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 ИЗ»

на тему «Бинарное дерево поиска»

Выполнил:

студент групп 22ВВВ2

Изосин М.А.

Приняли:

д.т.н. профессор Митрохин М.А.

к.э.н. доцент Акифьев И.В.

Пенза 2023

**Название**

Бинарное дерево поиска

**Цель работы**

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

Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.

**Лабораторное задание**

**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**

1 и 2)

#include <stdio.h>

#include <locale.h>

#include <malloc.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*left;

struct 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\_data(struct Node \*root, struct Node \*r, int element, int k)

{

if (r != NULL)

{

if (element == r->data)

{

k++;

Find\_data(r, r->right, element, k);

}

else if (element > r->data)

Find\_data(r, r->left, element, k);

else

Find\_data(r, r->right, element, k);

}

else

{

return k;

}

}

int main()

{

setlocale(LC\_ALL, "");

int D, start = 1;

int element, k = 0;

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);

printf("Введите элемент, который хотите найти: \n");

scanf("%d", &element);

k = Find\_data(root, root, element, k);

printf("Нашли %d - %d раз\n", element, k);

scanf\_s("%d", &D);

return 0;

}

2)

#include <stdio.h>

#include <locale.h>

#include <malloc.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*left;

struct 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 if (data < root->data)

root->right = r;

else

{

free(r);

}

return r;

}

if (data > r->data)

CreateTree(r, r->left, data);

else if (data < r->data)

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;

int element, k = 0;

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);

scanf\_s("%d", &D);

return 0;

}

**Результат работы программы**

Результат работы программы показан на рисунках.

1 и 2)

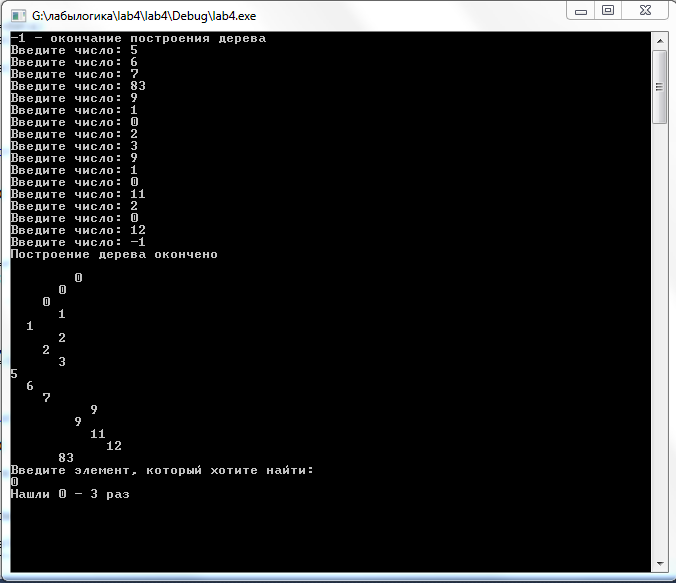


Рис. 1

3)

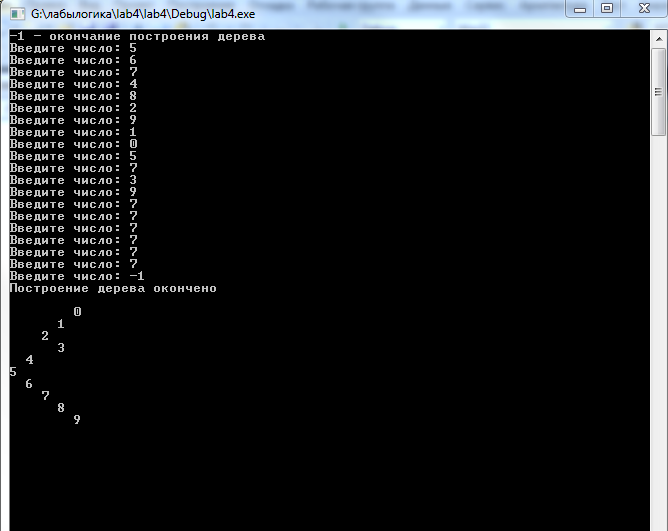


Рис. 2

**4)**

Сложность процедуры поиска по значению в бинарном дереве равна O(log n)

**Вывод:** в ходе выполнения лаб. работы мы научились создавать бинарное дерево, находить его элементы, а также число вхождений заданного элемента в дерево и исключать повторяющиеся элементы.

Ссылка на репозиторий: