Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра “Вычислительная техника”

**ОТЧЕТ**

по лабораторной работе №4

по дисциплине «Логика и основы алгоритмизации в инженерных задачах»

# на тему: «Бинарное дерево поиска»

Выполнили студенты группы 22ВВВ3:

Куракин Н.Н.

Майоров Н.А.

Матюшин К. М.

Приняли:

Юрова О.В.

Акифьев И.В.

Пенза 2023

**Название**

Бинарное дерево поиска.

**Цель работы**

Изучить реализацию бинарного дерева поиска и алгоритм поиска вводимого с клавиатуры значения.

**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.

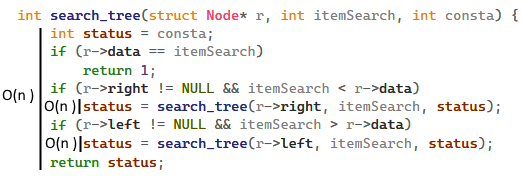
*Задание 3:*

Изменить функцию добавления элементов для исключения добавления одинаковых символов.

*Задание 4:*

Оценить сложность процедуры поиска по значению в бинарном дереве.

**Ход работы**

Задание 4:  


Сложность алгоритма поиска по значению в бинарном дереве – O(n)

**Программы**

**Задание 1-2:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <locale.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

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 searchTree(struct Node\* r, int itemSearch, int const\_1) {

int status = const\_1;

if (r == NULL)

return 0;

if (r->data == itemSearch) {

status++;

if (r->right != NULL)

status = searchTree(r->right, itemSearch, status);

return status;

}

if (r->right != NULL && itemSearch < r->data)

status = searchTree(r->right, itemSearch, status);

if (r->left != NULL && itemSearch > r->data)

status = searchTree(r->left, itemSearch, status);

return status;

}

int main() {

setlocale(LC\_ALL, "");

int D, start = 1;

struct Node\* 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);

while (1) {

printf("\nИскать элемент? (0/1) ");

switch (\_getch()) {

case '1':

printf("Поиск элемента...\nВведите искомое число: ");

(void)scanf("%d", &start);

start = searchTree(root, start, 0);

if (start)

printf("Элемент найден, число вхождения элемента в дерево - %d", start);

else

printf("Отсутствует искомый элемент или дерево");

break;

case '0':

return 0;

}

}

}  
**Задание 3-4:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <locale.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

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

if (r != NULL && r->data == data)

return r;

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 search\_tree(struct Node\* r, int itemSearch, int consta) {

int status = consta;

if (r->data == itemSearch)

return 1;

if (r->right != NULL && itemSearch < r->data)

status = search\_tree(r->right, itemSearch, status);

if (r->left != NULL && itemSearch > r->data)

status = search\_tree(r->left, itemSearch, status);

return status;

}

int main() {

setlocale(LC\_ALL, "");

int D, start = 1;

struct Node\* 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);

while (1) {

printf("\nИскать элемент? (0/1) ");

switch (\_getch()) {

case '1':

printf("Поиск элемента...\nВведите искомое число: ");

(void)scanf("%d", &start);

printf("%s", !search\_tree(root, start, 0) ? "Отсутствует искомый элемент или дерево" : "Элемент найден");

break;

case '0':

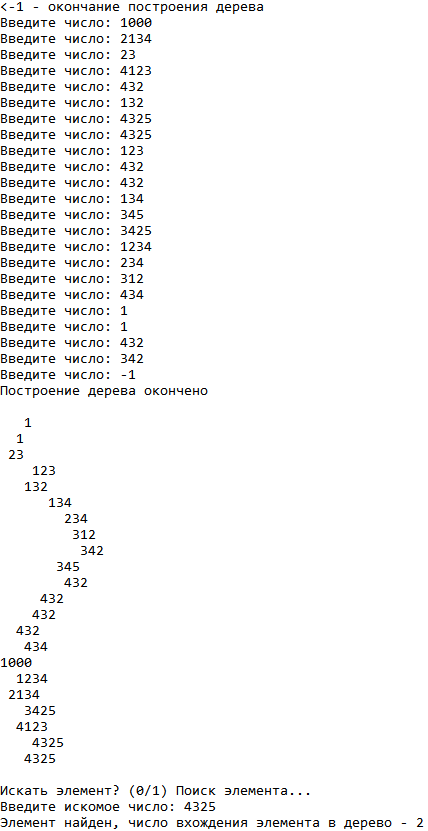
return 0;

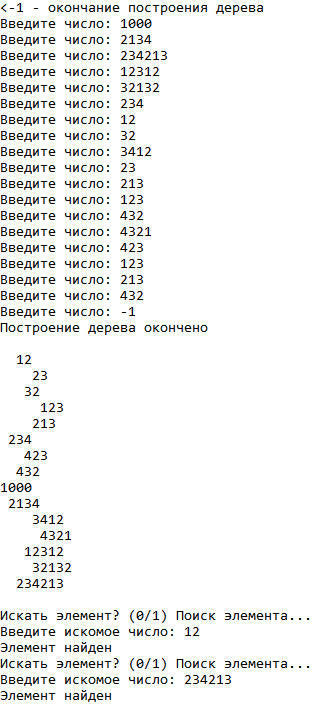
}

}

}

**Результаты выполнения программы:**

Программа 1:  


Программа 2:  


**Вывод:** в ходе выполнения лабораторной работы были получены/повторены навыки реализации бинарного дерева поиска. Была изучена одна из операций над бинарным деревом поиска – поиск узла по значению.