

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

*Факультет 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 221-3710

Сычугова П.А.

(Фамилия И.О.)

Дата, подпись _____
(Дата) (Подпись)

Проверил: _____
(Фамилия И.О., степень, звание) (Оценка)

Дата, подпись _____
(Дата) (Подпись)

Замечания: _____

Москва

2025

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.

Ход работы

1. Функция для получения чисел, кратных заданному значению

```
function getDivisibleValues(values: number[], divisor: number): number[] {  
    return values.filter((val) => val % divisor === 0);  
}
```

2. Функция объединения строк с разделителем

```
function concatenateText(items: string[], separator: string): string {  
    return items.join(separator);  
}
```

3. Функция сортировки массива объектов по заданному полю

```
function orderByField<T, K extends keyof T>(list: T[], field: K): T[] {  
    const copy = [...list];  
    copy.sort((itemA, itemB) => {  
        const aVal = itemA[field];  
        const bVal = itemB[field];
```

```
    if (aVal < bVal) return -1;
    if (aVal > bVal) return 1;
    return 0;
  });
  return copy;
}
```

4. Функция-обёртка для логирования перед выполнением

```
function logBeforeExecution<T extends (...args: any[]) => any>(callback: T): T {
  {
    return function (...args: Parameters<T>): ReturnType<T> {
      const name = (callback as any).name || "unnamed";
      console.log(`Invoking function "${name}" with:`, args);
      const output = callback(...args);
      console.log(`Output of "${name}":`, output);
      return output;
    } as T;
  }
}
```

Примеры данных:

```
const dataSet = [1, 10, 15, 20, 22, 25, 34, 40];
const phrases = ["I'm", "sorry", "I", "didn't", "send", "the", "lab", "on", "time",
"(T-T)"];
const people = [
  { username: "Tom", score: 88 },
  { username: "Anna", score: 92 },
  { username: "Mike", score: 75 },
];

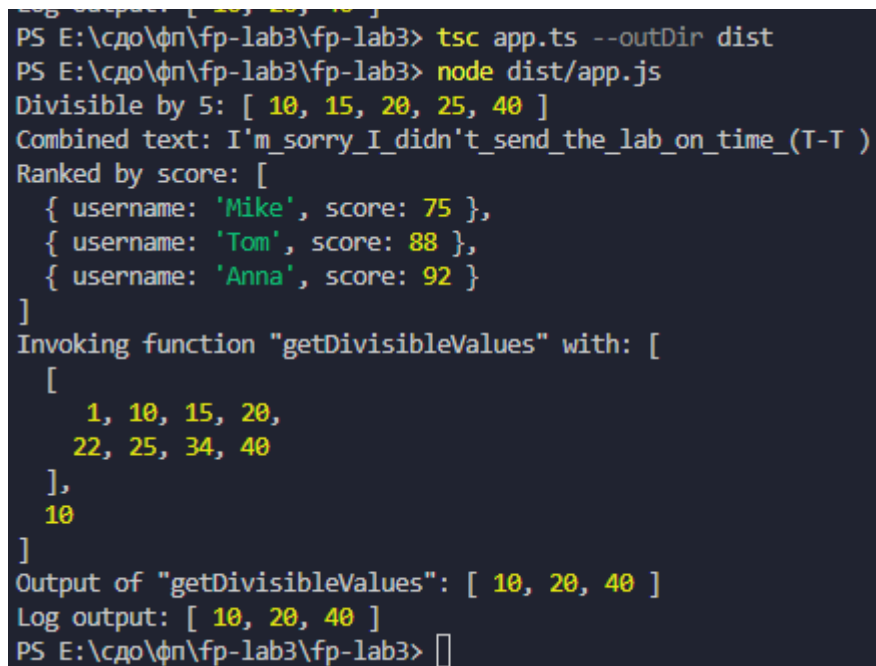
const divisibleBy5 = getDivisibleValues(dataSet, 5);
console.log("Divisible by 5:", divisibleBy5);
```

```
const combinedText = concatenateText(phrases, "_");
console.log("Combined text:", combinedText);

const rankedPeople = orderByField(people, "score");
console.log("Ranked by score:", rankedPeople);

const loggedDivisibleCheck = logBeforeExecution(getDivisibleValues);
const logOutput = loggedDivisibleCheck(dataSet, 10);
console.log("Log output:", logOutput);
```

Результат:



```
PS E:\сдо\фп\fp-lab3\fp-lab3> tsc app.ts --outDir dist
PS E:\сдо\фп\fp-lab3\fp-lab3> node dist/app.js
Divisible by 5: [ 10, 15, 20, 25, 40 ]
Combined text: I'm_sorry_I_didn't_send_the_lab_on_time_(T-T )
Ranked by score: [
  { username: 'Mike', score: 75 },
  { username: 'Tom', score: 88 },
  { username: 'Anna', score: 92 }
]
Invoking function "getDivisibleValues" with: [
  [
    1, 10, 15, 20,
    22, 25, 34, 40
  ],
  10
]
Output of "getDivisibleValues": [ 10, 20, 40 ]
Log output: [ 10, 20, 40 ]
PS E:\сдо\фп\fp-lab3\fp-lab3> 
```

Рисунок 1.Пример работы