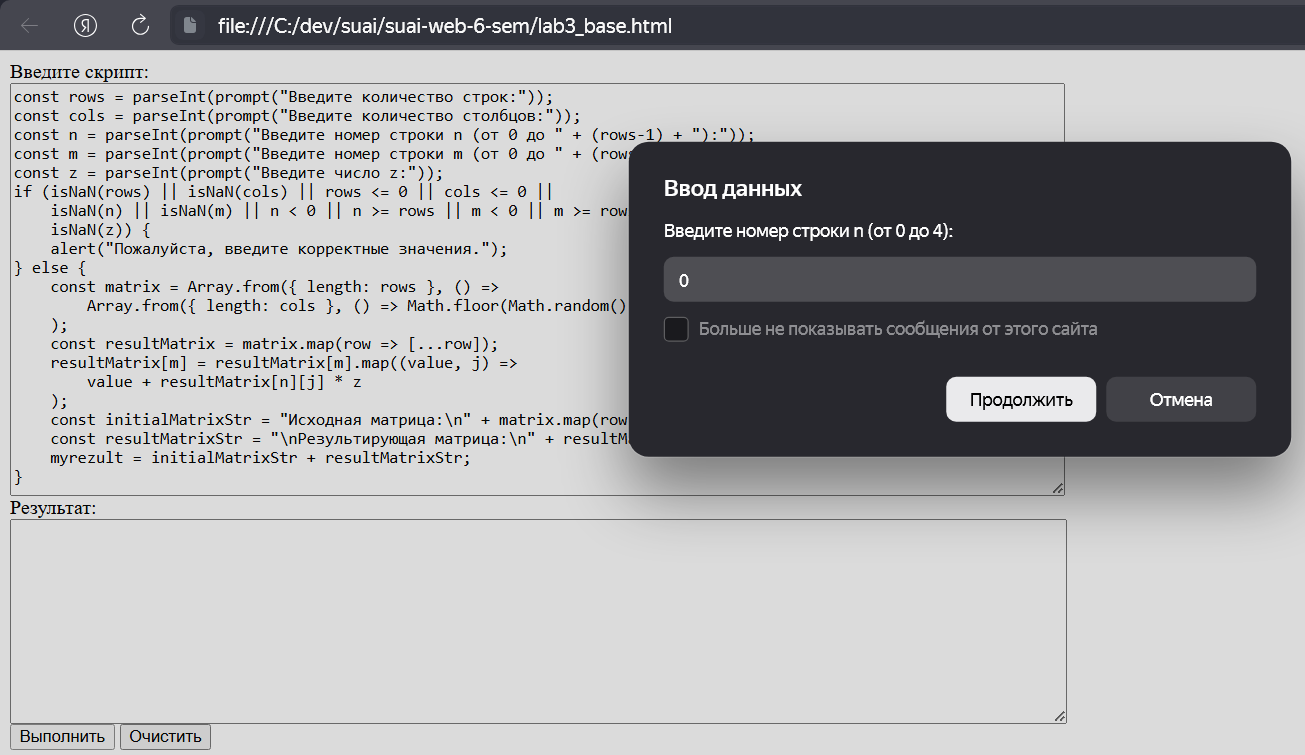


Рисунок 3 – запрос номера строки n

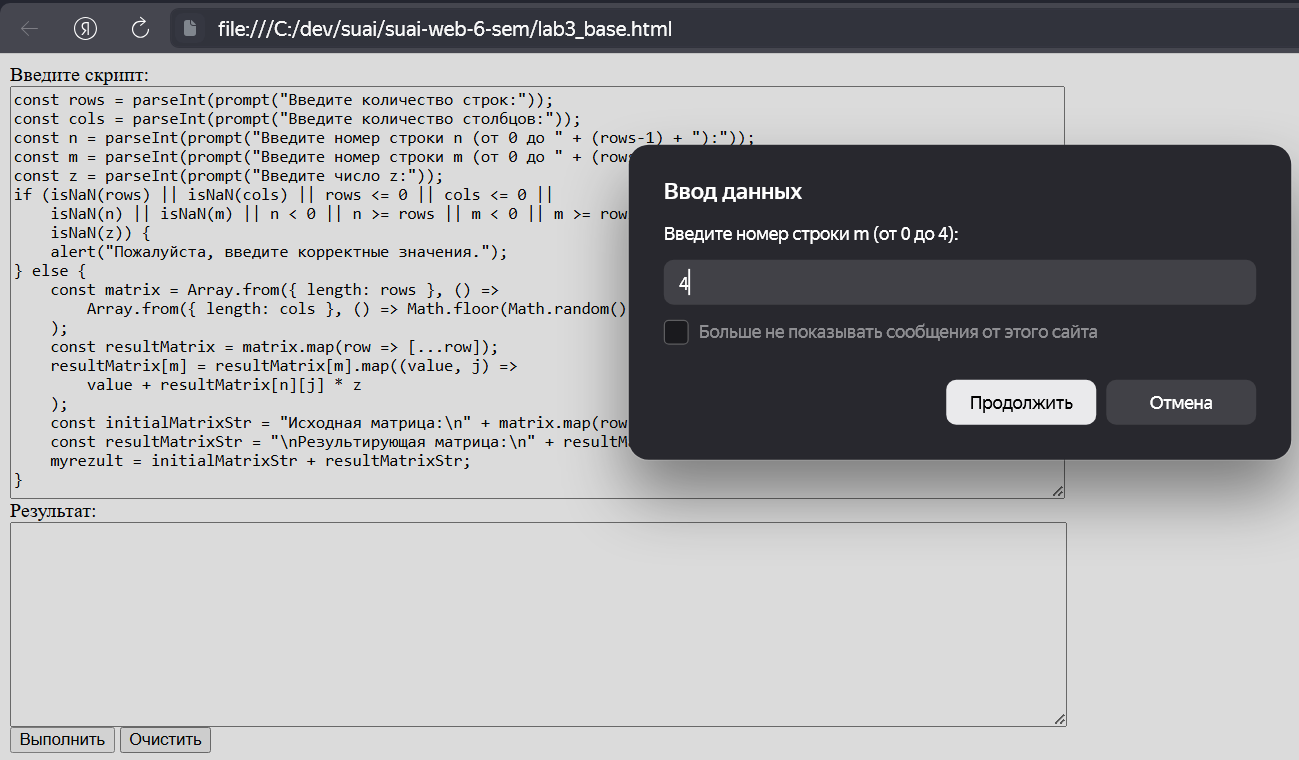


Рисунок 4 – запрос номера строки m

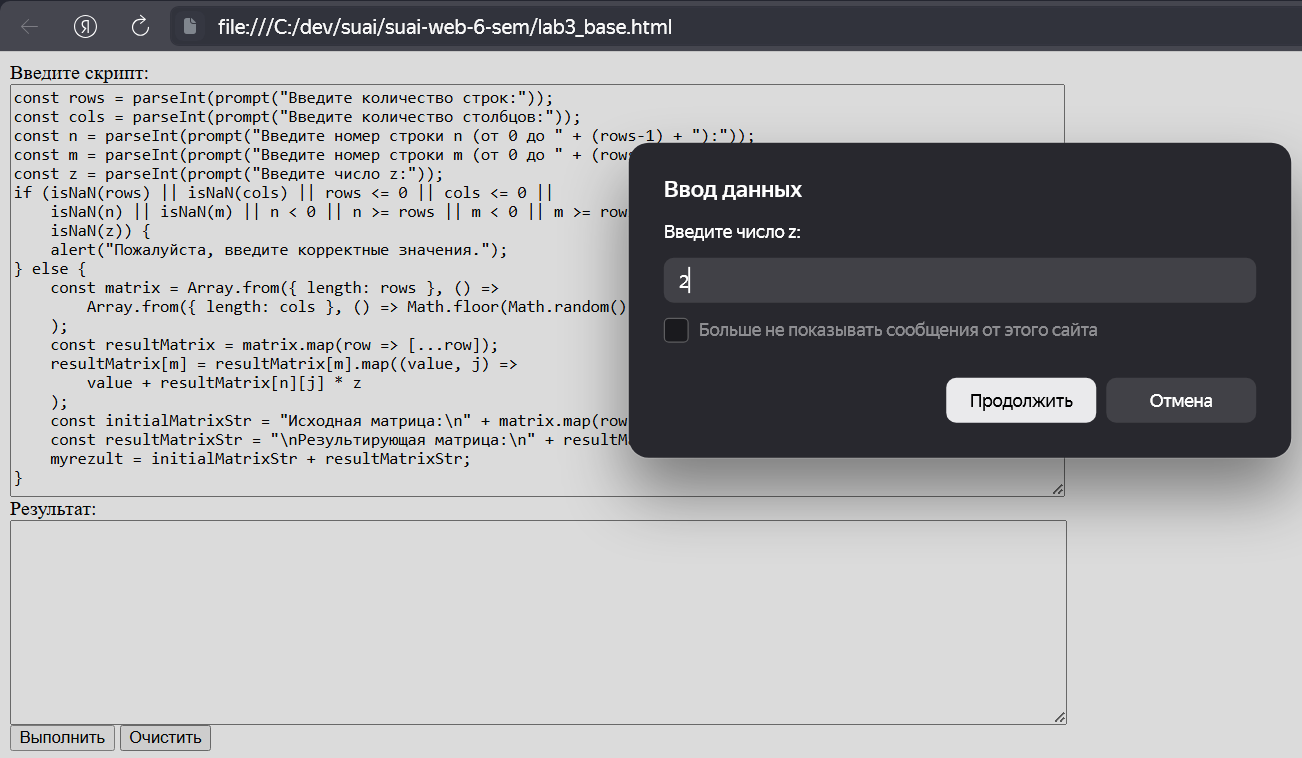


Рисунок 5 – запрос числа z

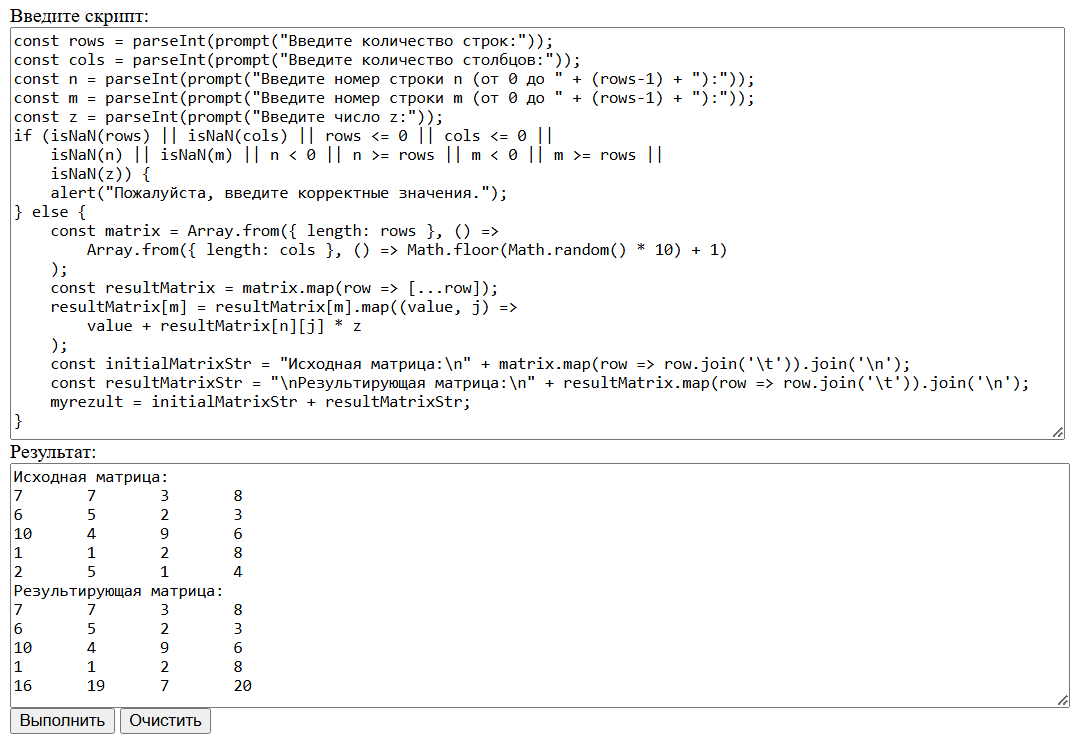


Рисунок 6 – успешный вывод матриц

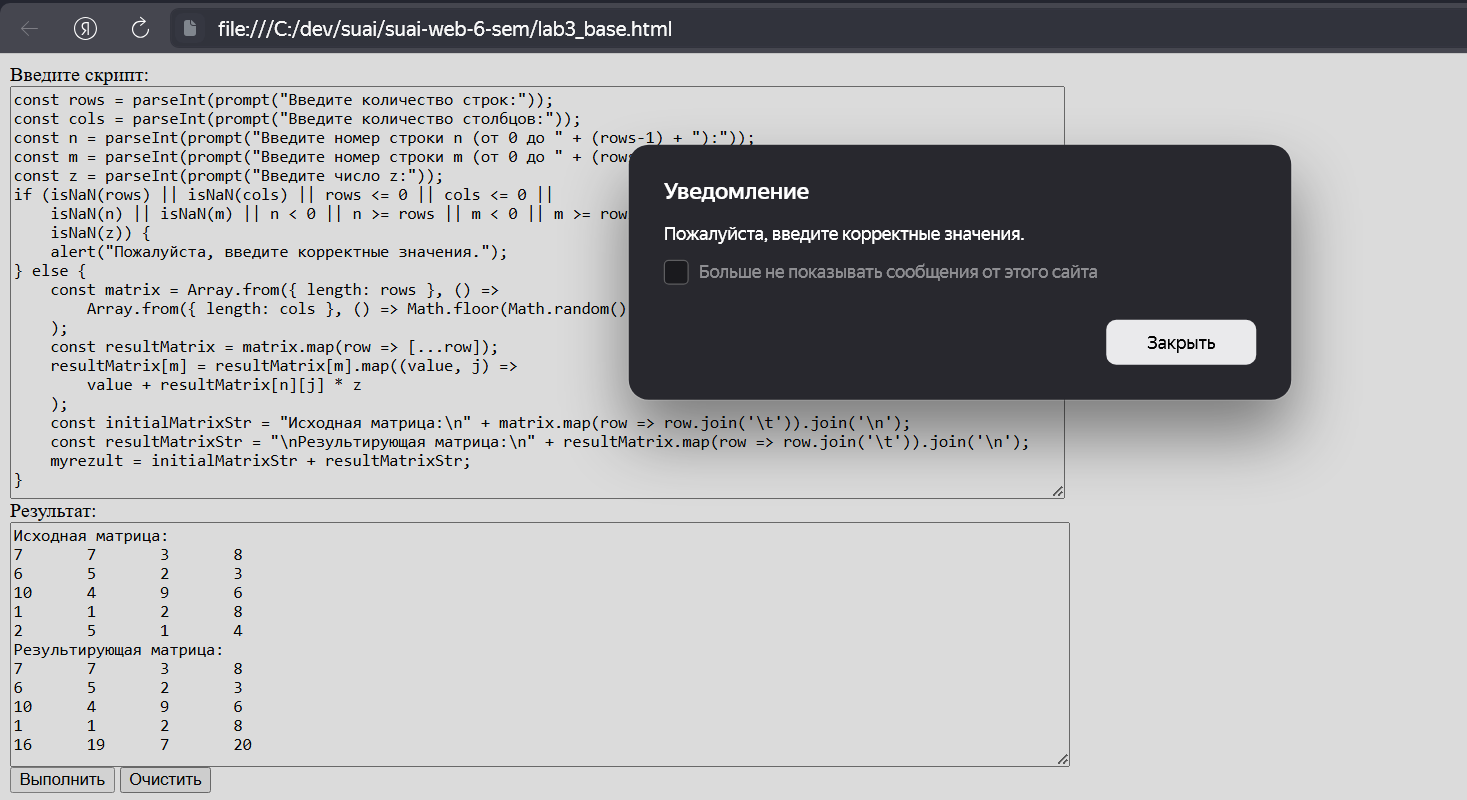


Рисунок 7 – вывод при ошибке ввода

На рисунке 8 представлен результат работы программы генерирующий рисунок с заданным в варианте паттерном. Программа рисует заданную фигуру на <canvas> с учетом введенных пользователем размеров. Циклы эффективно обрабатывают повторяющиеся элементы: четные строки заполняются полностью, а количество прямоугольников в нечетных строках увеличивается согласно формуле (i + 1) / 2.

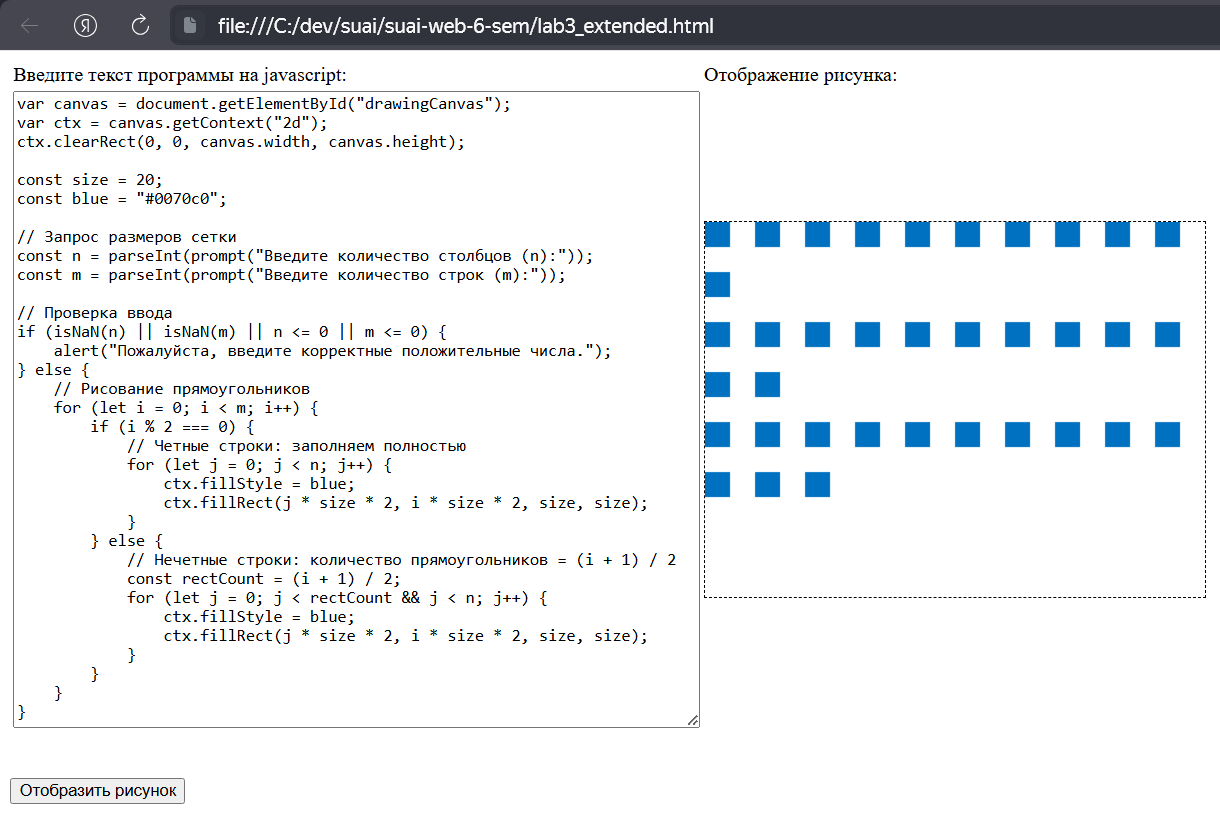


Рисунок 8 – вывод работы расширенного задания на страницу

На рисунках 8-9 показан запрос размеров матрицы для расширенного задания.

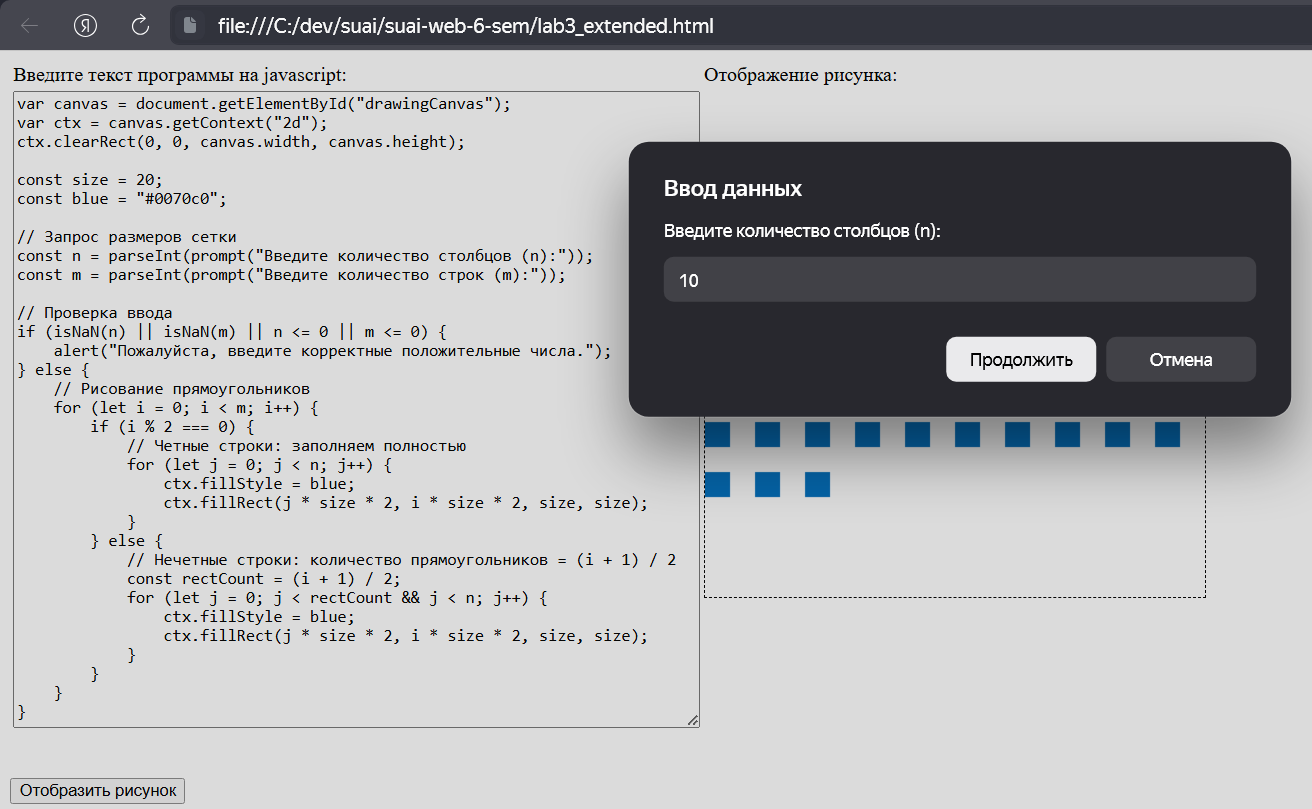


Рисунок 8 – запрос количества столбцов

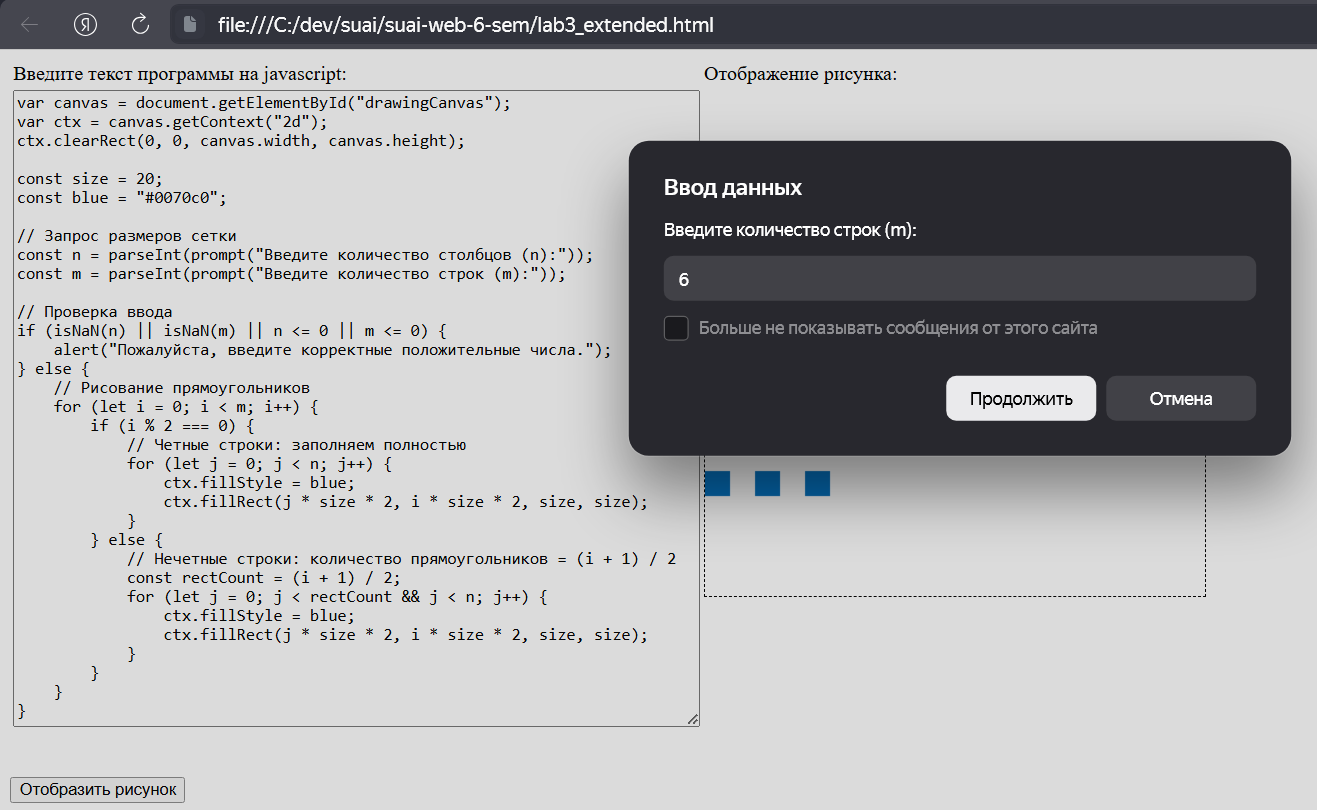


Рисунок 9 – запрос количества строк

# 6. Листинг программного кода

**Базовое задание:**

const rows = parseInt(prompt("Введите количество строк:"));

const cols = parseInt(prompt("Введите количество столбцов:"));

const n = parseInt(prompt("Введите номер строки n (от 0 до " + (rows-1) + "):"));

const m = parseInt(prompt("Введите номер строки m (от 0 до " + (rows-1) + "):"));

const z = parseInt(prompt("Введите число z:"));

if (isNaN(rows) || isNaN(cols) || rows <= 0 || cols <= 0 ||

isNaN(n) || isNaN(m) || n < 0 || n >= rows || m < 0 || m >= rows ||

isNaN(z)) {

alert("Пожалуйста, введите корректные значения.");

} else {

const matrix = Array.from({ length: rows }, () =>

Array.from({ length: cols }, () => Math.floor(Math.random() \* 10) + 1)

);

const resultMatrix = matrix.map(row => [...row]);

resultMatrix[m] = resultMatrix[m].map((value, j) =>

value + resultMatrix[n][j] \* z

);

const initialMatrixStr = "Исходная матрица:\n" + matrix.map(row => row.join('\t')).join('\n');

const resultMatrixStr = "\nРезультирующая матрица:\n" + resultMatrix.map(row => row.join('\t')).join('\n');

myrezult = initialMatrixStr + resultMatrixStr;

}

**Расширенное задание:**

var canvas = document.getElementById("drawingCanvas");

var ctx = canvas.getContext("2d");

ctx.clearRect(0, 0, canvas.width, canvas.height);

const size = 20;

const blue = "#0070c0";

// Запрос размеров сетки

const n = parseInt(prompt("Введите количество столбцов (n):"));

const m = parseInt(prompt("Введите количество строк (m):"));

// Проверка ввода

if (isNaN(n) || isNaN(m) || n <= 0 || m <= 0) {

alert("Пожалуйста, введите корректные положительные числа.");

} else {

// Рисование прямоугольников

for (let i = 0; i < m; i++) {

if (i % 2 === 0) {

// Четные строки: заполняем полностью

for (let j = 0; j < n; j++) {

ctx.fillStyle = blue;

ctx.fillRect(j \* size \* 2, i \* size \* 2, size, size);

}

} else {

// Нечетные строки: количество прямоугольников = (i + 1) / 2

const rectCount = (i + 1) / 2;

for (let j = 0; j < rectCount && j < n; j++) {

ctx.fillStyle = blue;

ctx.fillRect(j \* size \* 2, i \* size \* 2, size, size);

}

}

}

}

# 7. Выводы

В ходе выполнения лабораторной работы я изучил работу с языком JavaScript, включая создание и обработку двумерных массивов, а также использование <canvas> для визуализации данных. Программа успешно реализует операцию над матрицей, умножая строку с номером n на число z и прибавляя результат к строке с номером m, а затем выводит исходную и результирующую матрицы на страницу. Для расширенного задания применен объект <canvas> для построения фигуры из синих прямоугольников по заданному паттерну, с учетом данных, введенных пользователем. В работе использованы методы массивов Array.from(), map() и join(), а также стандартные методы для обработки ввода (parseInt, isNaN) и работы с DOM (getElementById, getContext).