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 22ВВВ1:

Ганин И.Р.

Курушин Я.С.

Приняли:

К.т.н, доцент Акифьев И. В.

К.т.н, доцент Юрова О. В.

Пенза 2023

**Цель работы**

Изучить бинарное дерево поиска. Реализовать функции: построение, создание, вывод дерева на экран и поиск элемента.

**Лабораторное задание**

1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
2. Реализовать функцию подсчета 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**

**Теоретическая часть**

Бинарные деревья – это деревья, у каждого узла которого возможно наличие

только двух сыновей. Двоичные деревья являются упорядоченными.

В качестве информации в дереве хранятся целые числа.

Обращение к дереву и его элементам осуществляется посредством указателей:

Так как деревья по своей сути являются рекурсивными структурами данных, то и

большинство функций, работающих с деревьями, рекурсивны.

**Практическая часть**

Основное задание:

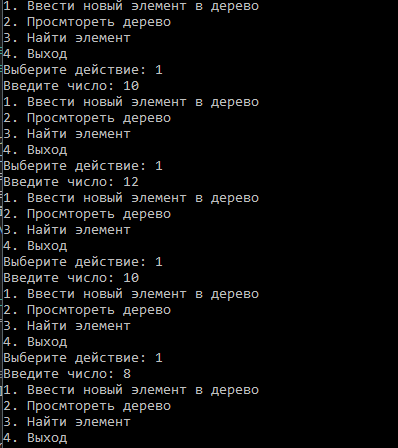


Рисунок 1 – ввели 4 элемента

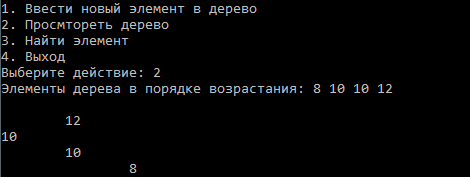


Рисунок 2 – вывели дерево

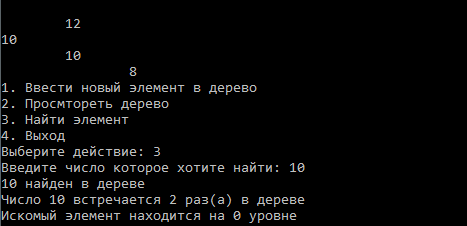


Рисунок 3 – нашли элемент 10

Дополнительное задание:

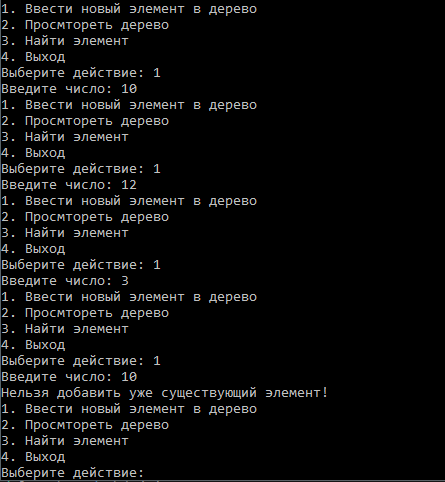
\

Рисунок 4 – ввели 4 элемента

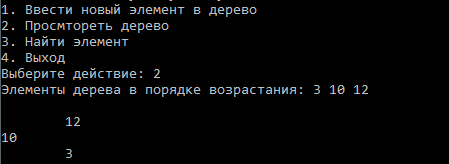


Рисунок 5 – вывели дерево

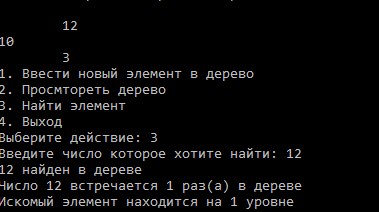


Рисунок 6 – выполнили поиск элемента

Оценили сложность алгоритма. Из-за рекурсии сложность будет равна О(log(n)).

**Вывод**: В ходе данной лабораторной работы было изучено бинарное дерево поиска. Были реализованы функции: построение, создание, вывод дерева на экран и поиск элемента.

**Приложение А  
Листинг**

**Файл Source.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

// Структура для узла бинарного дерева

typedef struct Node {

int data;

struct Node\* left;

struct Node\* right;

} Node;

int p = 0;

// Функция для создания нового узла

Node\* createNode(int data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

// Функция для вставки нового узла в дерево

Node\* insert(Node\* root, int data) {

if (root == NULL) {

root = createNode(data);

}

else if (data <= root->data) {

root->left = insert(root->left, data);

}

if (data > root->data) {

root->right = insert(root->right, data);

}

//else { // Если элемент уже существует в дереве, ничего не делаем (е сли не закоментировать этот else, то повторов будет всего 1 - что логично)

// printf("Нельзя добавить уже существующий элемент!\n");

// free(createNode(data));

// return root;

//}

return root;

}

// Функция для поиска элемента в дереве

Node\* search(Node\* root, int data) {

if (root == NULL || root->data == data) {

return root;

}

else if (data < root->data) {

p++;

return search(root->left, data);

}

else {

p++;

return search(root->right, data);

}

}

// Функция для печати элементов дерева в порядке возрастания

void printInOrder(Node\* root) {

if (root != NULL) {

printInOrder(root->left);

printf("%d ", root->data);

printInOrder(root->right);

}

}

void print\_tree(Node\* root, int l)

{

if (root == NULL)

{

return;

}

print\_tree(root->right, l + 1);

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

{

printf("\t");

}

printf("%d\n", root->data);

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

}

int countOccurrences(Node\* root, int data) {

if (root == NULL) {

return 0;

}

if (root->data == data) {

return 1 + countOccurrences(root->left, data) + countOccurrences(root->right, data);

}

else if (data < root->data) {

return countOccurrences(root->left, data);

}

else {

return countOccurrences(root->right, data);

}

}

void main() {

setlocale(LC\_ALL, "RUS");

Node\* root = NULL;

Node\* searchResult = NULL;

int choice, data, searchData, occurrences = 0;

do {

printf("1. Ввести новый элемент в дерево\n");

printf("2. Просмтореть дерево\n");

printf("3. Найти элемент\n");

printf("4. Выход\n");

printf("Выберите действие: ");

scanf("%d", &choice);

switch (choice) {

case 1: // Вставка элементов в дерево

printf("Введите число: ");

scanf("%d", &data);

root = insert(root, data);

break;

case 2:

printf("Элементы дерева в порядке возрастания: ");

printInOrder(root);

printf("\n\n");

print\_tree(root, 0);

break;

case 3: // Поиск элемента в дереве

printf("Введите число которое хотите найти: ");

scanf("%d", &searchData);

searchResult = search(root, searchData);

if (searchResult != NULL) {

printf("%d найден в дереве\n", searchData);

}

else {

printf("%d не найден в дереве\n", searchData);

}

occurrences = countOccurrences(root, searchData);

printf("Число %d встречается %d раз(а) в дереве\n", searchData, occurrences);

printf("Искомый элемент находится на %d уровне\n", p);

p=0;

break;

case 4:

system("cls");

printf("До свиания!\n");

break;

default:

system("cls");

printf("Некоректный выбор!\n");

break;

}

} while (choice != 4);

}

**Файл Source2.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

// Структура для узла бинарного дерева

typedef struct Node {

int data;

struct Node\* left;

struct Node\* right;

} Node;

int p = 0;

// Функция для создания нового узла

Node\* createNode(int data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

// Функция для вставки нового узла в дерево

Node\* insert(Node\* root, int data) {

if (root == NULL) {

root = createNode(data);

}

else if (data < root->data) {

root->left = insert(root->left, data);

}

else if (data > root->data) {

root->right = insert(root->right, data);

}

else { // Если элемент уже существует в дереве, ничего не делаем (е сли не закоментировать этот else, то повторов будет всего 1 - что логично)

printf("Нельзя добавить уже существующий элемент!\n");

free(createNode(data));

return root;

}

return root;

}

// Функция для поиска элемента в дереве

Node\* search(Node\* root, int data) {

if (root == NULL || root->data == data) {

return root;

}

else if (data < root->data) {

p++;

return search(root->left, data);

}

else {

p++;

return search(root->right, data);

}

}

// Функция для печати элементов дерева в порядке возрастания

void printInOrder(Node\* root) {

if (root != NULL) {

printInOrder(root->left);

printf("%d ", root->data);

printInOrder(root->right);

}

}

void print\_tree(Node\* root, int l)

{

if (root == NULL)

{

return;

}

print\_tree(root->right, l + 1);

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

{

printf("\t");

}

printf("%d\n", root->data);

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

}

int countOccurrences(Node\* root, int data) {

if (root == NULL) {

return 0;

}

if (root->data == data) {

return 1 + countOccurrences(root->left, data) + countOccurrences(root->right, data);

}

else if (data < root->data) {

return countOccurrences(root->left, data);

}

else {

return countOccurrences(root->right, data);

}

}

void main() {

setlocale(LC\_ALL, "RUS");

Node\* root = NULL;

Node\* searchResult = NULL;

int choice, data, searchData, occurrences = 0;

do {

printf("1. Ввести новый элемент в дерево\n");

printf("2. Просмтореть дерево\n");

printf("3. Найти элемент\n");

printf("4. Выход\n");

printf("Выберите действие: ");

scanf("%d", &choice);

switch (choice) {

case 1: // Вставка элементов в дерево

printf("Введите число: ");

scanf("%d", &data);

root = insert(root, data);

break;

case 2:

printf("Элементы дерева в порядке возрастания: ");

printInOrder(root);

printf("\n\n");

print\_tree(root, 0);

break;

case 3: // Поиск элемента в дереве

printf("Введите число которое хотите найти: ");

scanf("%d", &searchData);

searchResult = search(root, searchData);

if (searchResult != NULL) {

printf("%d найден в дереве\n", searchData);

}

else {

printf("%d не найден в дереве\n", searchData);

}

occurrences = countOccurrences(root, searchData);

printf("Число %d встречается %d раз(а) в дереве\n", searchData, occurrences);

printf("Искомый элемент находится на %d уровне\n", p);

p=0;

break;

case 4:

system("cls");

printf("До свиания!\n");

break;

default:

system("cls");

printf("Некоректный выбор!\n");

break;

}

} while (choice != 4);

}