МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

**Институт экономики и предпринимательства**

Кафедра информационных технологий

и инструментальных методов в экономике

отчет о прохождении практики по получению первичных профессиональных умений и навыков

Работу выполнил студент

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_

Подпись

\_\_\_\_\_\_\_\_\_\_\_

Дата

Группы\_\_3523Б7ПИ1\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_С.А.Блинов

(подпись)

Номер зачетной книжки\_\_\_\_\_

\_\_\_ 23350290\_\_\_\_\_\_\_\_

Руководитель:

|  |
| --- |
| Кандидат педагогических наук, |
|  |

Доцент

\_ \_\_ А.А. Беспалько

(подпись)

Н. Новгород, 2025 г.

## **ИНДИВИДУАЛЬНОЕ** **ЗАДАНИЕ** **НА** **ПРАКТИКУ**

Студент: Блинов Степан Андреевич

Факультет/институт/филиал: Институт Экономики

Форма обучения: Заочная

Направление/специальность: Прикладная информатика в экономике

Содержание задания на практику (перечень подлежащих рассмотрению вопросов):

1. За время практики студенты должны научиться писать простейшие программы на VBA:
2. макросы в Excel VBA,
3. программы в Excel VBA (желательно),
4. c использованием форм в стиле VB.net (C#, Qt5 и т.п.).

Задания и варианты решений находятся на сервере по адресу:

https://docs.google.com/spreadsheets/d/1kPb-uQMzkRDlYk44b6hlw0SdRB9dGZnEXocN79otmtI/edit?gid=0#gid=0

Надо найти свою фамилию в списке группы и посмотреть на номер.

Этот номер является номером задачи, которую надо решить.

Тексты задач находятся в файлe и приведены в приложении 2:

https://drive.google.com/drive/folders/1YDyhdULPKo5AIeZRF3AD3B0AeicLrXho

Защита с оценкой. Отчет по вычислительной практике должен быть напечатан. Объем его от 6 до 12 страниц Тексты программ принято печатать моноширинным шрифтом, например, шрифтом "Courier New Cyr". Программы должны быть структурными, внимательно следите за отступами! На отчете должна стоять одна фамилия студента, отчет должен иметь уникальное название.

Дата выдачи задания: 3 февраля 2025 года

Руководитель практики от факультета/

института/филиала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

личная подпись И.О. Фамилия

Ознакомлен

Студент группы \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

личная подпись И.О. Фамилия

**Оглавление**

[ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРАКТИКУ 2](#_Toc208675391)

[ВВЕДЕНИЕ 4](#_Toc208675392)

[1. Задача на обработку одномерных массивов 7](#_Toc208675393)

[2. Задача на обработку числовых рядов 12](#_Toc208675394)

[3. Задача на обработку двумерных массивов. 16](#_Toc208675395)

[Заключение 20](#_Toc208675396)

# **ВВЕДЕНИЕ**

Ознакомительная практика является важным этапом в процессе подготовки будущего специалиста в области информационных технологий. Она позволяет применить теоретические знания, полученные в ходе обучения, для решения практических задач, а также приобрести ценный опыт разработки программного обеспечения. В рамках данной практики была поставлена задача разработки комплекса математических веб-приложений, включающего три основных модуля: Обработка одномерных массивов, Числовые ряды и Обработка двумерных массивов.

Обоснование выбора технологий:

Для реализации проекта был выбран комплекс технологий на основе JavaScript, HTML и CSS, что обусловлено следующими факторами:

1. Кросс-платформенность и доступность

JavaScript является языком программирования, который выполняется на стороне клиента в любом современном веб-браузере. Это обеспечивает:

* Нулевую стоимость развертывания - не требуется установка дополнительного ПО
* Кроссплатформенную совместимость - приложения работают на Windows, macOS, Linux, iOS и Android
* Мгновенную доступность - пользователи могут начать работу без скачивания и установки

2. Интерактивность и отзывчивый интерфейс

Современный JavaScript в сочетании с HTML5 и CSS3 позволяет создавать:

* Высокую степень интерактивности - динамическое обновление контента без перезагрузки страницы
* Богатый пользовательский опыт - анимации, плавные переходы, мгновенная валидация данных
* Адаптивный дизайн - автоматическое подстраивание интерфейса под различные устройства

3. Математические вычисления на клиентской стороне

JavaScript обеспечивает:

* Выполнение сложных вычислений непосредственно в браузере пользователя
* Мгновенную обратную связь - результаты отображаются сразу после ввода данных
* Снижение нагрузки на сервер - все вычисления производятся локально

4. Современные возможности языка

ES6+ предоставляет мощные возможности для математических вычислений:

* Поддержка математических операций через объект Math
* Стрелочные функции для компактной записи алгоритмов
* Деструктуризация для удобной работы с массивами и объектами
* Шаблонные строки для форматированного вывода результатов

5. Простота разработки и отладки

Комбинация HTML/CSS/JS предлагает:

* Быстрый процесс разработки - мгновенное обновление изменений в браузере
* Мощные инструменты отладки - встроенные Developer Tools в браузерах
* Простота тестирования - возможность быстрой проверки функционала

6. Образовательная ценность

Выбор веб-технологий особенно актуален потому, что:

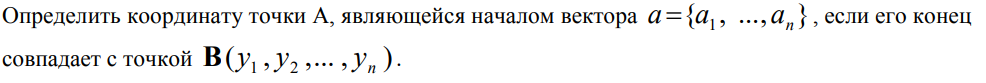
* Соответствует современным трендам веб-разработки
* Развивает универсальные навыки востребованные на рынке труда
* Позволяет демонстрировать результаты широкой аудитории без ограничений

Выбор JavaScript в сочетании с HTML и CSS для реализации математических приложений оказался оптимальным решением, сочетающим в себе современные подходы к разработке, кросс-платформенность и высокую производительность. Данный технологический стек позволил создать интуитивно понятные и функциональные инструменты для математических вычислений, доступные любому пользователю с современным веб-браузером.

Разработанный комплекс приложений демонстрирует практическое применение веб-технологий для решения образовательных задач и служит наглядным примером возможностей клиентской веб-разработки в области математического моделирования и визуализации.

# **Задача на обработку одномерных массивов**

Первая задача в рамках практики по получению первичных профессиональных умений и навыков связана с обработкой одномерных массивов (Изображение 1).



Изображение 1 – Постановка задачи.

В рамках выполнения поставленной работы была представлена математическая модель задачи:

пусть A=(x1,…,xn), вектор a⃗=(a1,…,an) направлен от A к B, и конец вектора совпадает с B=(y1,…,yn). По определению вектора.



Отсюда для каждой координаты:



Итого координаты точки А:



Интерфейс программы принимает входные данные в виде двух строк: компонент вектора 𝑎 (через запятую) и координат точки 𝐵 также через запятую), и выводит координаты 𝐴.

При инициализации кода (Изображение – 1) создаются ссылки на элементы DOM, необходимые для взаимодействия с пользователем. Каждой константе присвоена роль:

vecAInput — ссылка на поле ввода для компонентов вектора 𝑎;

pointBInput — ссылка на поле ввода для координат точки 𝐵;

calcBtn — ссылка на кнопку «Вычислить», событие которой запускает вычисления;

clearBtn — ссылка на кнопку «Очистить», реализующую сброс формы;

outputDiv — контейнер для отображения результата или сообщений об ошибках.

Для надёжной интерпретации текстового ввода внутри программы предусмотрена вспомогательная функция parseList(s) (Изображение – 1). Функция получает строку s и возвращает либо массив чисел, либо null при некорректном вводе. Обработка выполняется в несколько последовательных шагов: сначала проверяется тип аргумента (строка), затем строка разбивается методом split(',') по запятым, каждому элементу удаляются внешние пробелы (trim()), пустые элементы исключаются (filter(p => p.length > 0)). После этого каждое текстовое представление преобразуется в число посредством метода Number(p). Полученные значения проверяются с помощью метода Number.isFinite: только конечные числовые значения допускаются; при появлении NaN либо бесконечности функция возвращает null. Такая валидация предотвращает дальнейшие вычисления на некорректных данных и исключает опасные состояния (например, наличие бесконечности).

Обработчики событий привязаны к двум интерактивным элементам. Обработчик clearBtn (Изображение 3) прост и выполняет утилиту удобства: очищает оба поля ввода, восстанавливает стартовое информационное сообщение в outputDiv и переводит фокус в поле ввода вектора. Это повышает эргономику работы пользователя, но в вычислительной сущности задачи роли не играет.

Основной функциональный блок реализован в обработчике события calcBtn.addEventListener('click') (Изображение 2). Именно здесь и выполняется математическая трансформация 𝐴 = 𝐵 − 𝑎⃗ и сопроводительные проверки. Его пошаговая логика подробно представлена ниже.

Сначала производится извлечение входных строк: aStr = vecAInput.value и bStr = pointBInput.value. Далее эти строки передаются в parseList; результатом должны стать массивы чисел a и b. Возврат null из parseList интерпретируется как некорректный ввод, и в этом случае функция прерывает выполнение, формируя сообщение об ошибке в outputDiv. Это является первой ступенью валидации (синтаксическая и семантическая проверка элементов).

Если оба массива успешно разобраны, проверяется их соответствие по длине: a.length === b.length. Это критическое семантическое требование (предусловие алгоритма), поскольку операция покоординатного вычитания определена только при равной размерности. Несоблюдение этого требования приводит к отказу, предотвращающему некорректные доступы к элементам массива.

При выполнении предусловий происходит собственно вычисление координат точки. Это реализовано в выразительном виде с использованием метода Array.prototype.map.

Здесь для каждого индекса i вычисляется разность val = yi - ai, что соответствует аналитической формуле xi=yi−aix\_i = y\_i - a\_ixi​=yi​−ai​. Далее применены два практико-инженерных приёма, направленных на улучшение представления результата и уменьшение визуальных артефактов численных вычислений.

Первый приём — детекция отрицательного нуля с помощью Object.is(val, -0). В JavaScript существует различие между значениями -0 и +0, которое не выявляется при стандартном сравнении ===, но фиксируется функцией Object.is. Наличие -0 в выводе может быть нежелательным (пользовательскому интерфейсу это затрудняет чтение и вызывает излишнее смущение), поэтому код явно заменяет -0 на 0.

Второй приём — округление результата до фиксированного числа знаков после десятичной точки. Для этого используется техника «домножить — округлить — поделить»: (Math.round((val + Number.EPSILON) \* 1e12) / 1e12). Умножение на 1012 и последующее округление целесообразно, если требуется ограничить вывод погрешностей численного представления до 12 знаков после запятой. Добавление Number.EPSILON перед операцией округления — практическая мера, направленная на снижение эффекта двоичной погрешности при граничных случаях округления (это небольшой сдвиг порядка 10−16, значительно меньший чем масштаб округления). После получения округлённого значения выполняется приведение к строке и обратно в число (toString() затем Number(...)), что служит нормализации представления и устранению редких артефактов форматирования. В целом, эти операции носят форматно-представительный характер: они не меняют математического смысла операции xi=yi−aix\_i=y\_i-a\_ixi​=yi​−ai​, но повышают качество вывода и релевантность результатов.

После формирования массива A создаётся HTML-шаблон и присваивается outputDiv.innerHTML. Шаблон отображает размерность результирующего вектора и его компоненты в формате (x\_1, x\_2, ..., x\_n). Поскольку входные данные были предварительно проверены и приведены к числовому типу, риск инъекции опасного HTML через пользовательский ввод минимален; тем не менее для строгой безопасности предпочтительнее использовать методы DOM-ориентированной вставки текста (например, textContent или создание узлов document.createTextNode) вместо прямого innerHTML.

С точки зрения корректности и устойчивости, программа соблюдает ключевые позиции: входы либо корректно парсятся в массивы чисел равной длины и тогда вычисление выполняется; либо обнаруживается нарушение предусловий (некорректный формат, несовпадающая размерность, нечисловые значения, бесконечность) и формируется диагностическое сообщение. Это обеспечивает детерминированность поведения и удобство отладки.

Ниже приведён пример, иллюстрирующий работу программы на конкретных данных. При входе:

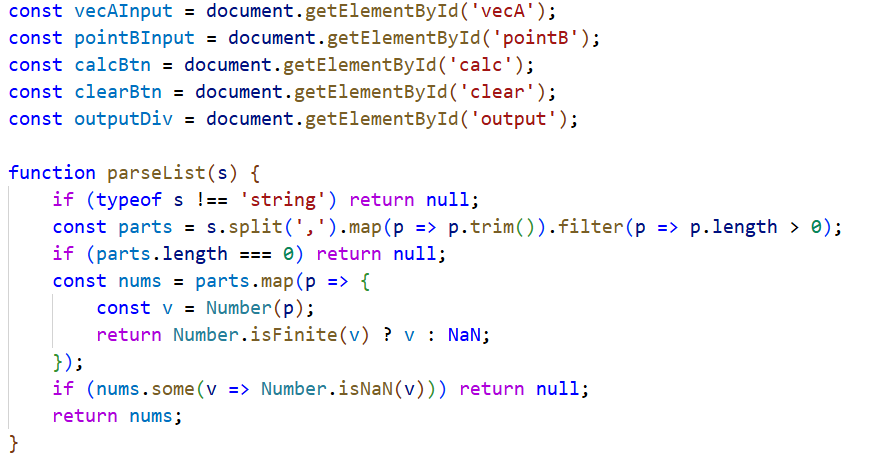
vecAInput.value = "2, 0, -3"

pointBInput.value = "5, 4, 1"

функция parseList вернёт массивы a = [2, 0, -3] и b = [5, 4, 1]. Основной блок вычислит по координатам xi=yi−ai: 5−2=3, 4−0=4, 1−(−3)=4. Результат отобразится как размерность 3 и координаты точки A=(3,4,4).

Основная вычислительная функция реализована компактно и корректно — она последовательно проверяет предпосылки (валидность ввода и конформность размерностей), выполняет покоординатное вычисление A=B−a⃗ с форматированием вывода и мерами по улучшению представления чисел (округление, устранение -0), и возвращает пользователю читаемый результат. Именно этот обработчик кнопки «Вычислить» является ядром программы, решающим математическую задачу, в то время как остальные элементы кода организуют ввод/вывод и пользовательскую эргономику.

Программное решение для данной задачи было описано следующим образом (Изображение 1 – Изображение 3):

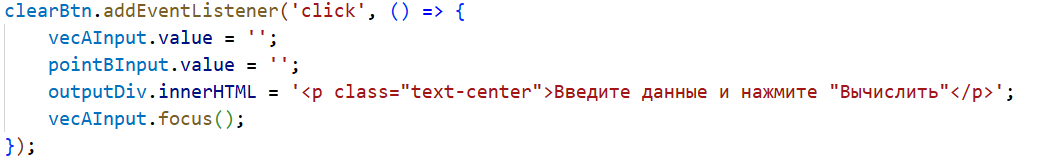


Изображение 1 – Объявление переменных, задействованных в программе,

метод перевода строки в числовой формат

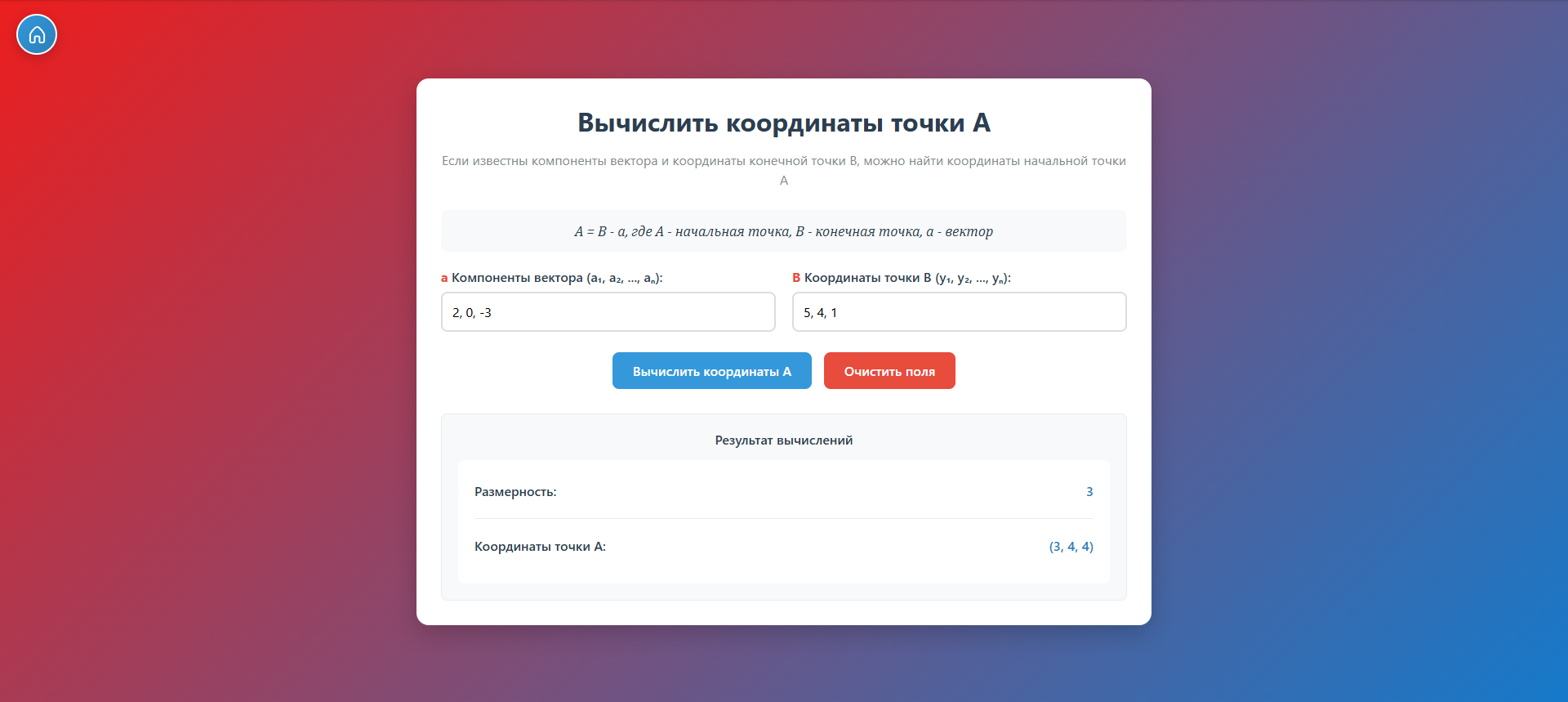


Изображение 2 – Основной программный метод, выполняющий вычисления

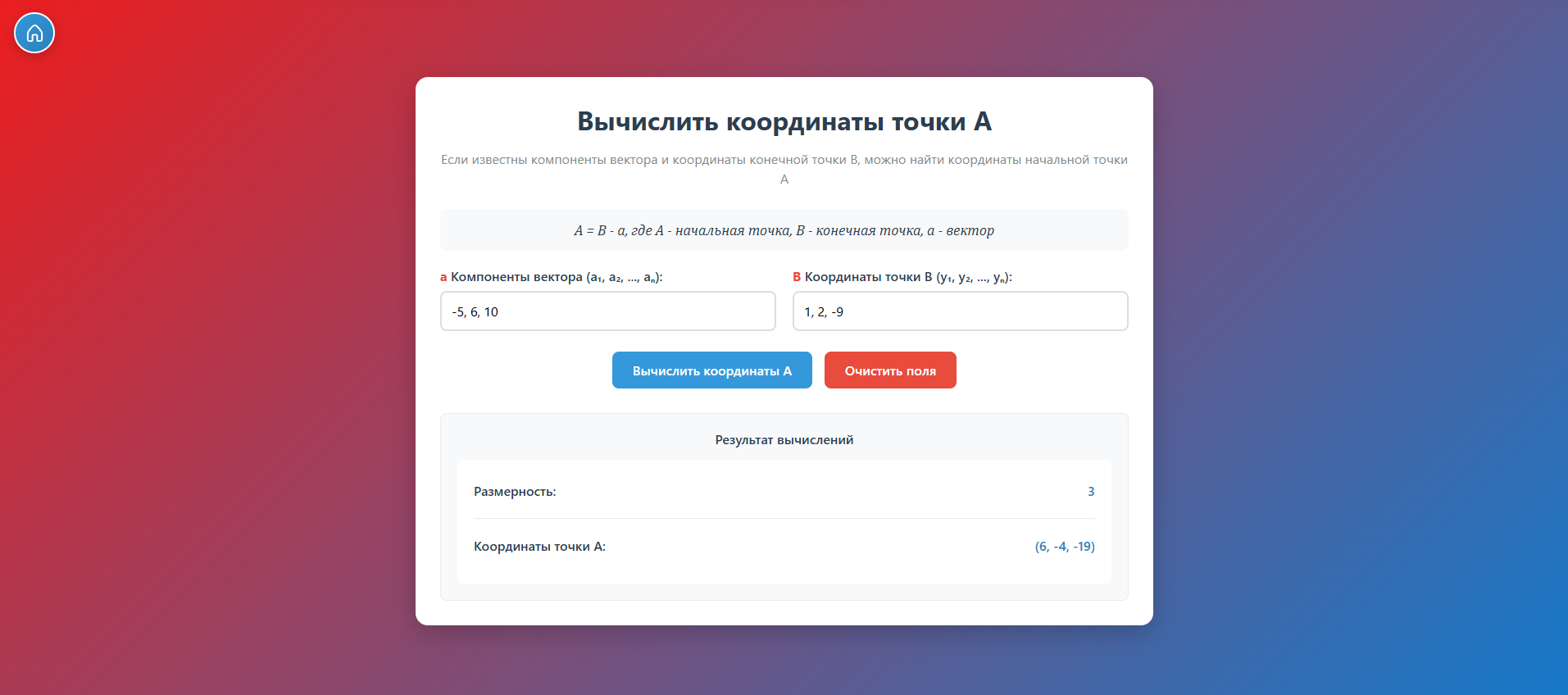


Изображение 3 – Метод очищения полей ввода

Примеры выполнения работы программой (Изображение 4 – Изображение 5)



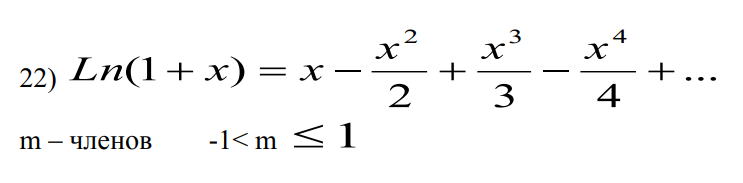
Изображение 4 – Пример работы программы ч. 1



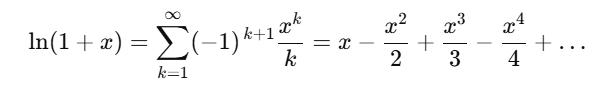
Изображение 5 – Пример работы программы ч. 2

# **Задача на обработку числовых рядов**

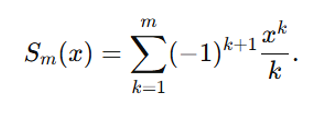
Вторая задача на обработку числовых рядов выглядит так:



Ряд Тейлора для ln(1+x) вокруг 0 (разложение в степенной ряд) имеет вид:



Этот ряд сходится при −1<x≤1 (при x=−1 расходится, при x=1 сходится условно к ln2). Частичная сумма первых m членов (то, что требуется вычислить) равна:



Оценка остаточного члена для чередующегося монотонно убывающего по модулю ряда: абсолютная погрешность ∣Rm∣=∣ln(1+x)−Sm(x)∣ не превосходит величины модуля следующего члена - ∣xm+1​/m+1​∣.

В представленной программе основной функциональный блок реализован внутри обработчика события нажатия на кнопку «Вычислить» (Изображение 6 – Изображение 8). Именно в этом месте сосредоточена ключевая вычислительная логика задачи.

Алгоритм начинается с получения исходных данных из элементов интерфейса: значения аргумента x и числа членов ряда m. Для обработки введённого пользователем значения аргумента применяется вспомогательная функция parseNumber, которая выполняет замену возможных десятичных запятых на точки, удаляет пробельные символы и преобразует строку в числовой формат. В случае некорректного ввода функция возвращает специальное значение NaN, что позволяет программе своевременно выявлять ошибки. Число членов ряда обрабатывается функцией Number, а затем преобразуется к целому с помощью Math.floor, что исключает возможность использования дробных значений.

Далее реализуется система проверки корректности входных данных. Если аргумент x не может быть интерпретирован как конечное число, если число членов ряда m не удовлетворяет условию m≥1, либо если аргумент не принадлежит области сходимости ряда (−1<x≤1), программа прекращает дальнейшие вычисления и выводит сообщение об ошибке. Такой механизм верификации данных обеспечивает устойчивость алгоритма и предотвращает выполнение некорректных операций.

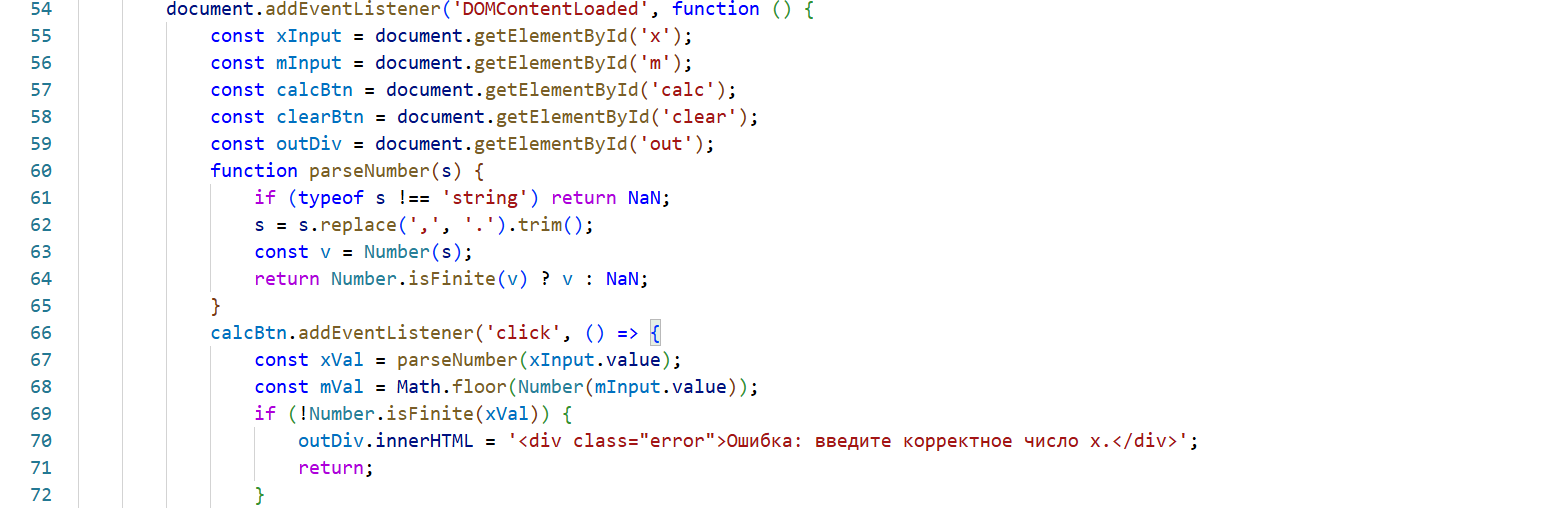
При успешном прохождении этапа проверки выполняется основное вычисление — суммирование частичных членов степенного ряда для функции ln(1+x). Используется цикл по индексу k от 1 до m, внутри которого вычисляется слагаемое xk/k​. Знак каждого слагаемого определяется закономерностью чередования: при нечётных k член прибавляется, при чётных — вычитается. В результате в переменной S накапливается частичная сумма ряда (Изображение 2):

Для последующего анализа точности полученного приближения вычисляется «эталонное» значение ln(1+x) с помощью встроенной функции Math.log1p, обладающей повышенной точностью при малых x. Абсолютная ошибка определяется как ∣ln(1+x)−Sm(x)∣, относительная — как отношение абсолютной ошибки к модулю точного значения. Дополнительно рассчитывается верхняя оценка остаточного члена ряда, равная по модулю следующему слагаемому ∣xm+1​/m+1​∣, что соответствует классической теореме о сходимости чередующихся рядов (признак Лейбница).

Завершающий этап алгоритма связан с формированием отчётной информации. Вычисленные величины форматируются в удобный для восприятия вид: частичная сумма и точное значение выводятся с фиксированным количеством знаков после запятой, а ошибки и оценка остатка — в экспоненциальной форме. Сформированный фрагмент HTML-разметки вставляется в элемент интерфейса, предназначенный для отображения результатов.

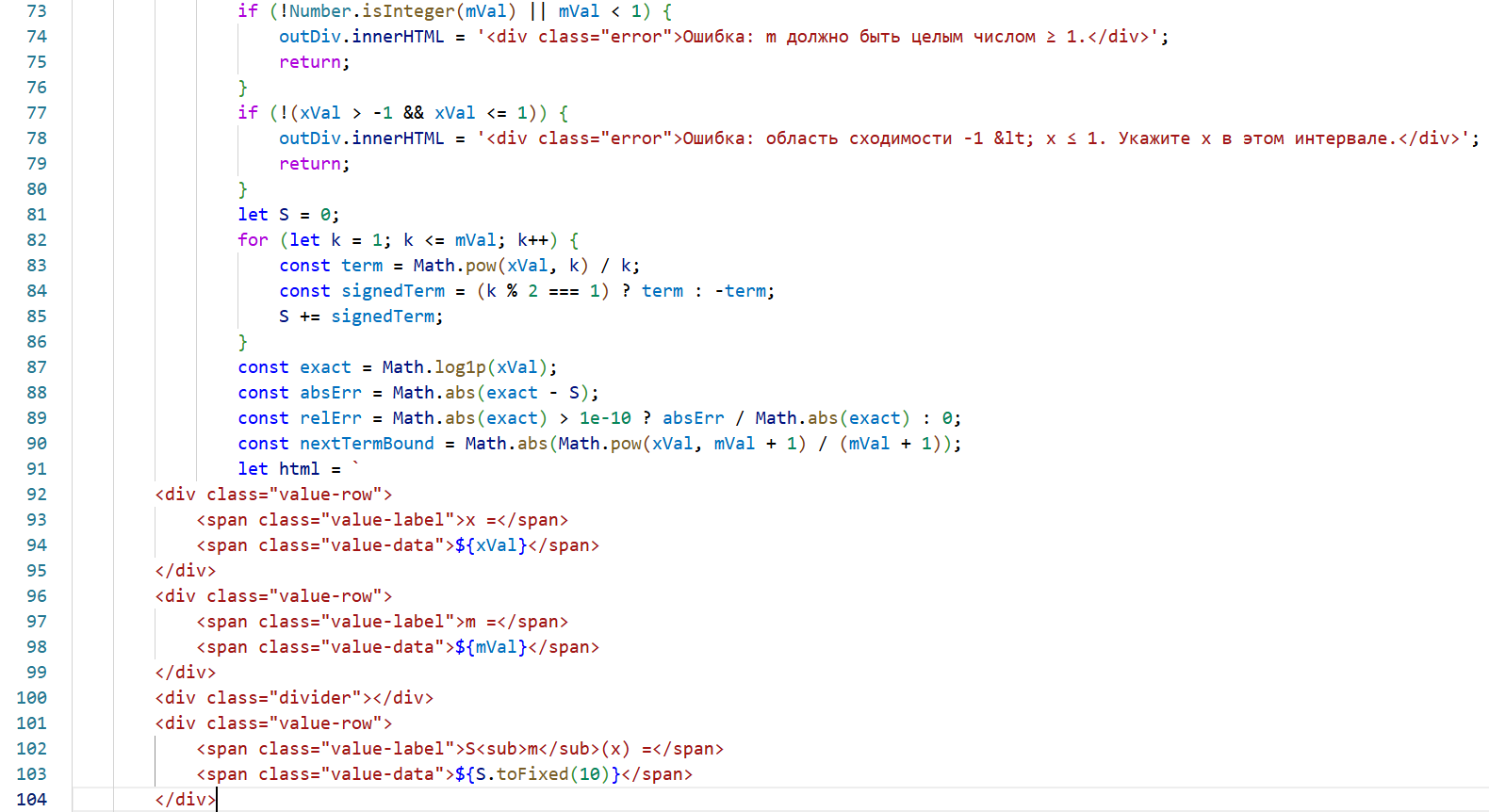
Таким образом, данный блок кода выполняет три взаимосвязанные функции: проверку корректности исходных данных, реализацию численного алгоритма приближённого вычисления натурального логарифма по его степенному разложению и представление результатов в структурированном и визуально наглядном виде. Именно здесь сосредоточено решение основной вычислительной задачи программы.

Программное решение задачи на обработку числовых рядов было представлено следующим образом (Изображение 6 – Изображение 8):



Изображение 6 – Объявление переменных, задействованных в программе,

метод перевода строки в числовой формат, начало основного метода программы

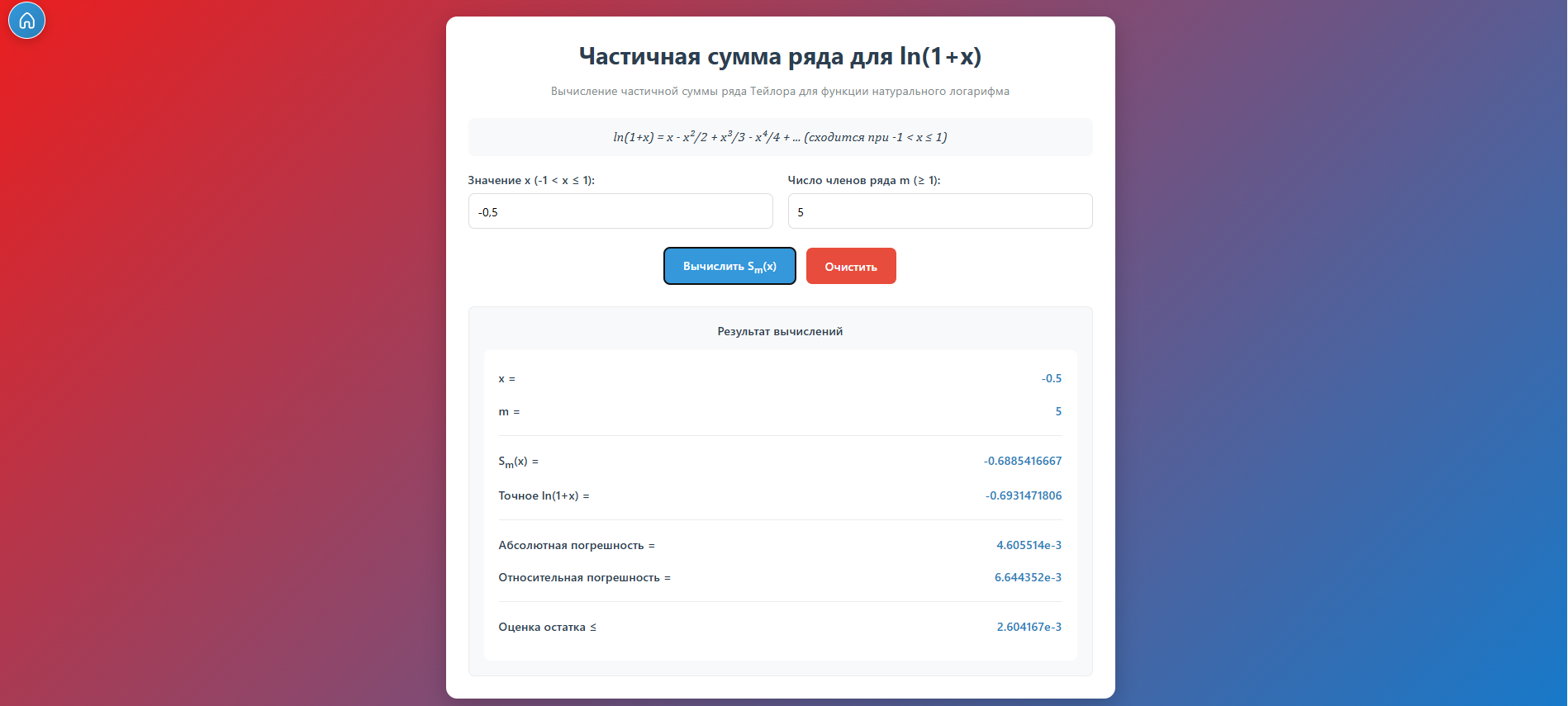


Изображение 7 – Продолжение основного метода программы



Изображение 8 – Конец основного метода программы

Примеры выполнения работы программой (Изображение 9):

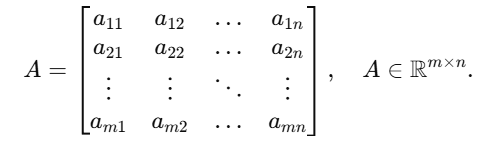


Изображение 9 – Пример работы программы обработки числовых рядов

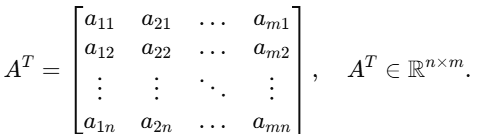
# **Задача на обработку двумерных массивов.**

Третья задача на обработку двумерных массивов звучит так: Дано матрица А размерности m\*n. Найти транспонированную матрицу.

Пусть дана матрица:



Транспонированная матрица AT получается заменой строк на столбцы:



То есть выполняется (AT)ij = Aij

Функция createMatrixInput (Изображение10) отвечает за динамическую генерацию формы ввода матрицы в зависимости от текущих значений rowsInput.value и colsInput.value. Значения преобразуются через parseInt, затем выполняется проверка: m и n должны быть положительны и не превышать 10. Предел 10×10 введён явно как интерфейсное ограничение, чтобы избежать избыточной нагрузки на страницу и сохранить удобство ввода. Если параметры проходят валидацию, цикл по строкам и вложенный цикл по столбцам конструируют HTML-строку, содержащую набор <input type="number"> с id в формате cell-i-j. Присвоение matrixA.innerHTML = html заменяет содержимое контейнера, тем самым создаёт нужные поля ввода. В противном случае (некорректная размерность) отображается диагностическое сообщение об ошибке.

Отслеживание изменения размерности реализовано через обработчики input на полях rowsInput и colsInput. Это даёт интерактивное поведение: при изменении одной из величин форма перестраивается автоматически. Изначальный вызов createMatrixInput() при инициализации обеспечивает создание полей по умолчанию при загрузке страницы.

Функция fillRandomly (Изображение 11) служит вспомогательным инструментом для оперативного тестирования и заполнения матрицы случайными целыми значениями в диапазоне [−10,10]. Реализация Math.floor(Math.random() \* 21) - 10 генерирует равномерные целые значения от 0 до 20, сдвинутые на −10, что даёт требуемый диапазон. Для каждой ячейки извлекается элемент по id cell-i-j и, если элемент существует, его value присваивается randomValue. После заполнения в контейнерах originalMatrix и transposedMatrix выводятся информационные сообщения о том, что матрица заполнена случайными значениями.

Основной вычислительный модуль реализован в функции transposeMatrix. Именно в этом блоке сосредоточена основная математическая логика приложения: чтение пользовательской матрицы из формы, проверка корректности, построение внутреннего представления в виде двумерного массива, вычисление транспонированной матрицы и вывод результатов. Процедура начинается с извлечения текущих значений m и n через parseInt. Затем создаётся пустой массив matrix и булева переменная hasError, используемая для сигнализации о нарушении предпосылок (например, если ожидаемая ячейка отсутствует в DOM).

Сбор значений выполняется двойным вложенным циклом: по i = 0..m−1 создаётся matrix[i] = [], затем по j = 0..n−1 извлекается DOM-элемент cell-i-j. Если элемент не найден, это считается ошибкой (возможна рассогласованность DOM и заданных размерностей) и цикл прерывается. Для каждого найденного элемента значение читается функцией parseFloat(cell.value) || 0. Такая запись приводит к следующему поведению: если parseFloat возвращает корректное число, оно используется; если возвращается NaN (например, пустая строка или нечисловая последовательность), результат логического || 0 заменяет его на 0. Это интуитивный, но потенциально замаскировывающий ошибку механизм: пустые или некорректные поля интерпретируются как нули, вместо того чтобы информировать пользователя об ошибке ввода. При обнаружении ошибки (отсутствие DOM элемента) функция завершает работу, выводя диагностическое сообщение в оба контейнера.

Если сбор матрицы прошёл успешно, первоначально вызывается displayMatrix(matrix, originalMatrix, 'Исходная матрица не создана'), что визуализирует исходную матрицу в блоке originalMatrix. Функция displayMatrix проверяет непустоту входного массива и затем формирует HTML-представление матрицы: для каждой строки создаётся контейнер <div class="matrix-value-row">, внутри которого значения каждой ячейки оборачиваются в <div class="matrix-value">. После этого HTML присваивается соответствующему контейнеру.

Выработка транспонированной матрицы реализована очевидным и корректным алгоритмом (Изображение 12 – Изображение 13): создаётся пустой массив transposed и два вложенных цикла, где внешним индексом перебираются столбцы исходной матрицы (i = 0..n−1), а внутренним — строки (j = 0..m−1), и выполняется присвоение transposed[i][j] = matrix[j][i]. Это прямое следование определению транспонирования​. После формирования транспонированной матрицы вызывается displayMatrix(transposed, transposedMatrix, 'Транспонированная матрица не создана') для отображения результата.

С точки зрения корректности вычислений, алгоритм выполняет покомпонентное присваивание без каких-либо численных преобразований (значения берутся как числа с плавающей точкой parseFloat).

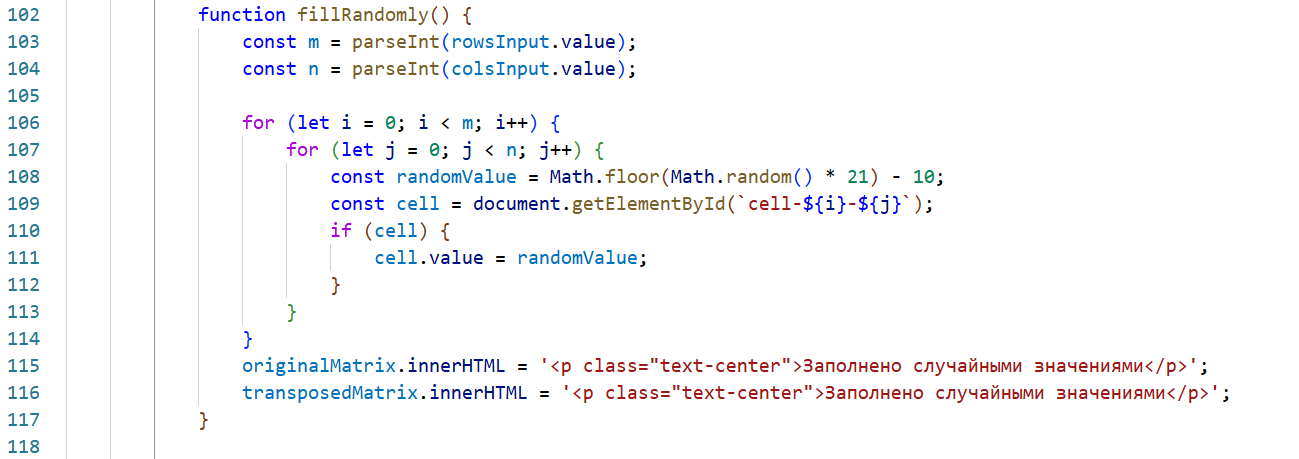
В заключение, архитектура программы чётко разделяет интерфейс и вычислительную логику: динамическая генерация полей ввода и вспомогательные операции (заполнение случайными числами) упрощают экспериментирование, а функция transposeMatrix концентрирует основную математическую задачу — чтение матрицы, проверка предпосылок, построение внутреннего двумерного массива и непосредственное вычисление транспонированной матрицы с выводом результатов.

Программное решение для данной задачи было описано следующим образом (Изображение 10 – Изображение 13):



Изображение 10 – Объявление переменных, вызов методов

и описание функции генерирования матрицы



Изображение 11 – Функция произвольного машинного заполнения матрицы



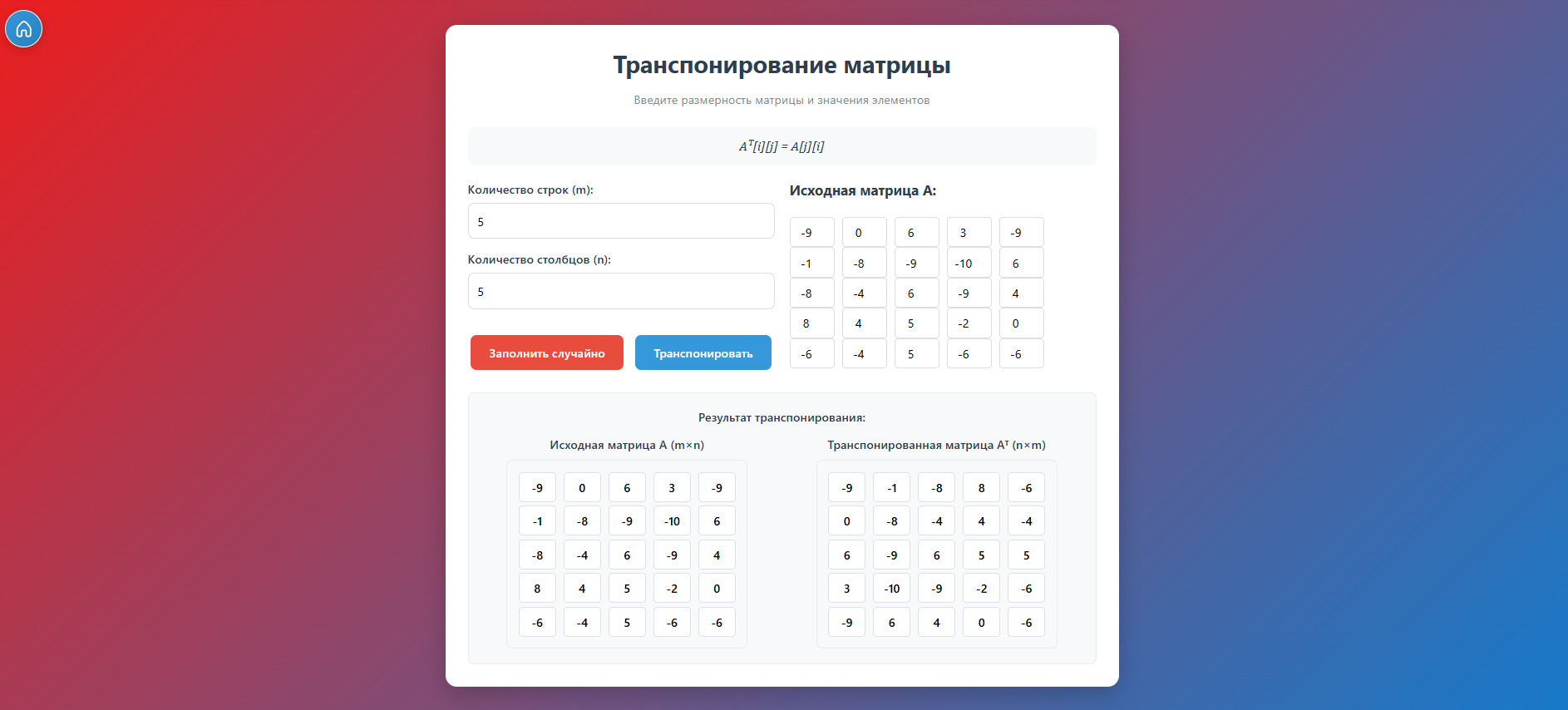
Изображение 12 – Начало функции транспонирования матрицы



Изображение 13 – Конец функции транспонирования матрицы

и функция отображения результата транспонирования

Пример выполнения работы программой (Изображение 14):



Изображение 14 – Пример работы программы транспонирования матрицы

# **Заключение**

В ходе ознакомительной практики была успешно реализована поставленная задача по созданию комплекса математических веб-приложений, включающего три основных модуля: обработку одномерных массивов, вычисление частичных сумм числовых рядов и обработку двумерных массивов (матриц). Каждое из приложений было спроектировано и реализовано с использованием современных веб-технологий — JavaScript, HTML и CSS, что позволило обеспечить кроссплатформенность, интерактивность и удобство использования.

В процессе разработки были решены следующие задачи:

* создано приложение для обработки одномерных массивов, реализующее вычисление координат точки A по заданному вектору и точке B;
* разработан модуль для работы с числовыми рядами, позволяющий вычислять частичную сумму степенного ряда для функции ln(1+x), а также определять абсолютную и относительную погрешности приближения;
* реализован модуль для работы с матрицами, обеспечивающий формирование исходной матрицы, её случайное заполнение и вычисление транспонированной версии.

Особое внимание уделялось проверке корректности пользовательского ввода и обработке возможных ошибок, что сделало интерфейс приложений более надёжным и удобным. Кроме того, были реализованы механизмы форматированного вывода результатов, позволяющие пользователю получать наглядную и структурированную информацию.

Реализация проекта позволила закрепить и расширить практические навыки работы с веб-технологиями, а также продемонстрировала возможности JavaScript для выполнения математических вычислений непосредственно в браузере пользователя. Таким образом, цели и задачи, поставленные в рамках ознакомительной практики, были достигнуты полностью.

Разработанный комплекс приложений может быть использован как в учебных целях — для наглядной демонстрации работы с математическими объектами, так и в качестве основы для дальнейших разработок в области интерактивного математического моделирования и визуализации данных.

Результат проделанной работы можно найти и опробовать на веб-ресурсе: <https://dranikovich.github.io/PracticeBlinov/>