Министерство 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 21ВВ1.2

Брянцев Артём  
Сущёв Максим  
Тюрин Владислав

Приняли:

Митрохин М. А.

Юрова О. В.

Пенза 2022

**Название**

Бинарное дерево поиска.

**Лабораторное задание.**

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

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

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

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

**Листинг**

lb4.cpp

#include <iostream>

using namespace std;

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

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

void print\_tree(struct Node\* r, int l);

void search\_elem(struct Node\* r, int data);

int counter(Node\* root, int E);

int main()

{

setlocale(LC\_ALL, "");

srand(time(NULL));

int D, number;

struct Node\* root = NULL;

cout << "Введите размер дерева: ";

cin >> D;

while (D) {

root = CreateTree(root, root, rand() % 100);

D--;

}

cout << "Построение дерева окончено" << endl << endl;

print\_tree(root, 0);

cout << endl << "Введите элемент для поиска и подсчёта: ";

cin >> D;

search\_elem(root, D);

number = counter(root, D);

if (number)

cout << "Кол-во искомых элементов: " << number;

else

cout << "Ничего не найдено";

return 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)

{

cout << "Ошибка выделения памяти";

exit(0);

}

r->left = NULL;

r->right = NULL;

r->data = data;

if (root == NULL) return r;

if (data > root->data) root->right = r;

else root->left = r;

return r;

}

if (data > r->data)

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

else

CreateTree(r, r->left, 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++)

cout << (" ");

cout << r->data << endl;

print\_tree(r->left, l + 1);

}

void search\_elem(struct Node\* r, int data) //поиск по заданному элементу

{

if (r == NULL)

return;

if (r->data < data) {

search\_elem(r->right, data);

}

else if (r->data > data) {

search\_elem(r->left, data);

}

if (r->data == data)

cout << r->data << endl;

}

int counter(Node\* root, int E) //счётчик

{

if (root == 0) return 0;

return (root->data == E) + counter(root->left, E) + counter(root->right, E);

}

lb4.3.cpp

# include <iostream>

using namespace std;

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

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

void print\_tree(struct Node\* r, int l);

void search\_elem(struct Node\* r, int data, int counter);

int checker(Node\* root, int E);

int main()

{

setlocale(LC\_ALL, "");

srand(time(NULL));

int D, number;

struct Node\* root = NULL;

cout << "Введите размер дерева: ";

cin >> D;

while (D) {

do {

number = rand() % 100;

} while(checker(root, number));

root = CreateTree(root, root, number);

D--;

}

cout << "Построение дерева окончено" << endl << endl;

print\_tree(root, 0);

cout << endl << "Введите элемент для поиска: ";

cin >> D;

search\_elem(root, D, 1);

return 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)

{

cout << "Ошибка выделения памяти";

exit(0);

}

r->left = NULL;

r->right = NULL;

r->data = data;

if (root == NULL) return r;

if (data > root->data) root->right = r;

else root->left = r;

return r;

}

if (data > r->data)

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

else

CreateTree(r, r->left, 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++)

cout << (" ");

cout << r->data << endl;

print\_tree(r->left, l + 1);

}

void search\_elem(struct Node\* r, int data, int counter) //поиск по заданному элементу

{

if (r == NULL)

return;

if (r->data < data) {

cout << "right ";

search\_elem(r->right, data, counter + 1);

}

else if (r->data > data) {

cout << "left ";

search\_elem(r->left, data, counter + 1);

}

if (r->data == data)

cout << endl << r->data << endl << counter << " - уровень" << endl;

}

int checker(Node\* root, int E)

{

if (root == 0) return 0;

return (root->data == E) + checker(root->left, E) + checker(root->right, E);

}

**Результат работы программы**

Результаты работы программы (задание 1 и 2) показаны на рисунке 1.

Изображение выглядит как текст

Автоматически созданное описание**Рисунок 1 — Результаты работы программы**

Результаты работы программы (задание 3) показаны на рисунке 2.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 2 — Результаты работы программы**

### Выводы

Сложность – у функции поиска логарифмическая O(log n).

Бинарное дерево - иерархическая структура данных, в которой каждый узел имеет не более двух потомков. Как правило, первый называется родительским узлом, а дети называются левым и правым наследниками. Обычно слева находятся лепестки с меньшим значением по отношению к узлу, из которого они выходят, а справа большие.

Рекурсивные функции – функции, вызывающие сами себя.