Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра "Вычислительная техника"

Отчёт

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «бинарное дерево поиска»

Вариант 1

Выполнили:

Студенты группы 24ВВВ1

Захаров А.В.

Гурин А.Н.

Приняли:

к.т.н., доцент Юрова О.В.

к.т.н., доцент Деев М.В.

Пенза 2025

**Лабораторное 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.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS #include <stdio.h> #include <stdlib.h> #include <locale.h>

struct Node { int data; struct Node\* left; struct Node\* right; };

struct Node\* ins\_unq(struct Node\* root, int data) { if (root == NULL) { struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node)); newNode->data = data; newNode->left = NULL; newNode->right = NULL; return newNode; }

if (data < root->data) {  
 root->left = ins\_unq(root->left, data);  
}  
else if (data > root->data) {  
 root->right = ins\_unq(root->right, data);  
}  
return root;

}

int povtor(struct Node\* root, int key) { if (root == NULL) return 0;

if (root->data == key) return 1;  
else if (key < root->data)  
 return povtor(root->left, key);  
else  
 return povtor(root->right, key);

}

struct Node\* ins\_dup(struct Node\* root, int data) { if (root == NULL) { struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node)); newNode->data = data; newNode->left = NULL; newNode->right = NULL; return newNode; }

if (data < root->data) {  
 root->left = ins\_dup(root->left, data);  
}  
else {  
 root->right = ins\_dup(root->right, data);  
}  
return root;

}

int count\_dup(struct Node\* root, int key) { if (root == NULL) return 0;

int count = 0;  
if (root->data == key) count = 1;  
  
count += count\_dup(root->left, key);  
count += count\_dup(root->right, key);  
  
return count;

}

void print\_tree(struct Node\* root, int level) { if (root == NULL) return;

print\_tree(root->right, level + 1);  
for (int i = 0; i < level; i++) printf(" ");  
printf("%d\n", root->data);  
print\_tree(root->left, level + 1);

}

void free\_tree(struct Node\* root) { if (root == NULL) return; free\_tree(root->left); free\_tree(root->right); free(root); }

int main() { setlocale(LC\_ALL, ""); struct Node\* root = NULL; int choice, data;

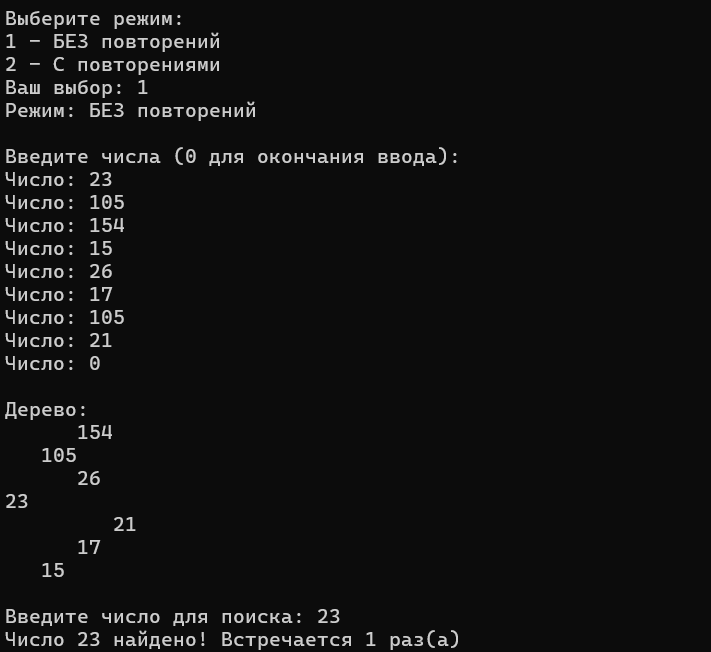
struct Node\* (\*insert\_func)(struct Node\*, int) = NULL;  
int (\*count\_func)(struct Node\*, int) = NULL;

while(1) {  
 printf("Выберите режим:\n");  
 printf("1 - БЕЗ повторений\n");  
 printf("2 - С повторениями\n");  
 printf("Ваш выбор: ");  
 scanf("%d", &choice);  
  
 switch (choice) {  
 case 1:  
 insert\_func = ins\_unq;  
 count\_func = povtor;  
 printf("Режим: БЕЗ повторений\n");  
 break;  
 case 2:  
 insert\_func = ins\_dup;  
 count\_func = count\_dup;  
 printf("Режим: С повторениями\n");  
 break;  
 default:  
 printf("Неверный выбор! Попробуйте снова.\n\n");  
 continue;  
 }  
 break;  
}  
  
printf("\nВведите числа (0 для окончания ввода):\n");  
while (1) {  
 printf("Число: ");  
 scanf("%d", &data);  
 if (data == 0) break;  
 if (data < 0) {  
 printf("Отрицательные числа не допускаются!\n");  
 continue;  
 }  
 root = insert\_func(root, data);  
}  
  
printf("\nДерево:\n");  
print\_tree(root, 0);  
  
int search\_key;  
printf("\nВведите число для поиска: ");  
scanf("%d", &search\_key);  
  
int count = count\_func(root, search\_key);  
if (count > 0) {  
 printf("Число %d найдено! Встречается %d раз(а)\n", search\_key, count);  
}  
else {  
 printf("Число %d не найдено\n", search\_key);  
}  
  
free\_tree(root);  
return 0;

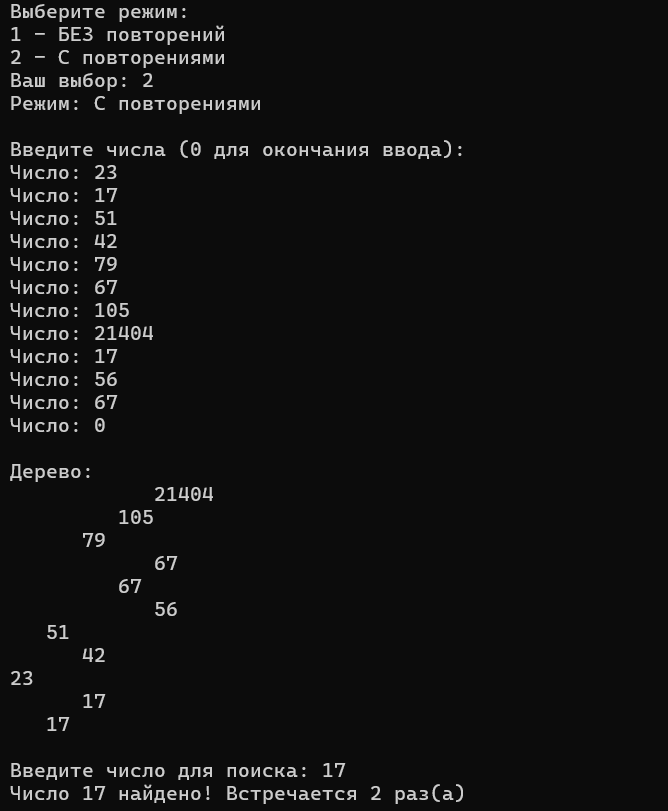
}

**Результат работы программ**

Результат работы программы представлен на рисунках 1 - 4.

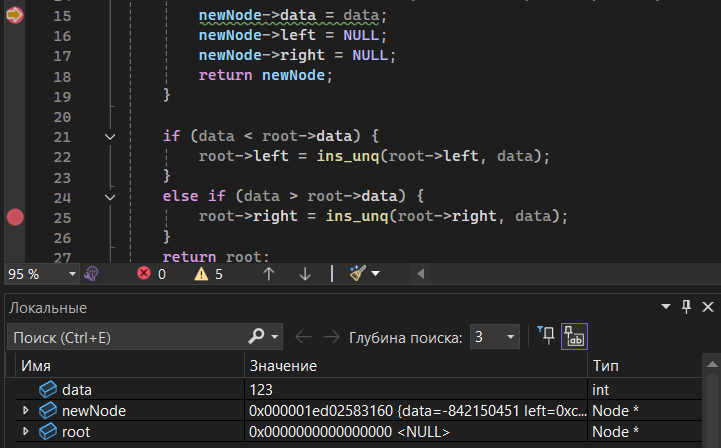


*Рисунок 1 - результат бинарного дерева поиска без повторений*

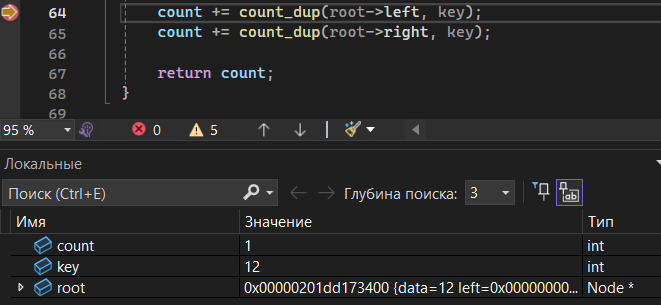


*Рисунок 2 - результат бинарного дерева поиска с повторениями*

**Протокол трассировки программы**



*Рисунок 3 – добавление элемента*



*Рисунок 4 – подсчёт количества вхождений*

**Вывод**

В ходе лабораторной работы была успешно реализована программа для работы с бинарным деревом поиска в двух режимах: с дубликатами и без. Все основные операции (добавление, поиск, подсчёт вхождений) работают корректно. Программа наглядно демонстрирует разницу в сложности алгоритмов между разными типами деревьев. Получены практические навыки работы с рекурсивными структурами данных и динамической памятью в языке СИ.