



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет Информационных технологий  
Кафедра Информатики и информационных технологий**

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

**ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ РАБОТА) № \_3\_**

**Дисциплина: Функциональное программирование**

**Тема:** \_\_\_\_\_ Основы TypeScript для функционального  
программирования \_\_\_\_\_

**Вариант**

**Выполнил(а): студент(ка) группы 221-374**

**Максимов Юрий Сергеевич**  
(Фамилия И.О.)

**Дата, подпись** 24.02.2025 \_\_\_\_\_  
(Дата) (Подпись)

**Проверил:** \_\_\_\_\_  
(Фамилия И.О., степень, звание) (Оценка)

**Дата, подпись** \_\_\_\_\_  
(Дата) (Подпись)

**Москва  
2025**

## Задание

Разработайте набор чистых функций для работы с массивами:

- 1) Функция, которая принимает массив чисел и возвращает новый массив, содержащий только числа, кратные заданному числу.
- 2) Функция, которая принимает массив строк и возвращает новую строку, содержащую все строки, объединенные заданным разделителем.
- 3) Функция, которая принимает массив объектов и возвращает новый массив, отсортированный по значению определенного свойства.

Создайте функцию, которая принимает другую функцию в качестве аргумента и возвращает новую функцию, которая выполняет логирование перед вызовом исходной функции.

## Код

```
// функция, которая принимает массив чисел и возвращает новый массив,  
// содержащий только числа, кратные заданному числу.  
  
function filter_multiples(array: number[], multiple: number):  
number[] {  
    return array.filter(num => num % multiple === 0)  
}  
  
// функция, которая принимает массив строк и возвращает новую строку,  
// содержащую все строки, объединенные заданным разделителем.  
  
function join_strs(array: string[], sep: string): string {  
    return array.join(sep)  
}  
  
// функция, которая принимает массив объектов и возвращает новый массив,  
// отсортированный по значению определенного свойства.
```

```

function array_sort<T>(array: T[], property: keyof T): T[] {
    return [...array].sort((a, b) => a[property] > b[property] ? 1
: -1)
}

// Создайте функцию, которая принимает другую функцию в качестве
аргумента и возвращает новую функцию,
// которая выполняет логирование перед вызовом исходной функции.

function logDecorator<T extends (...args: any[]) => any>(func: T):
T {
    return ((...args: Parameters<T>) => {
        console.log(`Arguments: ${JSON.stringify(args)}`);
        return func(...args);
    }) as T
}

// Примеры

// Числа, кратные 3
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9];
const multiples = filter_multiples(numbers, 2);
console.log(multiples);

// Объединение строк
const strings = ['a', 'b', 'c'];
const joinedStrings = join_strs(strings, '^^^');
console.log(joinedStrings);

// Сортировка объектов по свойству
interface Person {
    name: string;
    age: number;
}

```

```

const yuras: Person[] = [
    { name: 'Yura', age: 20 },
    { name: 'NeYura', age: 19 },
    { name: 'MegaYura', age: 22 }
];

const sorted = array_sort(yuras, 'age');
console.log(sorted[0], sorted[1], sorted[2])

// Логирование перед вызовом функции
function add(a: number, b: number): number {
    return a + b;
}

const loggedAdd = logDecorator(add);
console.log(loggedAdd(2, 3));

```

### Ссылка на код

<https://github.com/MaksimovYuriy/FunctionalProgramming/tree/main/Лаб%203>

### Скриншот выполнения

```

C:\Program Files\nodejs\node.exe .\Лаб 3\lab.js
> (4) [2, 4, 6, 8]
a^^^b^^^c
> {name: 'NeYura', age: 19} {name: 'Yura', age: 20} {name: 'MegaYura', age: 22}
Arguments: [2,3]
5

```

Рис. 1 – Выполнение программы