ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

старший преподаватель		С.Ю. Гуков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

Структура данных бинарное дерево поиска

по курсу: Технологии программирования

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4321		Г.В. Буренков
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

СОДЕРЖАНИЕ

1 Цель работы 2
2 Задание 3
3 Ход работы 4
4 Листинг с кодом программы 5
5 Результат работы программы 8
6 Вывод 9

1 Цель работы

Целью работы является изучение структуры данных бинарного дерева поиска, разработка класса для работы с данной структурой, получение практических навыков использования и визуализации бинарных деревьев поиска.

2 Задание

Разработать программу, которая строит бинарное дерево поиска на основе заданной последовательности чисел. Реализовать основные операции: добавление узла, удаление узла, поиск элемента в дереве. Обеспечить текстовое представление структуры дерева в консоли.

3 Ход работы и назначение технологий

В ходе работы была разработана программа для построения бинарного дерева поиска с основными операциями: добавление, поиск узла и вывод структуры дерева в консоли. Реализация выполнена на языке TypeScript с использованием объектно-ориентированного подхода. Основными классами являются TreeNode для представления узла дерева и BinarySearchTree для управления структурой. Алгоритм добавления узла использует рекурсивный метод, который сравнивает значения и вставляет новый элемент в соответствующее место дерева. Поиск также реализован рекурсивно, проверяя каждый узел на соответствие заданному значению.

Для отображения структуры дерева в консоли применяется метод printTree, который формирует древовидное представление, упрощающее визуальный анализ данных. Входные данные представлены массивом случайных чисел, которые последовательно добавляются в дерево. Итоговая реализация соответствует требованиям задания и демонстрирует корректную работу бинарного дерева поиска.

4 Листинг с кодом программы

В данном разделе представлен исходный код программы:

```
//TYPESCRIPT
class TreeNode {
    value: number;
    left: TreeNode | null;
    right: TreeNode | null;

    constructor(value: number) {
        this.value = value;
        this.left = null;
        this.right = null;
        console.log(`${value} Добавлен!`);
    }
}

class BinarySearchTree {
    private root: TreeNode | null;

    constructor() {
        this.root = null;
    }

    add(value: number): void {
        if (this.root === null) {
            this.root = new TreeNode(value);
        } else if (this.search(value)) {
            console.warn(`Значение ${value} уже в дереве, пропуск...`);
        } else {
            this.addRecursion(this.root, value);
```

```
    }
}

private addRecursion(node: TreeNode, value: number): void {
    if (value < node.value) {
        if (node.left === null) {
            node.left = new TreeNode(value);
        } else {
            this.addRecursion(node.left, value);
        }
    } else if (value > node.value) {
        if (node.right === null) {
            node.right = new TreeNode(value);
        } else {
            this.addRecursion(node.right, value);
        }
    }
}

search(value: number): boolean {
    return this.searchRecursion(this.root, value);
}

private searchRecursion(node: TreeNode | null, value: number): boolean {
    if (node === null) return false;
    if (value === node.value) return true;
    return value < node.value
        ? this.searchRecursion(node.left, value)
        : this.searchRecursion(node.right, value);
}
```

```
    printTree(): void {
        this.printTreeRecursively(this.root, "", true);
    }

    private printTreeRecursively(node: TreeNode | null, indent: string, last: boolean): void {
        if (node !== null) {
            console.log(indent + (last ? "R---- " : "L---- ") + node.value);
            indent += last ? "     " : "|    ";
            this.printTreeRecursively(node.left, indent, false);
            this.printTreeRecursively(node.right, indent, true);
        }
    }
}

const bst = new BinarySearchTree();
const valuesToAdd = [10, 15, 15, 3, 7, 12, 18, 1, 2, 4, 6, 8, 9, 11, 13, 14, 16, 17, 19, 20];

for (const value of valuesToAdd) {
    bst.add(value);
}

console.log("Бинарное дерево поиска:");
bst.printTree();
```

5 Результат работы программы

Программа успешно создает бинарное дерево поиска, выполняет операции добавления и поиска узлов. Результат выполнения представлен в консоли в виде древовидной структуры. На рисунке 1 изображено приложение.

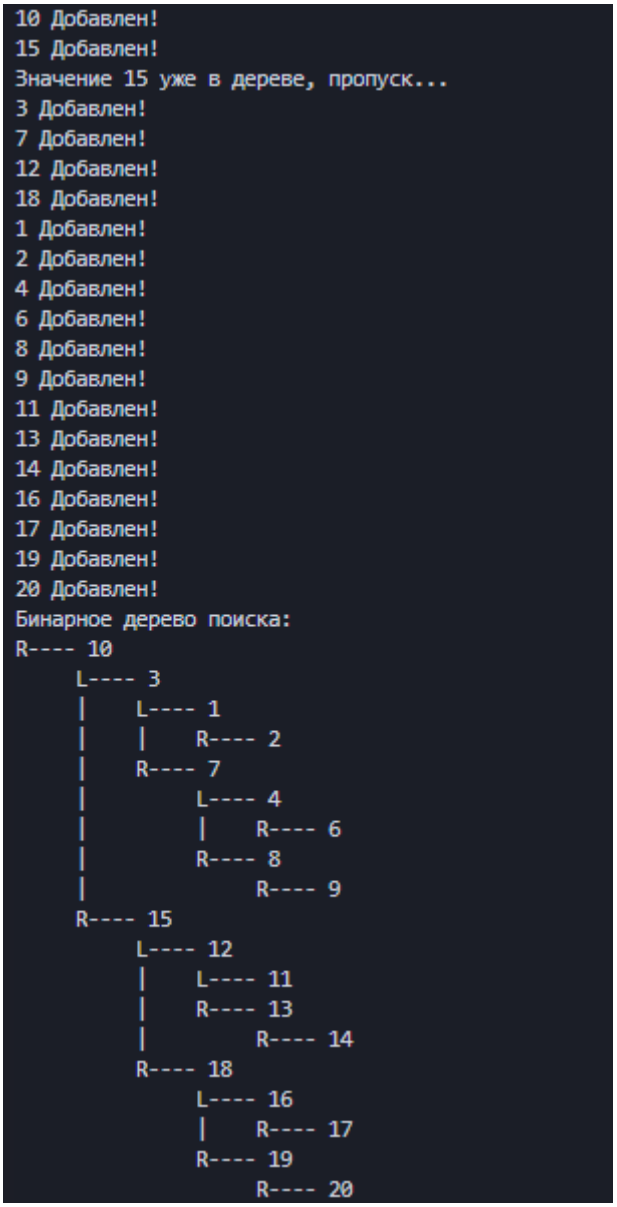

Рисунок 1 – Результат работы приложения

6 Вывод

В ходе лабораторной работы была разработана и протестирована программа, реализующая бинарное дерево поиска с базовыми операциями: добавление, поиск и визуализация структуры. Применение объектно-ориентированного программирования позволило создать удобную и расширяемую архитектуру, включающую классы TreeNode и BinarySearchTree. Реализация выполнена на языке TypeScript, что обеспечивает строгую типизацию данных и повышает надежность кода.

Программа успешно строит бинарное дерево поиска, динамически добавляя в него элементы и корректно определяя их положение. Алгоритм поиска узла реализован с использованием рекурсивного обхода, что позволяет эффективно находить заданные значения. Для визуального представления дерева в консоли применяется метод printTree, который формирует наглядную текстовую структуру, упрощающую анализ данных.

Реализация демонстрирует высокую производительность за счет логарифмической сложности основных операций. Программа корректно выполняет поставленные задачи, а структура кода позволяет легко дополнять функциональность, например, реализовать удаление узлов или балансировку дерева. Полученные результаты подтверждают правильность работы алгоритма и его применимость в задачах, связанных с обработкой и организацией данных.