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»

**Выполнили студенты группы 21вв1:**

Нечаев А.Д.

Киреев Д.А.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

**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.

**Описание метода решения задачи:**

Обращение к дереву и его элементам осуществляется посредством указателя struct node\* root. Дальше мы написали структуру struct node\* add которая добавляет элементы в дерево. Также написали структуру struct node\* add1 которая добавляет только те элементы, которых не было в дереве. С помощью функции void vivod мы выводим дерево на экран (дерево выводится повёрнутым на 90 градусов, корень находится слева). С помощью функции void poisk мы находим одинаковые значение в дереве и считаем их общее количество. Сложность процедуры поиска по значению в бинарном дереве O(Log n).

**Листинг:**

**Задание 1-2:**

#include <iostream>

#include <locale.h>

typedef struct node

{

int zn;

struct node\* left;

struct node\* right;

};

struct node\* root;

struct node\* add(int a, struct node\* root, struct node\* p)

{

if (p == NULL)

{

p = (node\*)malloc(sizeof(struct node));

p->left = NULL;

p->right = NULL;

p->zn = a;

if (root == NULL) return p;

if (a > root->zn) root->left = p;

else root->right = p;

return p;

}

if (a > p->zn) add(a, p, p->left);

else add(a, p, p->right);

return root;

}

void vivod(int i, struct node\* p)

{

if (p == NULL)

{

return;

}

vivod(i + 1, p->right);

for (int d = 0; d < i; d++)

{

printf(" ");

}

printf("(%d)", i);

printf(" %d\n", p->zn);

vivod(i + 1, p->left);

}

void poisk(int y, int x, int i, struct node\* p)

{

if (p == NULL)

{

return;

}

poisk(y, x, i + 1, p->right);

if (x == p->zn)

{

printf(" %d", p->zn);

y++;

printf("\nУровень:%d", i);

printf("\nЧисло вхождений: %d", y);

}

poisk(y, x, i + 1, p->left);

}

int main()

{

setlocale(LC\_ALL, "Rus");

int d; int begin = 0, x = 0;

root = NULL;

while (begin != 4)

{

printf("\n1.Добавить элемент\n");

printf("2.Вывести набор элементов\n");

printf("3.Поиск\n");

printf("4.Выход\n");

printf("Выбор: ");

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

if (begin == 1)

{

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

root = add(d, root, root);

}

if (begin == 2)

{

vivod(0, root);

}

if (begin == 3)

{

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

poisk(0, x, 0, root);

}

}

}

**Задание 3:**

#include <iostream>

#include <locale.h>

typedef struct node

{

int zn;

struct node\* left;

struct node\* right;

};

struct node\* root;

struct node\* add1(int a, struct node\* root, struct node\* p)

{

if (p == NULL)

{

p = (node\*)malloc(sizeof(struct node));

p->left = NULL;

p->right = NULL;

p->zn = a;

if (root == NULL) return p;

if (a == root->zn)

{

printf("Вы ввели одинаковое зн-е");

return p;

}

if (a > root->zn) root->left = p;

else root->right = p;

return p;

}

if (a == p->zn)

{

printf("Вы ввели одинаковое зн-е");

return p;

}

if (a > p->zn) add1(a, p, p->left);

else add1(a, p, p->right);

return root;

}

void vivod(int i, struct node\* p)

{

if (p == NULL)

{

return;

}

vivod(i + 1, p->right);

for (int d = 0; d < i; d++)

{

printf(" ");

}

printf("(%d)", i);

printf(" %d\n", p->zn);

vivod(i + 1, p->left);

}

void poisk(int y, int x, int i, struct node\* p)

{

if (p == NULL)

{

return;

}

poisk(y, x, i + 1, p->right);

if (x == p->zn)

{

printf(" %d", p->zn);

y++;

printf("\nУровень:%d", i);

printf("\nЧисло вхождений: %d", y);

}

poisk(y, x, i + 1, p->left);

}

int main()

{

setlocale(LC\_ALL, "Rus");

int d; int begin = 0, x = 0;

root = NULL;

while (begin != 4)

{

printf("\n1.Добавить элемент\n");

printf("2.Вывести набор элементов\n");

printf("3.Поиск\n");

printf("4.Выход\n");

printf("Выбор: ");

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

if (begin == 1)

{

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

root = add1(d, root, root);

}

if (begin == 2)

{

vivod(0, root);

}

if (begin == 3)

{

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

poisk(0, x, 0, root);

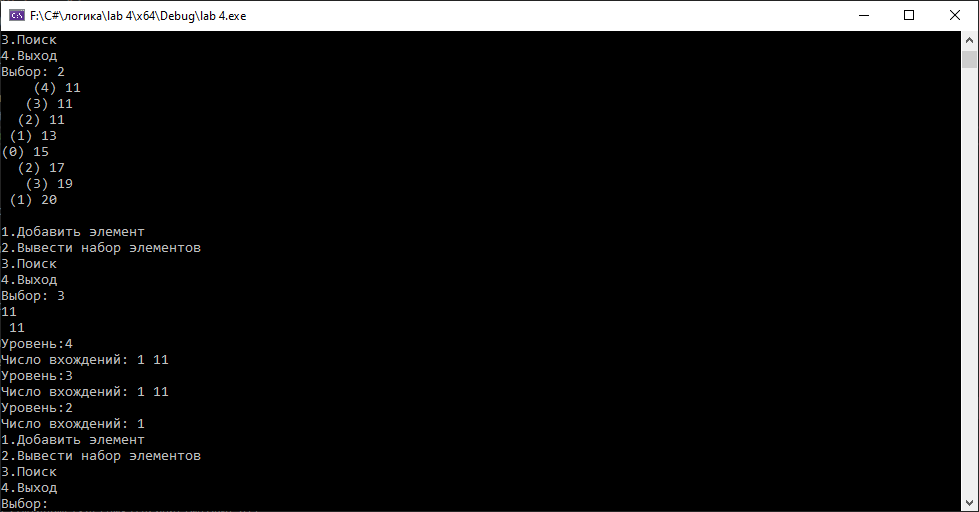
}

}

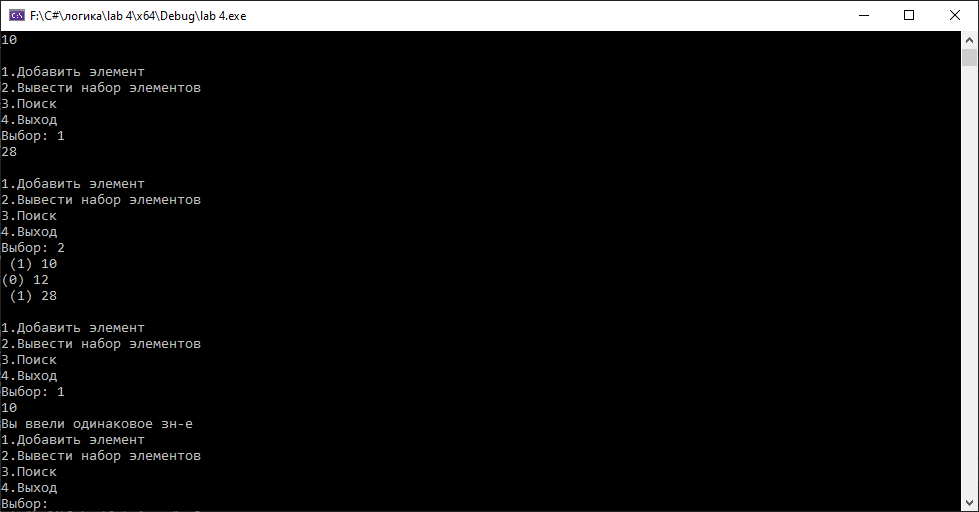
}

**Результаты работы программы:**

**Задание 1-2:**

****

**Задание 3:**

****

**Вывод:** в ходе выполнения лабораторной работы мы научились работать с бинарным деревом поиска. Результаты работы программ совпали с результатами трассировки, следовательно программы работают без ошибок.