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 21вв1.1:**

Федулов Е.Д.

Жбанников Д.Н.

**Приняли:**

Юрова О.В.

Акифьев И.В.

Пенза, 2022

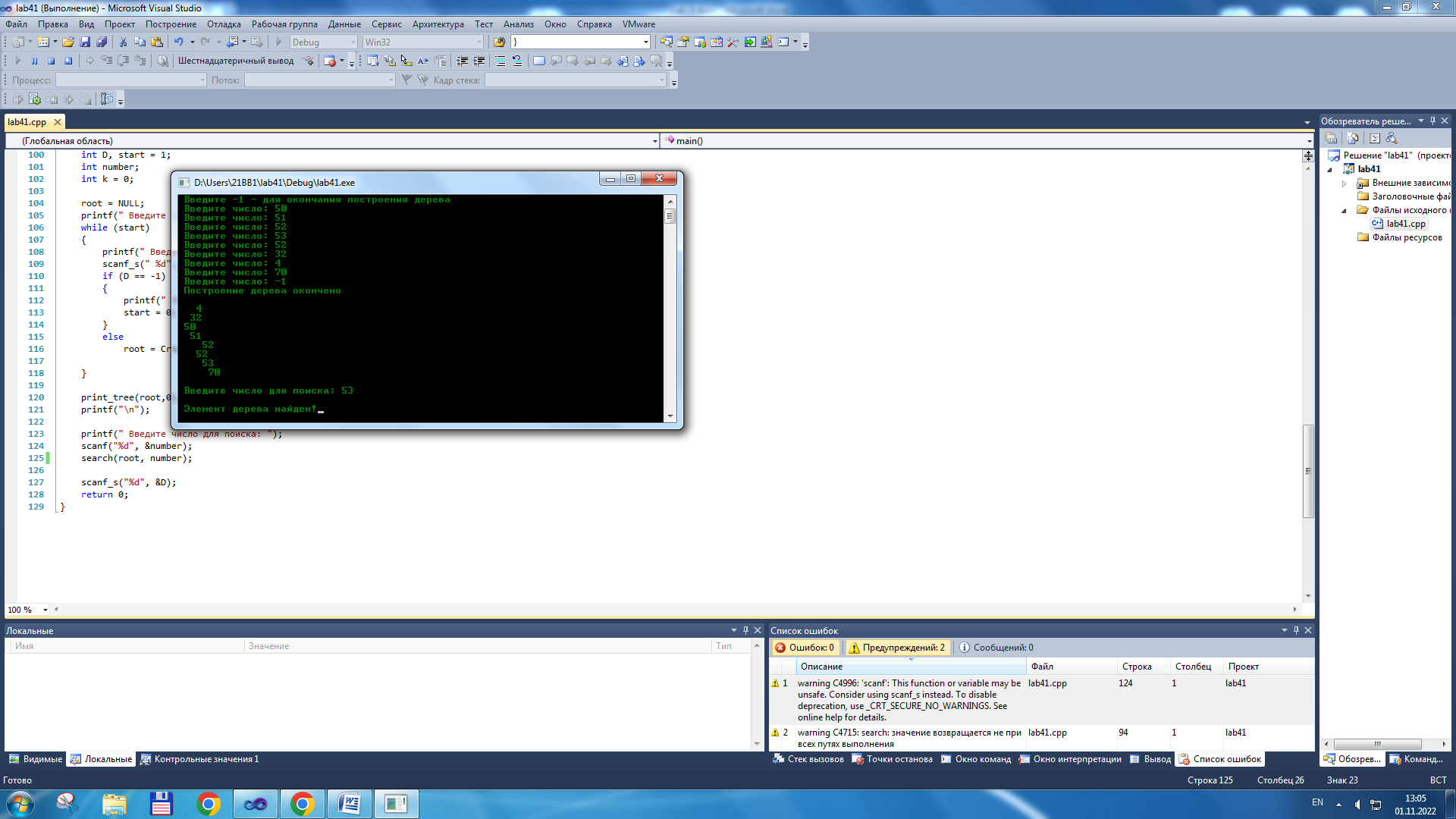
**Цель работы:**ознакомиться с бинарными деревьями, освоить основные виды манипуляций с ними в СИ.

**Задания:**

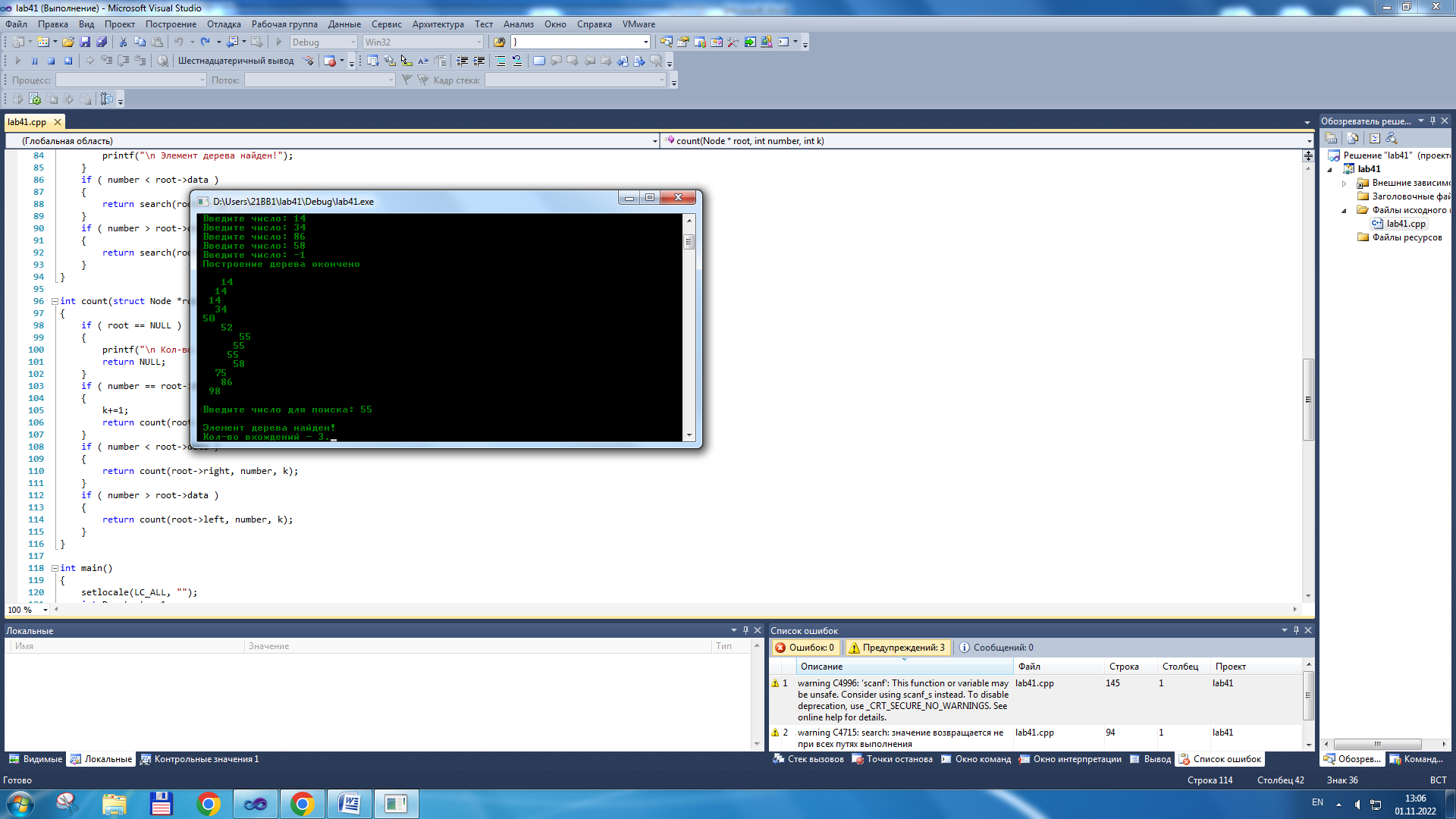
1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
2. Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.
3. Изменить функцию добавления элементов для исключения добавления одинаковых символов.
4. Оценить сложность процедуры поиска по значению в бинарном дереве.

**Ход работы**

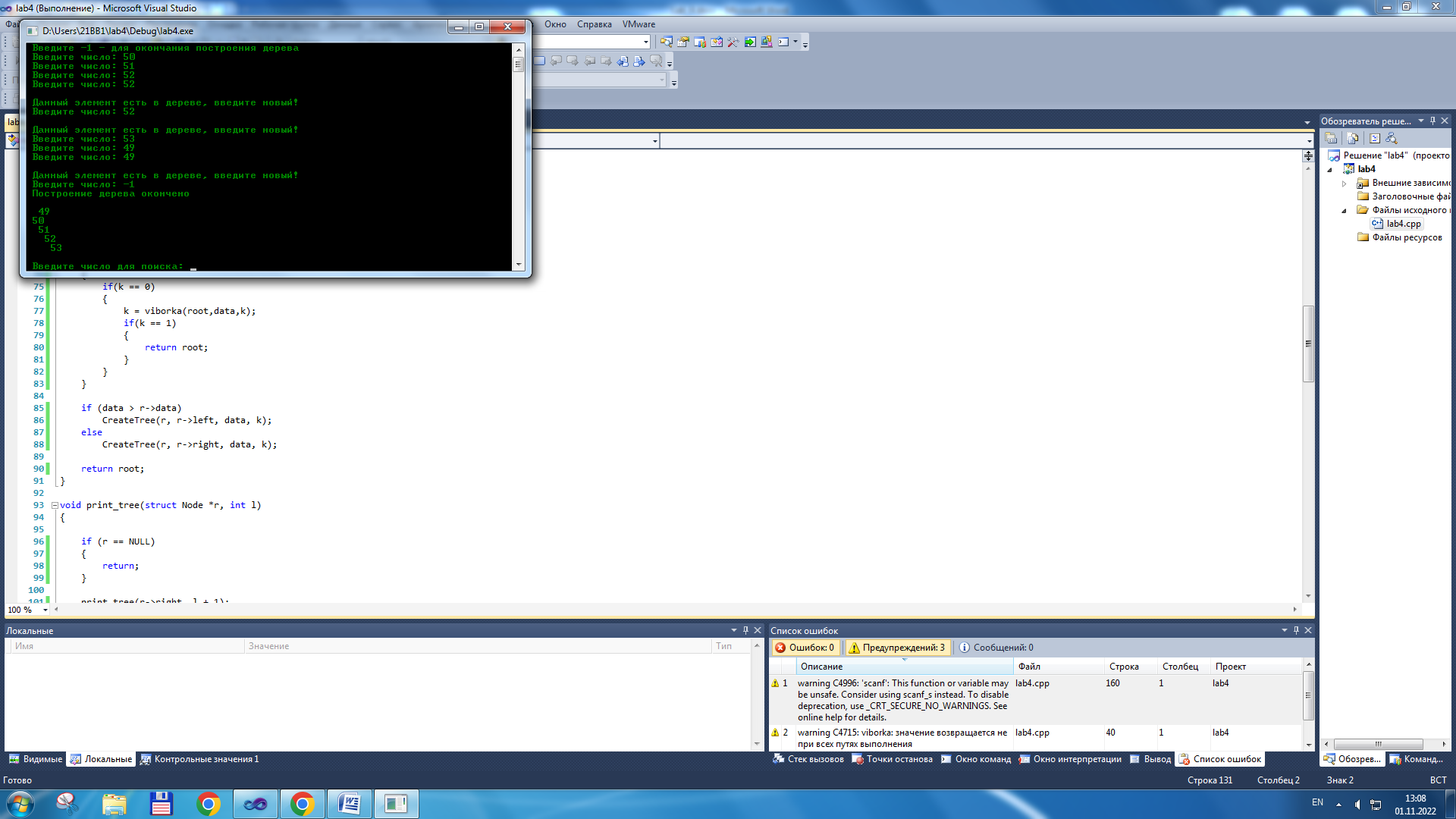
**Результат программы для первого задания:**

****

**Результат программы для второго задания:**



**Результат программы для третьего задания:**

****

**Задание 4:**

Сложность процедуры поиска в бинарном дереве -  **O(log n)**

**Вывод:** в лабораторной работе научились писать программы с применением алгоритмов для работы с бинарными деревьями на языке Си.

**Листинг:**

**Для задания 1 и 2:**

#include<stdio.h>

#include<ctype.h>

#include<string.h>

#include<stdlib.h>

#include<iostream>

usingnamespacestd;

structNode{

intdata;

structNode\*left;

structNode\*right;

};

structNode\*root;

structNode\*CreateTree(structNode\*root,structNode\*r,intdata,intk)

{

if(r==NULL)

{

r=(structNode\*)malloc(sizeof(structNode));

if(r==NULL)

{

printf(" Ошибка выделения памяти");

exit(0);

}

r->left=NULL;

r->right=NULL;

r->data=data;

if(root==NULL)

{

returnr;

}

if(data>root->data)

{

root->left=r;

}

else

{

root->right=r;

}

returnr;

}

if(data>r->data)

CreateTree(r,r->left,data,k);

else

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

returnroot;

}

voidprint\_tree(structNode\*r,intl)

{

if(r==NULL)

{

return;

}

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

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

{

printf(" ");

}

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

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

}

structNode\*search(structNode\*root,intnumber)

{

if(root==NULL)

{

printf("\n Элемент не найден!");

returnNULL;

}

if(number==root->data)

{

printf("\n Элемент дерева найден!");

}

if(number<root->data)

{

returnsearch(root->right,number);

}

if(number>root->data)

{

returnsearch(root->left,number);

}

}

intcount(structNode\*root,intnumber,intk)

{

if(root==NULL)

{

printf("\n Кол-во вхождений - %d.",k);

returnNULL;

}

if(number==root->data)

{

k+=1;

returncount(root->right,number,k);

}

if(number<root->data)

{

returncount(root->right,number,k);

}

if(number>root->data)

{

returncount(root->left,number,k);

}

}

intmain()

{

setlocale(LC\_ALL,"");

intD,start=1;

intnumber;

intk=0;

root=NULL;

printf(" Введите -1 - для окончания 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,k);

}

print\_tree(root,0);

printf("\n");

printf(" Введите число для поиска: ");

scanf("%d",&number);

search(root,number);

count(root,number,k);

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

return0;

}

**Задание 3:**

#include<stdio.h>

#include<ctype.h>

#include<string.h>

#include<stdlib.h>

#include<iostream>

usingnamespacestd;

structNode{

intdata;

structNode\*left;

structNode\*right;

};

structNode\*root;

intviborka(structNode\*root,intdata,intk)

{

if(root==NULL)

{

k=0;

return(k);

}

if(data==root->data)

{

printf("\n Данный элемент есть в дереве, введите новый!\n");

k=1;

return(k);

}

if(data<root->data)

{

returnviborka(root->right,data,k);

}

if(data>root->data)

{

returnviborka(root->left,data,k);

}

}

structNode\*CreateTree(structNode\*root,structNode\*r,intdata,intk)

{

if(r==NULL)

{

r=(structNode\*)malloc(sizeof(structNode));

if(r==NULL)

{

printf(" Ошибка выделения памяти");

exit(0);

}

r->left=NULL;

r->right=NULL;

r->data=data;

if(root==NULL)

{

returnr;

}

if(data>root->data)

{

root->left=r;

}

else

{

root->right=r;

}

returnr;

}

else

{

if(k==0)

{

k=viborka(root,data,k);

if(k==1)

{

returnroot;

}

}

}

if(data>r->data)

CreateTree(r,r->left,data,k);

else

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

returnroot;

}

voidprint\_tree(structNode\*r,intl)

{

if(r==NULL)

{

return;

}

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

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

{

printf(" ");

}

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

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

}

structNode\*search(structNode\*root,intnumber)

{

if(root==NULL)

{

printf("\n Элемент не найден!");

returnNULL;

}

if(number==root->data)

{

printf("\n Элемент дерева найден!");

}

if(number<root->data)

{

returnsearch(root->right,number);

}

if(number>root->data)

{

returnsearch(root->left,number);

}

}

intmain()

{

setlocale(LC\_ALL,"");

intD,start=1;

intnumber;

intk=0;

root=NULL;

printf(" Введите -1 - для окончания 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,k);

}

print\_tree(root,0);

printf("\n");

printf(" Введите число для поиска: ");

scanf("%d",&number);

search(root,number);

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

return0;

}