

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №4

по дисциплине: «Логика и основы алгоритмизации в инженерных задачах.»

на тему: «Бинарное дерево поиска.»

Выполнили:

студенты группы 21ВВ4

Колокольцева У. А.

Нагорная Д. А.

Принял:

Акифьев И. В.

Юрова О. В.

Пенза, 2022

Цель работы: разработать программный код для работы с бинарным деревом.

Лабораторная работа:

Задание 1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.

```
#include <stdlib.h>
#include "locale.h"
#include "malloc.h"
#include "iostream"

using namespace std;

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

struct Node* root;
int quantity = 0;

////////////////////////////////////

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->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 searchroot(Node* tree, int search)
{
    if (tree != NULL) {

        if (search == tree->data) {
            quantity++;
        }

        searchroot(tree->left, search);
        searchroot(tree->right, search);
        return 0;
    }
}

int main()
{
    struct Node* root;
    setlocale(LC_ALL, "");
    int D, start = 1;
    int k = 1;

    root = NULL;

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

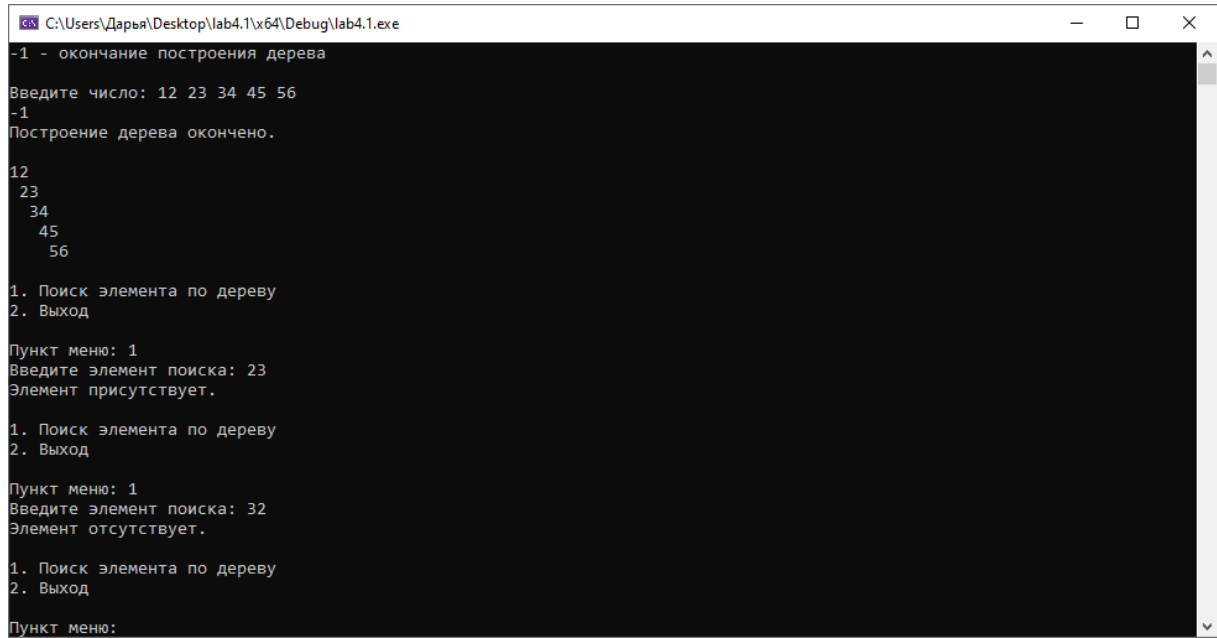
    print_tree(root, 0);
    while (k)
    {
        printf("\n1. Поиск элемента по дереву\n2. Выход\n\nПункт меню: ");
        cin >> k;
        if (k == 1)
        {
            int search;
            printf("Введите элемент поиска: ");
            cin >> search;
            searchroot(root, search);

            if (quantity != 0) {
                printf("Элемент присутствует. \n");
            }
            else
                printf("Элемент отсутствует. \n");
        }
    }
}

```

```
        quantity = 0;
        if (k == 2)
            return 0;
    }
    system("pause");
    return 0;
}
```

Результат работы программы:



```
C:\Users\Дарья\Desktop\lab4.1\Debug\lab4.1.exe
-1 - окончание построения дерева
Введите число: 12 23 34 45 56
-1
Построение дерева окончено.
12
 23
  34
   45
    56

1. Поиск элемента по дереву
2. Выход

Пункт меню: 1
Введите элемент поиска: 23
Элемент присутствует.

1. Поиск элемента по дереву
2. Выход

Пункт меню: 1
Введите элемент поиска: 32
Элемент отсутствует.

1. Поиск элемента по дереву
2. Выход

Пункт меню:
```

Задание 2. Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.

```
#include <stdlib.h>
#include "malloc.h"
#include "locale.h"
#include "iostream"

using namespace std;

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

struct Node* root;
int quantity = 0;

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->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 searchroot(Node* tree, int search)
{
}
```

```

if (tree != NULL) {

    if (search == tree->data) {
        quantity++;
    }

    searchroot(tree->left, search);
    searchroot(tree->right, search);
    return 0;
}
}

int main()
{
    struct Node* root;
    setlocale(LC_ALL, "");
    int D, start = 1;
    int k = 1;

    root = NULL;

    printf("-1 - окончание построения дерева\n");
    printf("\nВведите число: ");

    while (start)
    {

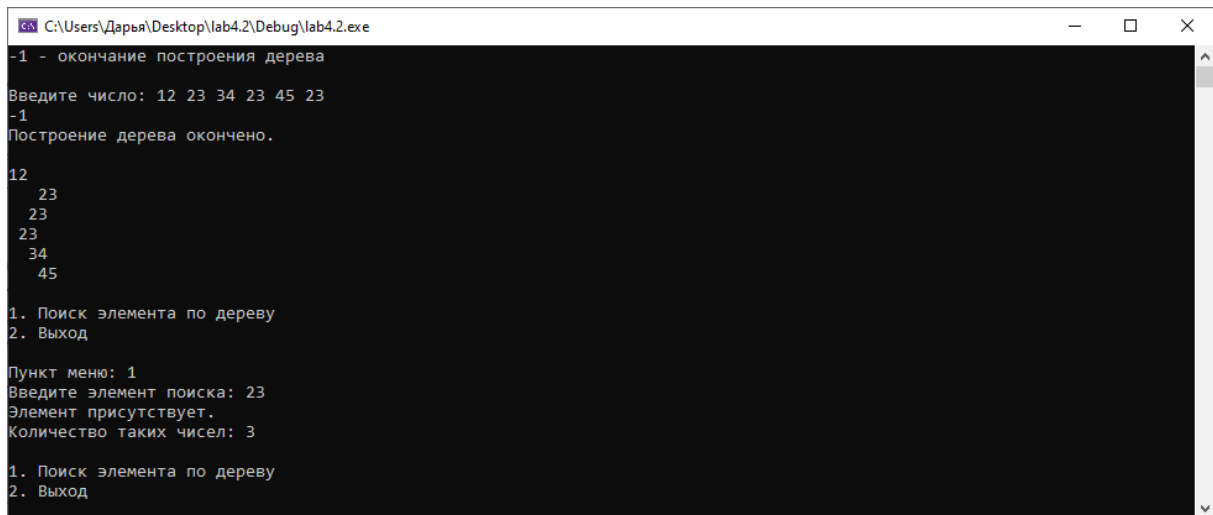
        scanf_s("%d", &D);
        if (D == -1)
        {
            printf("Построение дерева окончено.\n\n");
            start = 0;
        }
        else {
            root = CreateTree(root, root, D);
        }
    }

    print_tree(root, 0);
    while (k)
    {
        printf("\n1. Поиск элемента по дереву\n2. Выход\n\nПункт меню: ");
        cin >> k;
        if (k == 1)
        {
            int find;
            printf("Введите элемент поиска: ");
            cin >> find;
            searchroot(root, find);

            if (quantity != 0) {
                printf("Элемент присутствует. \n");
                cout << "Количество таких чисел: " << quantity << "\n";
            }
            else
                printf("Элемент отсутствует. \n");
        }
        quantity = 0;
        if (k == 2)
            return 0;
    }
    return 0;
}

```

Результат работы программы:



```
C:\Users\Дарья\Desktop\lab4.2\Debug\lab4.2.exe
-1 - окончание построения дерева
Введите число: 12 23 34 23 45 23
-1
Построение дерева окончено.

12
 23
 23
 23
 34
 45

1. Поиск элемента по дереву
2. Выход

Пункт меню: 1
Введите элемент поиска: 23
Элемент присутствует.
Количество таких чисел: 3

1. Поиск элемента по дереву
2. Выход
```

Задание 3. * Изменить функцию добавления элементов для исключения добавления одинаковых символов.

```
#include <stdlib.h>
#include "malloc.h"
#include "locale.h"
#include "iostream"

using namespace std;

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

struct Node* root;
int quantity = 0;

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->left = r;
        else root->right = r;
        return r;
    }

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

```

        }
        else
        {
            printf(" ");
            return root;
        }
    }
    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 searchroot(Node* tree, int search)
{
    if (tree != NULL) {

        if (search == tree->data) {
            quantity++;
        }

        searchroot(tree->left, search);
        searchroot(tree->right, search);
        return 0;
    }
}

int main()
{
    struct Node* root;
    setlocale(LC_ALL, "");
    int D, start = 1;
    int k = 1;

    root = NULL;

    printf("-1 - окончание построения дерева\n");
    printf("\nВведите числа: ");

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

    print_tree(root, 0);
}

```



```

while (k)
{
    printf("\n1. Поиск элемента по дереву\n2. Выход\n\nПункт меню: ");
    cin >> k;
    if (k == 1)
    {
        int search;
        printf("Введите элемент поиска: ");
        cin >> search;
        searchroot(root, search);

        if (quantity != 0) {
            printf("Элемент присутствует. \n");
            cout << "Количество таких чисел: " << quantity << "\n";
        }
        else
            printf("Элемент отсутствует. \n");
    }
    quantity = 0;
    if (k == 2)
        return 0;
}
return 0;
}

```

Результат работы программы:

```

C:\Users\Дарья\Desktop\lab4.3\Debug\lab4.3.exe
-1 - окончание построения дерева
Введите числа: 12 23 34 23 45 23
-1
Построение дерева окончено.
12
23
34
45

1. Поиск элемента по дереву
2. Выход

Пункт меню: 1
Введите элемент поиска: 23
Элемент присутствует.
Количество таких чисел: 1

1. Поиск элемента по дереву
2. Выход

Пункт меню: 1
Введите элемент поиска: 21
Элемент отсутствует.

```

Задание 4. * Оценить сложность процедуры поиска по значению в бинарном дереве.

Сложность порядка $O(n \log n)$, где n – глубина дерева. Вид операций: поиск по значению, вставка нового элемента, удаление элемента.

Вывод: мы реализовали алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве, функцию подсчёта числа вхождений заданного элемента в дерево, изменение функции добавления элементов для исключения добавления одинаковых символов, а также оценили сложность процедуры поиска по значению в бинарном дереве.