

Министерство науки и высшего образования  
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 23ВВВ4

Королёв Д.В.

Алешин К.А.

Приняли:

Юрова О.В.

Деев М.В.

Пенза, 2024

### **Общие сведения**

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

### **Задание 1. - Алгоритм поиска**

#### **Листинг**

```
#include <iostream>
#include <cstdlib>
#include <stdlib.h>

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

struct Node* root;

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("Error allocate memory");
            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);
}

bool find_value(Node* root, int value)
{
if (!root)
{
return false;
}
else if (root->data == value)

```

```

    {
        return true;
    }

    return find_value(root->right, value) ||
find_value(root->left, value);
}

```

```

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

    root = NULL;
    printf("-1 - for break\n");
    while (start)
    {
        printf("Enter number: ");
        scanf_s("%d", &D);
        if (D == -1)
        {
            printf("created tree finished\n\n");
            start = 0;
        }
        else
            root = CreateTree(root, root, D);
    }
    print_tree(root, 0);
    std::cout << "Enter value for search: ";
    int value;
    std::cin >> value;
    std::cout << std::endl;

    if (find_value(root, value))
    {
        std::cout << "True\n";
    }
    else

```

```

{
std::cout << "False\n";
}

//scanf_s("%d", &D);
return 0;
}

```

## Результат работы программы

```

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

    root = NULL;
    printf("-1 - for break\n");
    while (start)
    {
        printf("Enter number: ");
        scanf_s("%d", &D);
        if (D == -1)
        {
            printf("created tree finished\n\n");
            start = 0;
        }
        else
        {
            root = CreateTree(root, root, D);
        }
    }

    print_tree(root, 0);

    std::cout << "Enter value for search: ";
    int value;
    std::cin >> value;
    std::cout << std::endl;

    if (find_value(root, value))
    {
        std::cout << "True\n";
    }
}

```

```

-1 - for break
Enter number: 1
Enter number: 2
Enter number: 3
Enter number: 4
Enter number: 5
Enter number: 6
Enter number: 23
Enter number: 22
Enter number: 213
Enter number: 99
Enter number: -1
created tree finished

1
 2
 3
 4
 5
 6
 22
 23
 99
 213
Enter value for search: 22

True

```

Рисунок № 1

## Задание 2. - Алгоритм посчета вхождения элемента в дерево.

### Листинг

```

#include <iostream>
#include <cstdlib>
#include <stdlib.h>

```

```

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

```

```

};

struct Node* root;

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 find_count_value(Node* root, int value, int counter =
0)
{
if (!root)
{
return counter;
}
else if (root->data == value)
{
counter++;
}
counter = find_count_value(root->right, value, counter);
counter = find_count_value(root->left, value, counter);

return counter;
}

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

root = NULL;
printf("-1 - окончание построения дерева\n");
while (start)
{

```

```

printf("Введите число: ");
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);

std::cout << "Введите число для поиска: ";
int value;
std::cin >> value;
std::cout << std::endl;

std::cout << find_count_value(root, value);

return 0;
}

```

### Результат работы программы

```

76
77     return counter;
78 }
79
80
81 int main()
82 {
83     setlocale(LC_ALL, "");
84     int D, start = 1;
85
86     root = NULL;
87     printf("-1 - окончание построения дерева\n");
88     while (start)
89     {
90         printf("Введите число: ");
91         scanf_s("%d", &D);
92         if (D == -1)
93         {
94             printf("Построение дерева окончено\n\n");
95             start = 0;
96         }
97         else
98             root = CreateTree(root, root, D);
99     }
100
101     print_tree(root, 0);
102
103     std::cout << "Введите число для поиска: ";
104     int value;
105     std::cin >> value;
106

```

```

-1 - окончание построения дерева
Введите число: 23
Введите число: 1
Введите число: 2
Введите число: 5
Введите число: 23
Введите число: 1
Введите число: 6565
Введите число: 23
Введите число: 4
Введите число: 5
Введите число: 6
Введите число: -1
Построение дерева окончено
1
1
2
4
5
5
6
23
23
23
6565
Введите число для поиска: 23
3
C:\Users\kirya\source\repos\123\123\Debug\123.exe (process 28488) exited with 0.
To automatically close the console when debugging stops, enable Tools->Options->
Press any key to close this window . . .

```

Рисунок 2)



### **Задание 3. - процедура добавления не повторяющихся элементов.**

#### **Листинг**

```
#include <iostream>
#include <cstdlib>
#include <stdlib.h>

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

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

root = NULL;
printf("-1 - окончание построения дерева\n");
while (start)
{
printf("Введите число: ");
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);

return 0;
}

```

Результат работы программы

Рисунок 3)

```

Microsoft Visual Studio Debug
-1 - окончание построения дерева
Введите число: 10
Введите число: 10
Введите число: 10
Введите число: 10
Введите число: 2
Введите число: 3
Введите число: 4
Введите число: 5
Введите число: 6
Введите число: 7
Введите число: 1
Введите число: 2
Введите число: 3
Введите число: 4
Введите число: 5
Введите число: 6
Введите число: 7
Введите число: 45
Введите число: 12
Введите число: 3
Введите число: 4
Введите число: -1
Построение дерева окончено

1
2
3
4
5
6
7
10
12
45

C:\Users\kirya\source\repos\123\x64\Debug\123.exe (process 20832) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.

```

#### Задание 4. - Оценка сложности процедуры поиска по значению

Сложность процедуры поиска по значению в бинарном дереве поиска в среднем случае составляет  $O(\log n)$ . В этом случае дерево сбалансировано, и высота дерева приблизительно равна  $\log n$ , где  $n$  — количество узлов в дереве.

Однако в наихудшем случае, когда дерево становится вырожденным (например, все узлы — правые потомки одного перешагиваемого узла), сложность поиска может увеличиться до  $O(n)$

. Это происходит, когда бинарное дерево поиска теряет свою сбалансированность и становится похожим на связный список.

Для поддержания дерева в сбалансированном состоянии часто используют самобалансирующиеся деревья, такие как красно-черные деревья или AVL-деревья, в которых сложность поиска остаётся  $O(\log n)$  в худшем случае.

**Вывод** - были получены навыки реализации бинарного дерева, также процедур обхода дерева, нахождения элементов, подсчет элементов, оценка сложности алгоритма и его процедур.