Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»
Институт радиоэлектроники и информационных технологий - РТФ
Школа бакалавриата

ЛАБОРАТОРНАЯ РАБОТА №3

«Знакомство с Javascript»

Выполнил Филатов Д. С.
группа РИМ-240950

Екатеринбург
2025

Цель: Познакомиться с Javascript

Задачи:

1. **Подготовка проекта**
   - Инициализировать npm-проект.
   - Добавить в package.json скрипт “start” для запуска index.js
2. **Сортировка массива**
   - Реализовать алгоритм сортировки массива (любой).
   - Оценить сложность по времени (O(...)) и памяти.
3. **Бинарный поиск**
   - Реализовать бинарный поиск в отсортированном массиве.
   - Возвращать индекс найденного элемента или -1.
   - Указать временную и пространственную сложность.
4. **Проверка скобочных последовательностей**
   - Реализовать функцию проверки корректности скобок ()[]{} в строке.
   - Условие корректности: каждая открывающая скобка должна иметь соответствующую закрывающую и быть правильно вложена.

Github: https://github.com/K-gns/JS-Lab-3-kgns

ХОД ВЫПОЛНЕНИЯ

1. Часть 1

Инициализировал npm-проект с помощью npm init, создал js-файлы для каждой задачи, добавил скрипты для запуска файлов:

```
"scripts": {
  "start-sort": "node sort.js",
  "start-search": "node binarySearch.js",
  "start-brackets": "node bracketsTask.js"
}
```

2. Часть 2

Реализовал сортировку вставками в sort.js:

```
/**
 * Сортировка вставками
 * @param {Array} arr - Массив для сортировки
 * @param {string} direction - Направление сортировки: 'asc' или 'desc'
 * @returns {Array} - Отсортированный массив
 */
function insertionSort(arr, direction = 'asc') {
    for (let i = 1; i < arr.length; i++) {
        const key = arr[i];
        let j = i - 1;

        // Сравниваем и сдвигаем элементы в зависимости от direction
        if (direction === 'asc') {
            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j--;
            }
        } else {
            while (j >= 0 && arr[j] < key) {
                arr[j + 1] = arr[j];
                j--;
            }
        }

        // Вставляем элемент на нужную позицию
          arr[j + 1] = key;
    }

    return arr;
}

// Временная сложность:
```

```
// Средний случай: O(n²) - в среднем примерно n²/4 сравнений
// Лучший случай: O(n) - когда массив уже отсортирован, цикл
выполняется минимальное количество раз

//  Сложность по памяти:
// O(1) - константная память, меняем исходный массив

// Примеры использования
const numbers = [66, 33, 22, 44, 11, 77, 99];
console.log('Исходный массив:', [...numbers]);

const numbers1 = [...numbers]
insertionSort(numbers1, 'asc');
console.log('Сортированный по возрастанию:', numbers1);

const numbers2 = [...numbers]
insertionSort(numbers2, 'desc');
console.log('Сортированный по убыванию:', numbers2);
```

3. Часть 3

Реализовал бинарный поиск в отсортированном массиве в binarySearch.js:

```
/**
* Бинарный поиск
* @param {Array} arr - Отсортированный массив
* @param {*} target - Искомый элемент
* @returns {number} - Индекс элемента или -1, если не найден
*/
function binarySearch(arr, target) {
    let left = 0;
    let right = arr.length - 1;
    let step = 1;

    while (left <= right) {
        const mid = Math.floor(left + (right - left) / 2);

        if (arr[mid] === target) {
            return mid;
        }

        if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
        step++;
    }

    return -1;
}
```

```
const sortedArray = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25];
console.log("Отсортированный массив:", [...sortedArray])

const index = binarySearch(sortedArray, 9);
console.log(index > 0 ? `Найдено на позиции ${index}` : "Элемент не
найден" )
```

4. Часть 4

Реализовал функцию проверки корректности скобок в bracketsTask.js:

```
/**
 * Проверка корректности скобочной последовательности
 * @param {string} str - Строка для проверки
 * @returns {boolean} - true если скобки корректны, false иначе
 */
function isValidBrackets(str) {
    const stack = [];

    for (let i = 0; i < str.length; i++) {
        const char = str[i];

        // Если это открывающая скобка - добавляем в стек
          if (char === '(' || char === '[' || char === '{') {
            stack.push(char);
        }

        // Если это закрывающая скобка
          else if (char === ')' || char === ']' || char === '}') {
            // Проверяем, есть ли что-то в стеке
              if (stack.length === 0) {
                return false;
            }

            // Достаём последнюю открывающую скобку
              const top = stack.pop();

            // Проверяем соответствие для каждого типа
              if (char === ')' && top !== '(') {
                return false;
            }
            if (char === ']' && top !== '[') {
                return false;
            }
            if (char === '}' && top !== '{') {
                return false;
            }
        }

        // Остальные символы игнорируем
    }
```

```
    return stack.length === 0;
}

console.log('{[}]:', isValidBrackets('{[}]'));
console.log('{[]}:', isValidBrackets('{[]}'));
console.log('(():', isValidBrackets('(()'));
```

Код лабораторной работы можно посмотреть в github репозитории: **Github:** https://github.com/K-gns/JS-Lab-3-kgns

Вывод: В ходе выполнения лабораторной работы познакомились с основами JavaScript и npm-проектов. Были реализованы и протестированы три задачи: алгоритм сортировки (сортировка вставками), бинарный поиск в отсортированном массиве и проверка корректности скобочных последовательностей. Поставленные задачи выполнены и позволяют закрепить базовые навыки программирования на JavaScript.