**федеральное государственное автономное образовательное**



**учреждение высшего образования**

**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**(ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ)**

**(Факультет информационных технологий)**

***(Институт Принтмедиа и информационных технологий) Кафедра Информатики и информационных технологий***

**направление подготовки 09.03.02 «Информационные системы и технологии»**

# ЛАБОРАТОРНАЯ РАБОТА № 3

**Дисциплина: Функциональное программирование.**

**Тема: Основы TypeScript для функционального программирования.**

**Выполнил(а): студент(ка) группы 221-3711**

**Мироненко Р. Е.**

## (Фамилия И.О.)

**Дата, подпись** **19.02.2025**  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

## **Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва2025**

**Цель:** Освоить основы функционального программирования в TypeScript, включая статическую типизацию и создание функций высшего порядка.

**Задание:**

1. **Разработайте набор чистых функций для работы с массивами:**
   * Функция, которая принимает массив чисел и возвращает новый массив, содержащий только числа, кратные заданному числу.
   * Функция, которая принимает массив строк и возвращает новую строку, содержащую все строки, объединенные заданным разделителем.
   * Функция, которая принимает массив объектов и возвращает новый массив, отсортированный по значению определенного свойства.
2. **Создайте функцию, которая принимает другую функцию в качестве аргумента и возвращает новую функцию, которая выполняет логирование перед вызовом исходной функции.**

**Требования:**

* Все функции должны быть чистыми и иметь явные аннотации типов.
* Используйте generics для повышения гибкости функций.
* Код должен быть хорошо оформлен и легко читаем.

**Ход работы:**

Гит: <https://github.com/Roman784/FuncProg.git>

Листинг 1 lab1\_main.js

|  |
| --- |
| // Интерфейсы.  *interface* User {      name: string;      age: number;  }  *interface* AnyFunc {      (...args: any[]): any;  }  // Функции.  // Функция, которая принимает массив чисел и возвращает новый массив, содержащий только числа, кратные заданному числу.  *function* filterByMultiplier(arr: number[], multiplier: number): number[] {      return arr.filter(num *=>* num % multiplier === 0);  }  // Функция, которая принимает массив строк и возвращает новую строку, содержащую все строки, объединенные заданным разделителем.  *function* joinStrings(arr: string[], separator: string): string {      return arr.join(separator);  }  // Функция, которая принимает массив объектов и возвращает новый массив, отсортированный по значению определенного свойства.  *function* sortByProperty<T>(arr: T[], compareFunc: (a: T, b: T) *=>* number): T[] {  *const* newArr: T[] = arr.slice();      return newArr.sort((a, b) *=>* compareFunc(a, b));  }  // Сравнение пользователей по возрасту.  *function* compareUsersByAge(a: User, b: User): number {      return a.age - b.age;  }  // Функция, которая принимает другую функцию в качестве аргумента и возвращает новую функцию, которая выполняет логирование перед вызовом исходной функции.  *function* logProxy<T extends AnyFunc>(func: T, logMessage: string): T {      return *function*(...args: Parameters<T>): ReturnType<T> {          console.log(logMessage);          return func(...args);      } as T;  }  // Демонстрация работы.  *const* numbers: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9];  *const* strings: string[] = ['Hello,', 'my', 'name', 'Roman'];  *const* users: User[] = [      { name: 'Man', age: 25},      { name: 'Roman', age: 20 },      { name: 'Vika', age: 17 },      { name: 'Yarik', age: 22 }  ];  console.log('Массив чисел: ' + numbers);  console.log('Массив строк: ' + strings)  console.log('Массив пользователей ' + JSON.stringify(users) + '\n');  console.log('Числа кратные 3: ' + filterByMultiplier(numbers, 3));  console.log('Объединённые строки: ' + joinStrings(strings, ' '));  console.log('Отсортированные по возрасту пользователи: ' + JSON.stringify(sortByProperty(users, compareUsersByAge)));  *const* loggedAdder = logProxy((a: number, b: number): number *=>* a + b, 'INFO: Суммирование');  console.log('Сумма 1 и 2: ');  console.log(loggedAdder(1, 2)); |

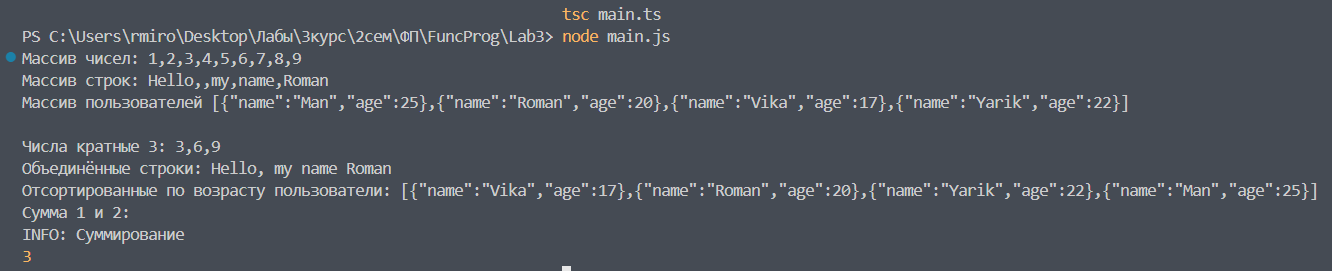


Рисунок 1 Результат работы в консоли