

Разработка и реализация класса Matrix на TypeScript

Матвеева Дарья Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье рассматривается разработка класса Matrix на языке TypeScript для выполнения операций над матрицами. Описана структура класса, реализация сложения, вычитания, умножения, транспонирования, вычисления определителя и ранга. Описаны особенности статической типизации TypeScript, обеспечивающие контроль типов на этапе компиляции. Разработанный класс может быть использован в учебных целях, а также как компонент для вычислительных задач и обработки данных в веб-приложениях.

Ключевые слова: TypeScript, класс Matrix, матричные операции.

Development and implementation of the Matrix class in TypeScript

Matveeva Darya Sergeevna

Sholom-Aleichem Priamursky State University

Student

Abstract

This article discusses the development of a Matrix class in TypeScript for performing operations on matrices. It describes the class structure, the implementation of addition, subtraction, multiplication, transposition, and the calculation of the determinant and rank. The article also highlights the features of TypeScript's static typing, which provides type control during the compilation phase. The developed class can be used for educational purposes, as well as as a component for computational tasks and data processing in web applications.

Keywords: TypeScript, Matrix class, matrix operations.

1 Введение

1.1 Актуальность

В современном образовательном процессе и прикладной веб-разработке часто возникает необходимость выполнения матричных операций (сложение, умножение, транспонирование). Однако стандартный JavaScript не обеспечивает контроль типов, что приводит к ошибкам в вычислениях на этапе выполнения. TypeScript решает эту проблему, а реализация класса Matrix позволяет многократно использовать проверенный код в учебных и реальных проектах.

1.2 Обзор исследований

И. В. Князев и А. В. Коптева анализировали постепенное внедрение проверки и вывода типов данных на языке TypeScript [1]. Е. С. Завадский писал о TypeScript как об основном инструменте для разработки интерфейсов [2]. Л. В. Маркова, Е. А. Корчевская и А. Н. Красоткина описали объектно-ориентированный подход к изучению вычислительных методов алгебры [3]. В. А. Куров и Е. С. Паньков рассмотрели реализацию класса *Matrix* на языке C++ [4]. И. В. Путевская представила педагогическую методику поэтапного освоения Web-программирования, включающую изучение типизации данных [5].

1.3 Цель исследования

Целью данной работы является разработка и реализация на TypeScript класса *Matrix*, обеспечивающего хранение матриц произвольного размера и выполнение базовых операций: сложение, вычитание, умножение, умножение на скаляр, транспонирование, вычисление определителя и ранга.

2 Материалы и методы

Для реализации использовалась среда Node.js версии 20 и TypeScript 5.0. Разработка велась в редакторе Visual Studio Code с включённым режимом строгой проверки типов. Класс создавался в соответствии с принципами объектно-ориентированного программирования: данные спрятаны в приватное поле, методы возвращают новые матрицы, не меняя старые. Для проектирования схемы взаимодействия методов класса применялся онлайн-редактор *draw.io*, обеспечивающий удобство визуализации архитектуры программных компонентов.

3 Результаты

На первом этапе была спроектирована структура класса. Внутри матрица хранится как приватное поле `private data: number[][]`. Конструктор класса принимает количество строк, количество столбцов и начальное значение (по умолчанию — ноль). Также реализован статический метод `fromArray`, который создаёт матрицу на основе двумерного массива, проверяя при этом, что все строки имеют одинаковую длину (рис. 1).

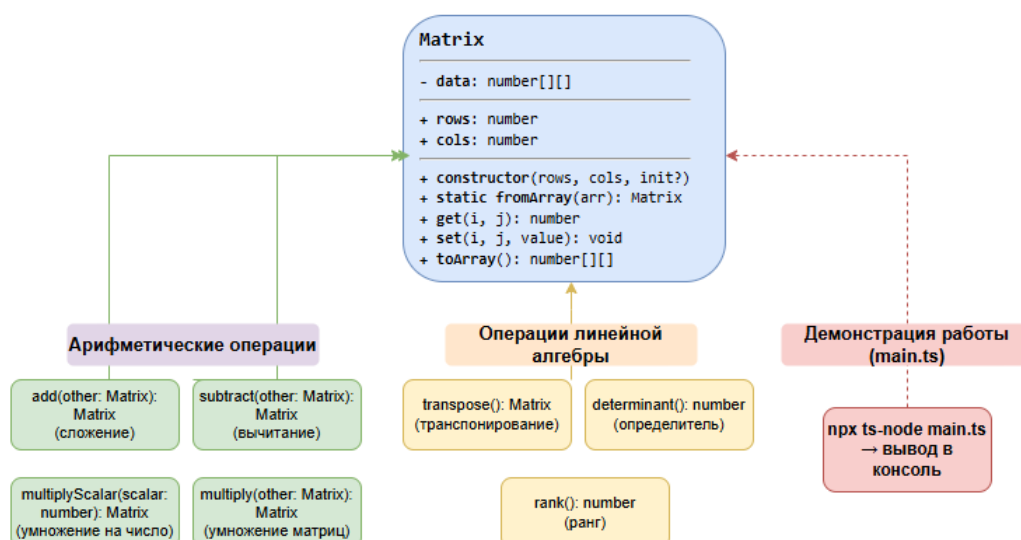


Рисунок 1 – Схема взаимодействия методов класса Matrix

Далее были последовательно реализованы арифметические операции. Метод `add` выполняет поэлементное сложение двух матриц, предварительно проверяя совпадение их размерностей. Если размерности не совпадают, выбрасывается ошибка. Аналогично реализовано вычитание `subtract`. Умножение на скаляр `multiplyScalar` не требует проверок и просто умножает каждый элемент матрицы на число. Операция умножения матриц `multiply`: требует, чтобы число столбцов первой матрицы равнялось числу строк второй (рис. 2). Результирующая матрица имеет размерность (строки_первой) × (столбцы_второй).

```

multiply(other: Matrix): Matrix {
  if (this.cols !== other.rows) {
    throw new Error(
      `Невозможно умножить: число столбцов первой матрицы (${this.cols})`
      `не равно числу строк второй матрицы (${other.rows})`
    );
  }
  const result = new Matrix(this.rows, other.cols);
  for (let i = 0; i < this.rows; i++) {
    for (let j = 0; j < other.cols; j++) {
      let sum = 0;
      for (let k = 0; k < this.cols; k++) {
        sum += this.get(i, k) * other.get(k, j);
      }
      result.set(i, j, sum);
    }
  }
  return result;
}
  
```

Рисунок 2 – Фрагмент кода метода умножения

Транспонирование реализовано как метод `transpose`, создающий новую матрицу, в которой элемент с индексами (i, j) исходной матрицы становится элементом (j, i) . Для квадратных матриц был добавлен метод `determinant`, вычисляющий определитель рекурсивно: для матрицы 1×1 возвращается единственный элемент, для 2×2 — разность произведений диагональных элементов, для больших размерностей используется разложение по первой строке с чередованием знаков. Перед вычислением проверяется, что матрица квадратная, иначе выбрасываю ошибку. Ранг матрицы вычисляется методом `rank` на основе приведения к ступенчатому виду: алгоритм последовательно обнуляет элементы под ведущими, подсчитывая количество ненулевых строк.

Все перечисленные методы возвращают новые объекты класса `Matrix`, не изменяя исходные данные. Для проверки создан файл `main.ts`, в котором созданы две матрицы 2×2 и последовательно вызываются все операции: сложение, умножение, транспонирование, вычисление определителя и ранга, а также умножение на скаляр и вычитание. (рис. 3).

```
import { Matrix } from './Matrix';|

const A = Matrix.fromArray([[1, 2], [3, 4]]);
const B = Matrix.fromArray([[5, 6], [7, 8]]);

console.log('=== Демонстрация работы класса Matrix ===\n');

console.log('Матрица A:');
console.table(A.toArray());

console.log('Матрица B:');
console.table(B.toArray());

console.log('Сложение A + B:');
console.table(A.add(B).toArray());

console.log('Умножение A * B:');
console.table(A.multiply(B).toArray());

console.log('Транспонирование A:');
console.table(A.transpose().toArray());

console.log('Определитель A:', A.determinant());
console.log('Ранг A:', A.rank());

console.log('Умножение A на скаляр 2:');
console.table(A.multiplyScalar(2).toArray());

console.log('Вычитание A - B:');
console.table(A.subtract(B).toArray());
```

Рисунок 3 – Листинг файла `main.ts`

После этого запущен `npm run ts-node main.ts` в терминале VS Code для проверки правильности вычислений (рис. 4).

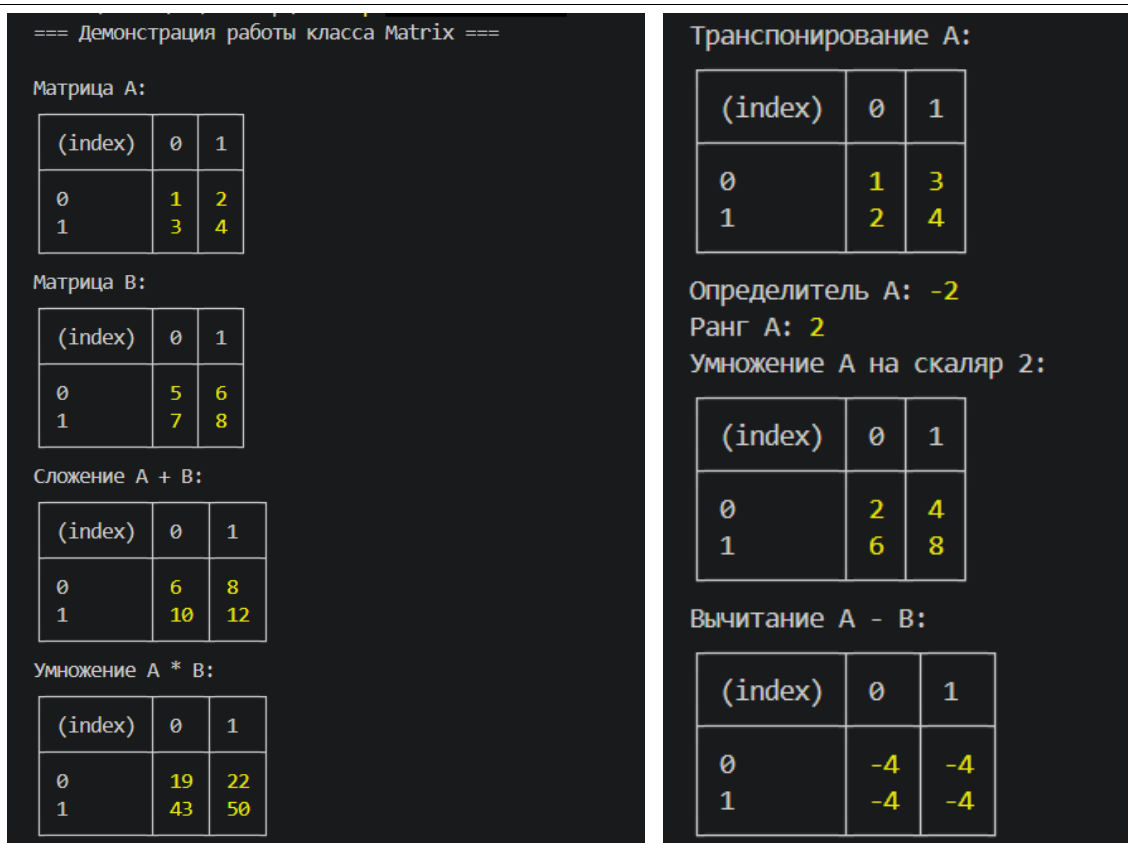


Рисунок 4 – Результат выполнения main.ts

4 Выводы

В ходе исследования спроектирован и реализован на TypeScript класс Matrix, поддерживающий операции сложения, вычитания, умножения, умножения на скаляр, транспонирования, вычисления определителя и ранга. Класс является типобезопасным, не изменяет исходные данные и обрабатывает ошибки. Разработанный класс может использоваться в учебных проектах и веб-приложениях, не требующих внешних зависимостей.

Библиографический список

1. Князев И. В., Коптева А. В. Разработка и анализ постепенного внедрения проверки и вывода типов данных с использованием языка программирования TypeScript и лямбда-исчисления. //European research. 2021. №. 1 (38). С. 6-9. URL: <https://cyberleninka.ru/article/n/razrabotka-i-analiz-postepennogo-vnedreniya-proverki-i-vyvoda-tipov-dannyh-s-pomoschyu-parametriceskogo-polimorfizma-i>
2. Завадский Е. С. TypeScript в современных веб-приложениях. // Репозиторий БГУИР. 2024 URL: https://libeldoc.bsuir.by/bitstream/123456789/57317/1/Zavadskij_TypeScript.pdf
3. Маркова Л. В., Корчевская Е. А., Красоткина А. Н. Объектная реализация методов вычислительной алгебры // Веснік Віцебскага дзяржаўнага

- університета. 2013. № 2(74). С. 18-22. URL: <https://www.elibrary.ru/item.asp?id=18960030>
4. Куров В. А., Паньков Е. С. Реализация класса Matrix для матричного калькулятора на языке программирования C+ //Проблемы и перспективы развития науки в России. 2016. С. 139. URL: <https://aeterna-ufa.ru/sbornik/NK-156-5.pdf#page=139>
5. Путевская И. В. Методика преподавания основ Web-программирования //Состояние, направления и перспективы развития среднего профессионального образования. 2019. С. 95-100. URL: <https://elibrary.ru/item.asp?id=37632791>