

**Министерство образования Российской Федерации**  
**Пензенский государственный университет**  
**Кафедра «Вычислительная техника»**

**Отчёт**

по лабораторной работе №4

по курсу «Л и О А в ИЗ»

на тему «Бинарное дерево поиска»

Выполнили студенты группы 24ВВВ4:

Кондратьев С.В.

Кошелев Р.Д.

Приняли к.т.н., доцент:

Юрова О.В.

к.э.н., доцент:

Акифьев И.В.

Пенза 2025

## **Общие сведения.**

Бинарные деревья – это деревья, у каждого узла которого возможно наличие только двух сыновей. Двоичные деревья являются упорядоченными.

## **Задание**

1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
2. Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.
3. \* Изменить функцию добавления элементов для исключения добавления одинаковых символов.
4. \* Оценить сложность процедуры поиска по значению в бинарном дереве.

## **Листинг**

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <string.h>

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* root = NULL;

struct Node* CreateTree(struct Node* root, struct Node* r, int data)
{
    if (r == NULL)
    {
        r = (struct Node*)malloc(sizeof(struct Node));
        if (r == NULL) {
```

```

        printf("Ошибка выделения памяти");
        exit(0);
    }
    r->left = NULL;
    r->right = NULL;
    r->data = data;
    if (root == NULL) return r;
    if (data > root->data)
        root->right = r;
    else
        root->left = r;

    return r;
}

if (data > r->data)
    CreateTree(r, r->right, data);
else
    CreateTree(r, r->left, data);
return root;
}

void print_tree(struct Node* r, int l) {
    if (r == NULL) {
        return;
    }
    print_tree(r->right, l + 1);
    for (int i = 0; i < l; i++) {
        printf(" ");
    }
    printf("%d\n", r->data);
    print_tree(r->left, l + 1);
}

```

```
}
```

```
int search(struct Node* root, int value) {
```

```
    if (root == NULL) {
```

```
        return 0;
```

```
    }
```

```
    if (root->data == value) {
```

```
        return 1;
```

```
    }
```

```
    if (value > root->data) {
```

```
        return search(root->left, value);
```

```
    }
```

```
    else {
```

```
        return search(root->right, value);
```

```
    }
```

```
}
```

```
int countOccurrences(struct Node* root, int value) {
```

```
    if (root == NULL) {
```

```
        return 0;
```

```
    }
```

```
    int count = 0;
```

```
    if (root->data == value) {
```

```
        count = 1;
```

```
    }
```

```
    if (value >= root->data) {
```

```
        count += countOccurrences(root->left, value);
```

```
    }
```

```
    if (value <= root->data) {
```

```
        count += countOccurrences(root->right, value);
```

```
    }
```

```

    return count;
}
int main() {
    setlocale(LC_ALL, "");
    int D, start = 1;
    root = NULL;
    printf("-1 - окончание построения дерева\n");
    while (1) {
        printf("Введите число: ");
        if (fgets(input, sizeof(input), stdin) == NULL) {
            break;
        }
        input[strcspn(input, "\n")] = 0;
        if (strcmp(input, "stop") == 0) {
            printf("Построение дерева окончено\n\n");
            break;
        }
        if (sscanf(input, "%d", &D) == 1) {
            root = CreateTree(root, root, D);
        }
        else {
            printf("Ошибка! Введите число или 'стоп'\n");
        }
    }
    print_tree(root, 0); printf("Построенное дерево:\n");
    print_tree(root, 0);
    printf("\n");
    int searchValue;
    printf("Введите значение для поиска: ");

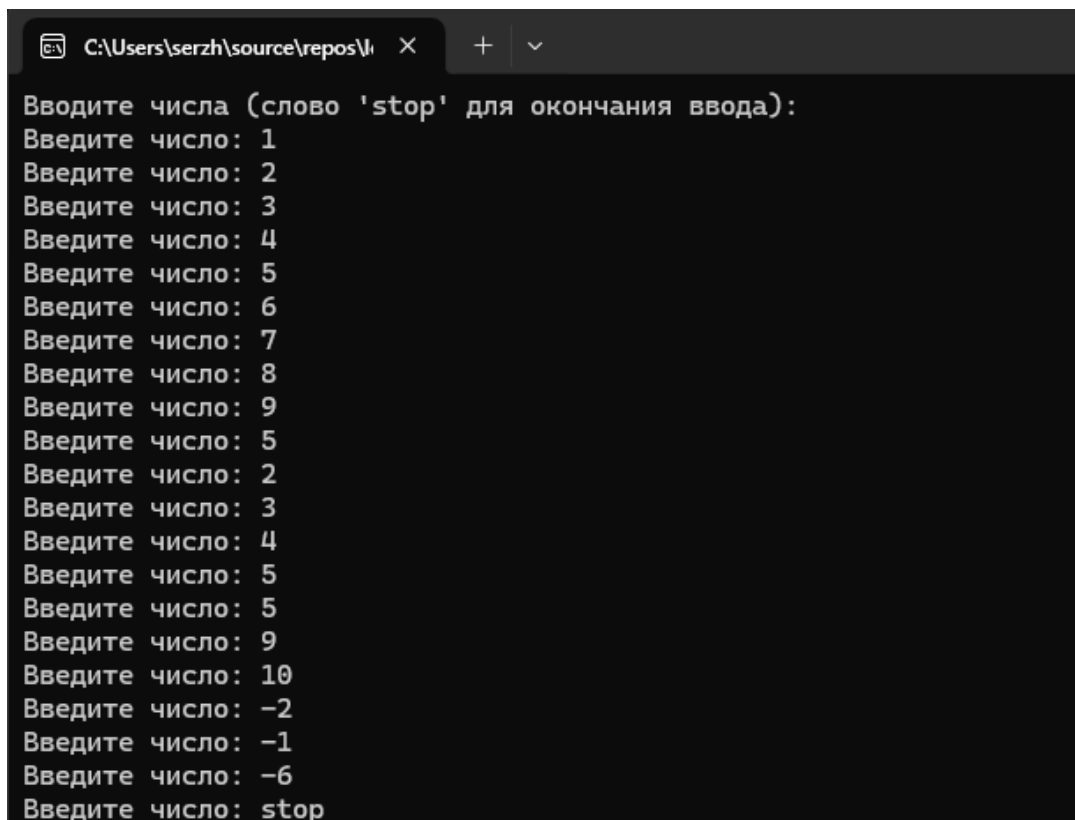
```

```

scanf("%d", &searchValue);
if (search(root, searchValue)) {
    printf("Элемент %d найден в дереве.\n", searchValue);
}
else {
    printf("Элемент %d не найден в дереве.\n", searchValue);
}
int countValue;
printf("Введите значение для подсчета вхождений: ");
scanf("%d", &countValue);
int occurrences = countOccurrences(root, countValue);
printf("Число вхождений элемента %d в дереве: %d\n", countValue,
occurrences);
return 0;
}

```

### Результат работы программы



```

C:\Users\serzh\source\repos\ x + v
Вводите числа (слово 'stop' для окончания ввода):
Введите число: 1
Введите число: 2
Введите число: 3
Введите число: 4
Введите число: 5
Введите число: 6
Введите число: 7
Введите число: 8
Введите число: 9
Введите число: 5
Введите число: 2
Введите число: 3
Введите число: 4
Введите число: 5
Введите число: 5
Введите число: 9
Введите число: 10
Введите число: -2
Введите число: -1
Введите число: -6
Введите число: stop

```

Рисунок 1 – Ввод чисел

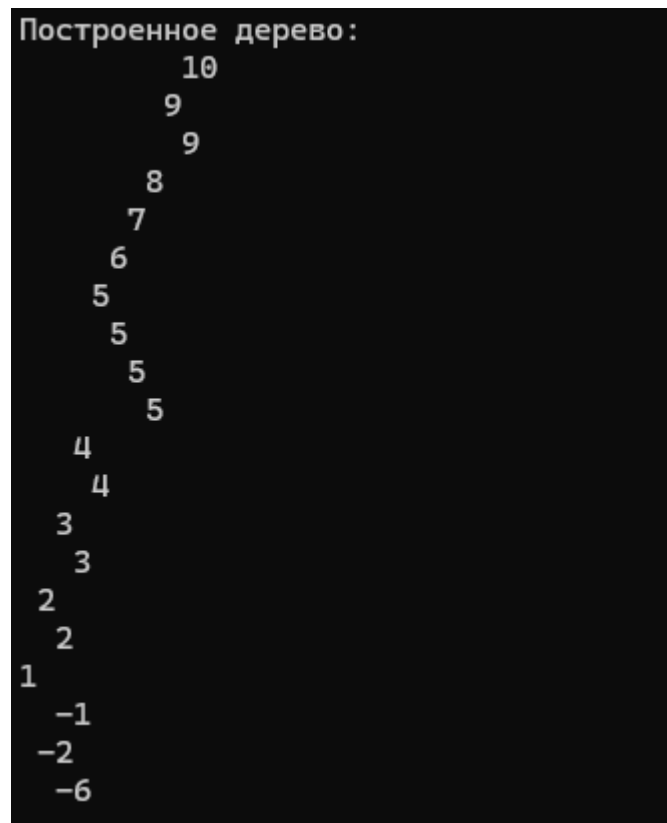


Рисунок 2 – Построенное дерево

**Вывод:** В результате выполнения лабораторной работы приобретены навыки работы с бинарными деревьями, поиска его элементов, создания функций подсчета вхождений заданного элемента.