Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации в инженерных задачах»

### на тему «Бинарное дерево поиска»

**Выполнил студент**

**группы 21вв1.1:**

Вартанов А.

**Приняли:**

Юрова О.В.

Акифьев И.В.

Пенза, 2022

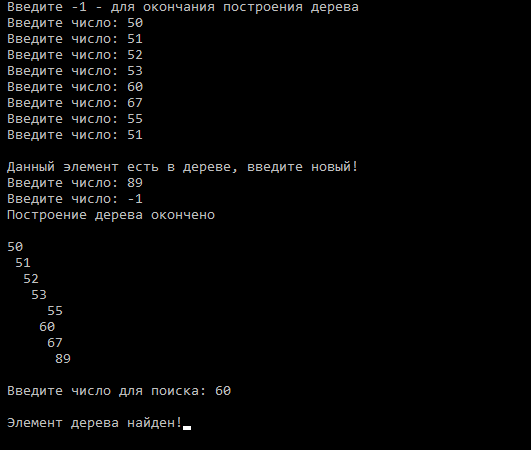
**Цель работы:** ознакомиться с бинарными деревьями, освоить основные виды манипуляций с ними в СИ.

**Задания:**

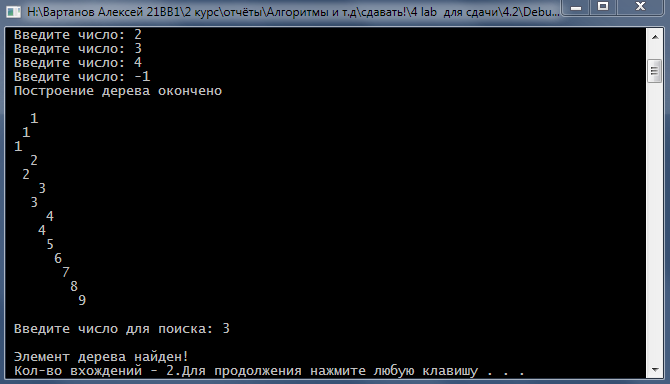
1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
2. Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.
3. Изменить функцию добавления элементов для исключения добавления одинаковых символов.
4. Оценить сложность процедуры поиска по значению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

**Ход работы**

**Результат программы для первого и третьего задания:**

****

**Результат программы для второго задания:**



**Задание 4:**

Сложность процедуры поиска в бинарном дереве -  **O(log n)**

**Вывод:** в лабораторной работе я научился писать программы с применением алгоритмов для работы с бинарными деревьями на языке Си.

**Листинг:**

**Для задания 1 и 3:**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdlib.h>

#include <iostream>

struct Node {

int data;

struct Node \*left;

struct Node \*right;

};

struct Node \*root;

int viborka(struct Node \*root, int data, int k)

{

if ( root == NULL )

{

k = 0;

return(k);

}

if ( data == root->data )

{

printf("\n Данный элемент есть в дереве, введите новый!\n");

k = 1;

return(k);

}

if ( data < root->data )

{

return viborka(root->right, data, k);

}

if ( data > root->data )

{

return viborka(root->left, data, k);

}

}

struct Node \*CreateTree(struct Node \*root, struct Node \*r, int data, int k)

{

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;

}

else

{

if(k == 0)

{

k = viborka(root,data,k);

if(k == 1)

{

return root;

}

}

}

if (data > r->data)

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

else

CreateTree(r, r->right, data, k);

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);

}

struct Node \*search(struct Node\* root, int number,int cout)

{

if ( root == NULL )

{

printf("\n Элемент не найден!");

return NULL;

}

if (number != root->data)

{

cout++;

}

if ( number == root->data )

{

//printf("\n Position: %2d",cout);

printf("\n Элемент дерева найден!\n \n Position: %2d",cout);

}

if ( number < root->data )

{

return search(root->right, number,cout);

}

if ( number > root->data )

{

return search(root->left, number,cout);

}

}

int main()

{

setlocale(LC\_ALL, "");

int D, start = 1;

int cout = 0;

int number;

int k = 0;

root = NULL;

printf(" Введите -1 - для окончания 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, k);

}

print\_tree(root,0);

printf("\n");

printf(" Введите число для поиска: ");

scanf("%d", &number);

search(root, number,cout);

printf("\n");

system("Pause");

return 0;

}

**Задание 2:**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdlib.h>

#include <iostream>

struct Node {

int data;

struct Node \*left;

struct Node \*right;

};

struct Node \*root;

struct Node \*CreateTree(struct Node \*root, struct Node \*r, int data, int k)

{

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, k);

else

CreateTree(r, r->right, data, k);

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);

}

struct Node \*search(struct Node\* root, int number)

{

if ( root == NULL )

{

printf("\n Элемент не найден!");

return NULL;

}

if ( number == root->data )

{

printf("\n Элемент дерева найден!");

}

if ( number < root->data )

{

return search(root->right, number);

}

if ( number > root->data )

{

return search(root->left, number);

}

}

int count(struct Node \*root, int number, int k)

{

if ( root == NULL )

{

printf("\n Кол-во вхождений - %d.", k);

return NULL;

}

if ( number == root->data )

{

k+=1;

return count(root->right, number, k);

}

if ( number < root->data )

{

return count(root->right, number, k);

}

if ( number > root->data )

{

return count(root->left, number, k);

}

}

int main()

{

setlocale(LC\_ALL, "");

int D, start = 1;

int number;

int k = 0;

root = NULL;

printf(" Введите -1 - для окончания 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, k);

}

print\_tree(root,0);

printf("\n");

printf(" Введите число для поиска: ");

scanf("%d", &number);

search(root, number);

count(root, number, k);

system("Pause");

return 0;

}