

Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ОТЧЁТ

по лабораторной работе №4
по курсу «Логика и основы алгоритмизации в инженерных задачах»
на тему «Бинарное дерево поиска»

Выполнили:
студенты группы 24ВВВЗ
Кукушкин Антон
Азаров Максим

Приняли:
Юрова О.В.
Деев М.В.

Пенза 2025

Цель работы – Ознакомиться с бинарными деревьями и освоить принципы работы с ними.

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.

Задание 1

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <locale.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\n");
            exit(0);
        }

        r->left = NULL;
        r->right = NULL;
        r->data = data;

        if (root == NULL) return r;

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

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

```

        return root;
    }

struct Node* search_tree(struct Node* root, int target)
{
    if (root == NULL)
        return NULL;

    if (root->data == target)
        return root;

    if (target > root->data)
        return search_tree(root->left, target);
    else
        return search_tree(root->right, target);
}

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 main()
{
    setlocale(LC_ALL, "");
    int D, start = 1;

    root = NULL;
    printf("-1 - окончание построения дерева\n");
    while (start)
    {
        printf("Введите число: ");
        scanf("%d", &D);
        if (D == -1)
        {
            printf("Построение дерева окончено\n\n");
            start = 0;
        }
        else
            root = CreateTree(root, root, D);
    }

    print_tree(root, 0);

    printf("Введите число для поиска: ");
    scanf("%d", &D);
    struct Node* found = search_tree(root, D);
    if (found != NULL)
        printf("Значение %d найдено в дереве.\n", D);
    else

```

```

        printf("Значение %d не найдено в дереве.\n", D);
    return 0;
}

```

Результат работы программы показан на рисунке 1

```

Консоль отладки Microsoft Visual Studio
-1 - окончание построения дерева
Введите число: 45
Введите число: 31
Введите число: 10
Введите число: 20
Введите число: 22
Введите число: 34
Введите число: 57
Введите число: 32
Введите число: 2
Введите число: 4
Введите число: -1
Построение дерева окончено

      2
     / \
    10  4
   / \
  31  20
 / \
45  32
   / \
  34  22
     /
    57

Введите число для поиска: 22
Значение 22 найдено в дереве.

```

Рисунок 1

Задание 2

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <locale.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\n");

```

```

        exit(0);
    }

    r->left = NULL;
    r->right = NULL;
    r->data = data;

    if (root == NULL) return r;

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

if (data > r->data)
    CreateTree(r, r->left, data);
else
    CreateTree(r, r->right, 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 count_occurrences(struct Node* r, int value)
{
    if (r == NULL)
        return 0;

    int count = 0;
    if (r->data == value)
        count = 1;

    return count + count_occurrences(r->left, value) +
count_occurrences(r->right, value);
}

int main()
{
    setlocale(LC_ALL, "");
    int D, start = 1;

    root = NULL;
    printf("-1 - окончание построения дерева\n");
}

```

```

while (start)
{
    printf("Введите число: ");
    scanf("%d", &D);
    if (D == -1)
    {
        printf("Построение дерева окончено\n\n");
        start = 0;
    }
    else
        root = CreateTree(root, root, D);
}

print_tree(root, 0);

printf("Введите число для подсчёта вхождений: ");
scanf("%d", &D);
int occurrences = count_occurrences(root, D);
printf("Число вхождений значения %d в дереве: %d\n", D,
occurrences);

return 0;
}

```

Результат работы программы показан на рисунке 2

```

Введите число: 6
Введите число: 4
Введите число: 239
Введите число: 22
Введите число: 22
Введите число: 58
Введите число: 33
Введите число: 43
Введите число: 33
Введите число: 10
Введите число: -1
Построение дерева окончено

2
  6   4
   \  /
    10 22 33
      / \
     33 43
    /  \
   54   58
    \   /
    239

Введите число для подсчёта вхождений: 33
Число вхождений значения 33 в дереве: 2

```

Рисунок 2

Задание 3

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

```

```

#include <locale.h>

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

struct Node* root = NULL;

struct Node* CreateTree(struct Node* r, int data)
{
    if (r == NULL)
    {
        r = (struct Node*)malloc(sizeof(struct Node));
        if (r == NULL)
        {
            printf("Ошибка выделения памяти\n");
            exit(0);
        }

        r->left = NULL;
        r->right = NULL;
        r->data = data;
        return r;
    }

    if (data == r->data)
    {
        printf("Значение %d уже существует в дереве, добавление  
пропущено\n", data);
        return r;
    }
    else if (data < r->data)
    {
        r->left = CreateTree(r->left, data);
    }
    else
    {
        r->right = CreateTree(r->right, data);
    }
    return r;
}

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);
}

```

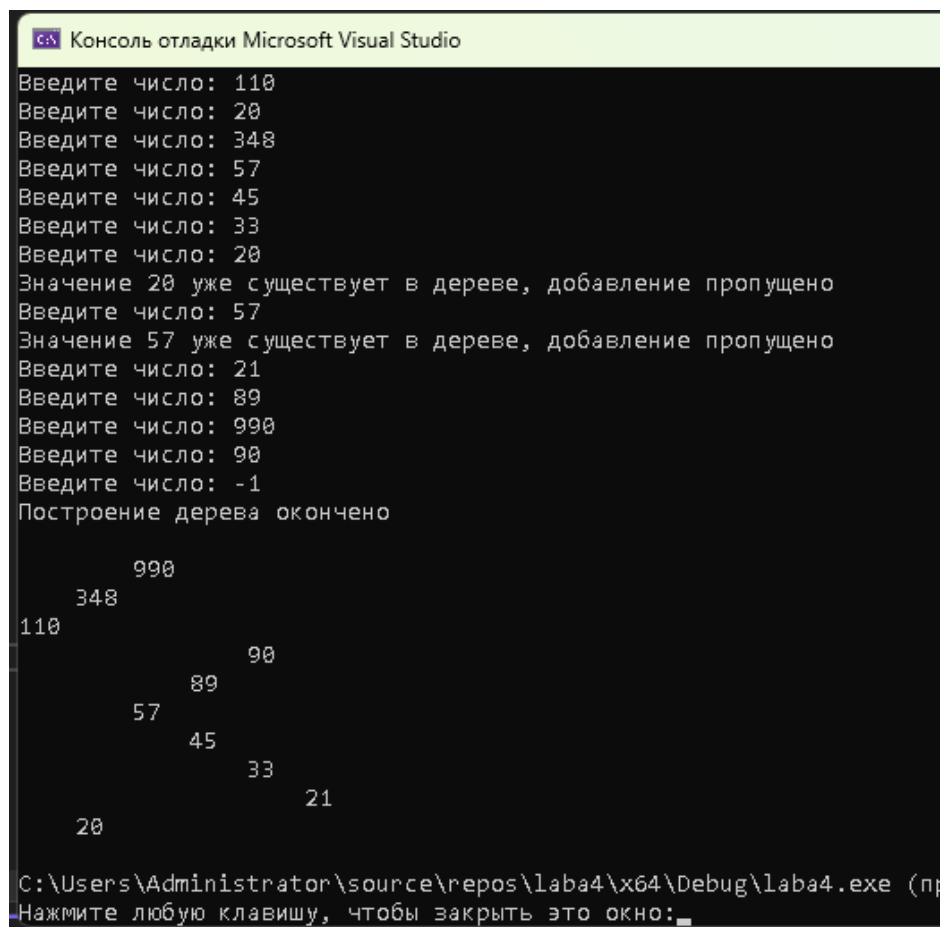
```
int main()
{
    setlocale(LC_ALL, "");
    int D, start = 1;

    root = NULL;
    printf("-1 - окончание построения дерева\n");
    while (start)
    {
        printf("Введите число: ");
        scanf("%d", &D);
        if (D == -1)
        {
            printf("Построение дерева окончено\n\n");
            start = 0;
        }
        else
            root = CreateTree(root, D);
    }

    print_tree(root, 0);

    return 0;
}
```


Результат работы программы представлен на рисунке 3



```
Консоль отладки Microsoft Visual Studio
Введите число: 110
Введите число: 20
Введите число: 348
Введите число: 57
Введите число: 45
Введите число: 33
Введите число: 20
Значение 20 уже существует в дереве, добавление пропущено
Введите число: 57
Значение 57 уже существует в дереве, добавление пропущено
Введите число: 21
Введите число: 89
Введите число: 990
Введите число: 90
Введите число: -1
Построение дерева окончено

      990
     /  \
    348   90
   /  \  / \
  110  89 45 33
   \  / \  \
    57 45 33 21
     \ / \
      20 21

C:\Users\Administrator\source\repos\laba4\x64\Debug\laba4.exe (пр
Нажмите любую клавишу, чтобы закрыть это окно: _
```

Рисунок 3

Задание 4

```
struct Node* search_tree(struct Node* root, int target)
{
    if (root == NULL)
        return NULL;

    if (root->data == target)
        return root;

    if (target > root->data)
        return search_tree(root->left, target);
    else
        return search_tree(root->right, target);
}
```

Сложность процедуры поиска элемента по значению в бинарном дереве поиска зависит от высоты дерева и может быть оценена следующим образом:

- В среднем / в сбалансированном дереве: $O(\log n)$, где n — количество узлов. Это связано с тем, что на каждом шаге поиска мы переходим только в одно из поддеревьев, эффективно сокращая область поиска вдвое.
- В худшем случае (для вырожденного дерева, похожего на список): $O(n)$, где n — количество узлов. При этом обход может потребовать проверки всех элементов, если дерево не сбалансировано и все элементы лежат в одном поддереве.

Таким образом, сложность алгоритма поиска рекурсивной функцией находится в диапазоне от $O(\log n)$ до $O(n)$ в зависимости от баланса дерева.

Вывод: в ходе выполнения лабораторной работы ознакомились с бинарными деревьями и освоили принципы работы с ними.