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»

Выполнил:

студент группы 21ВВ2

Шилов Алексей

Приняли:

д.т.н. Митрохин М.А.

к.т.н. Юрова О.В.

Пенза 2022

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.

Сложность поиска в бинарном дереве в среднем равно O(log n), в худшем случае O(n).

Листинг

#include <iostream>

#include <time.h>

#include <stdlib.h>

using namespace std;

int tabs = 0;

int Vcount = 0;

struct Tree {

int Data;

struct Tree\* left;

struct Tree\* right;

};

void Add(int x, Tree\*& tree) {

if (tree == NULL) { //Создание корневого элемента

tree = new Tree;

tree->Data = x;

tree->left = NULL;

tree->right = NULL;

return;

}

else if(x < tree->Data) {

Add(x, tree->left);

}

else if(x > tree->Data){

Add(x, tree->right);

}

}

void TPrint(struct Tree\* tree) {

if (tree != NULL) {

tabs += 3;

TPrint(tree->left);

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

cout << " ";

cout << tree->Data << endl;

TPrint(tree->right);

tabs -= 3;

}

}

void TObh(Tree\*& tree) { //Прямой обход

if (tree != NULL) {

cout << tree->Data << endl;

TObh(tree->left);

TObh(tree->right);

}

}

void AddElem(int x, Tree\*& tree) {

if (tree == NULL) {

tree = new Tree;

tree->Data = x;

tree->left = NULL;

tree->right = NULL;

}

else {

if (x < tree->Data) {

AddElem(x, tree->left);

}

else if (x > tree->Data) {

AddElem(x, tree->right);

}

else {

cout << "Element uzhe suschestvuet" << endl;

}

}

}

Tree\* DelElem(int x, Tree\*& tree) {

if (tree != NULL) {

if (x == tree->Data) {

Tree\* tmp;

if (tree->right == NULL)

tmp = tree->left;

else {

Tree\* ptr = tree->right;

if (ptr->left == NULL) {

ptr->left = tree->left;

tmp = ptr;

}

else {

Tree\* pmin = ptr->left;

while (pmin->left != NULL) {

ptr = pmin;

pmin = ptr->left;

}

ptr->left = pmin->right;

pmin->left = tree->left;

pmin->right = tree->right;

tmp = pmin;

}

}

delete tree;

return tmp;

}

else if (x < tree->Data)

tree->left = DelElem(x, tree->left);

else

tree->left = DelElem(x, tree->right);

return tree;

}

}

void TDel(Tree\* tree) {

if (tree == NULL)

return;

TDel(tree->left);

TDel(tree->right);

delete tree;

return;

}

void isEmpty(Tree\*& tree) {

if (tree == NULL)

{

cout << "Derevo pustoe" << endl;

}

else

{

cout << "Derevo ne pustoe" << endl;

}

}

void Find(int x, Tree\*& tree) {

if (tree != NULL) {

if (x == tree->Data) {

Vcount++;

cout << tree->Data << " ";

Find(x, tree->right);

}

else if (x < tree->Data)

Find(x, tree->left);

else if (x > tree->Data)

Find(x, tree->right);

}

}

void PV(int x, Tree\*& tree) {

if (tree != NULL) {

if (x == tree->Data) {

Vcount++;

PV(x, tree->right);

}

else if (x < tree->Data)

PV(x, tree->left);

else if (x > tree->Data)

PV(x, tree->right);

}

}

int main()

{

srand(time(NULL));

struct Tree\* Root = NULL;

int size = 0;

int x = 0, z = 0;

cout << "Vvedite kol-vo popitok generacii: ";

cin >> size;

cout << endl;

isEmpty(Root);

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

{

Add(rand() % 100, Root);

}

isEmpty(Root);

cout << "Derevo:" << endl;

TPrint(Root);

cout << endl;

cout << "Viberite deistvie:" << endl;

cout << " 1. Dobavlenie" << endl;

cout << " 2. Poisk" << endl;

cout << " 3. Podschet vhozdenii" << endl;

do {

cin >> z;

switch (z) {

case 1:

cout << "Dobavlenie elementa: ";

cin >> x;

AddElem(x, Root);

cout << endl;

cout << "Derevo:" << endl;

TPrint(Root);

cout << endl;

break;

case 2:

cout << "Vvedite chislo: ";

cin >> x;

Vcount = 0;

Find(x, Root);

if (Vcount == 0)

cout << "Element ne naiden";

cout << endl;

break;

case 3:

cout << "Vvedite chislo: ";

cin >> x;

Vcount = 0;

PV(x, Root);

cout << "Naideno " << Vcount << " vhozdenii" << endl;

break;

}

} while (z != 0);

TDel(Root);

system("pause");

return 0;

}

Результаты

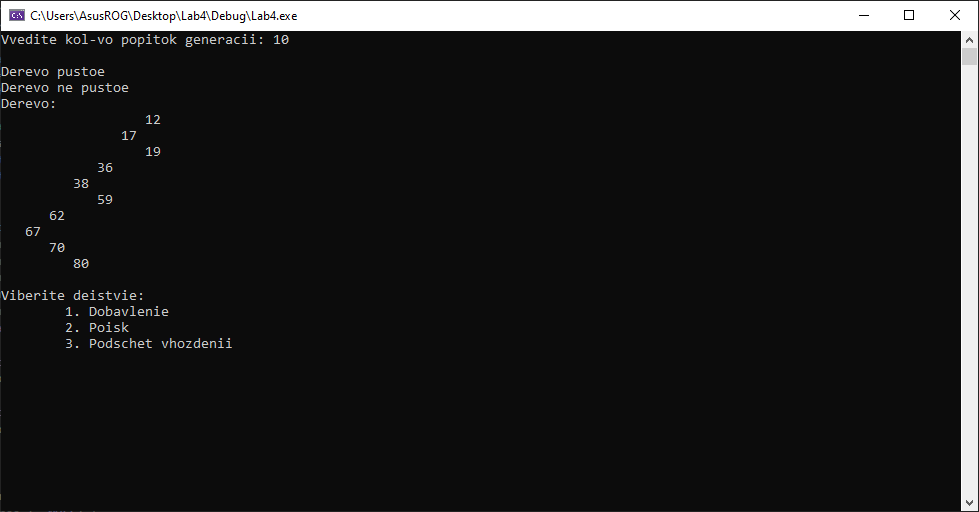


Рисунок 1 – Генерация и вывод бинарного дерева

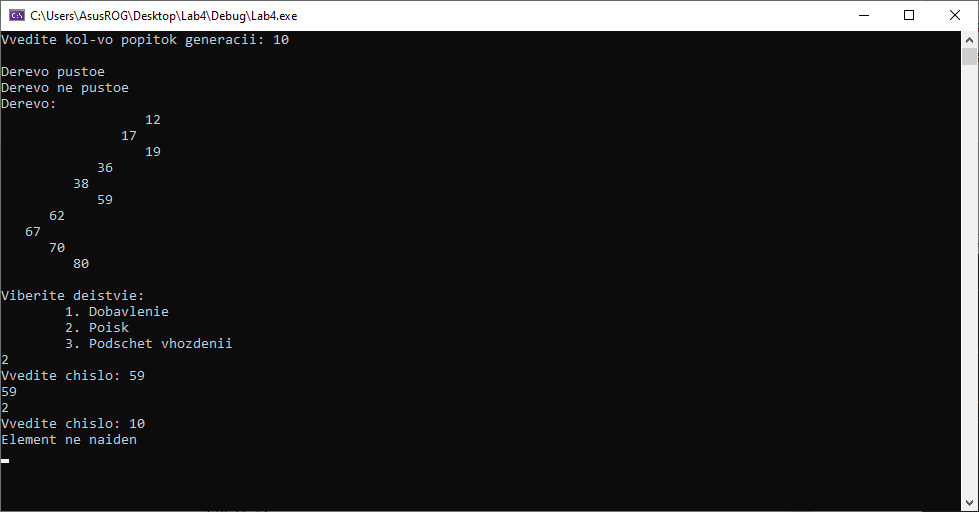


Рисунок 2 – Поиск существующего и не существующего элемента

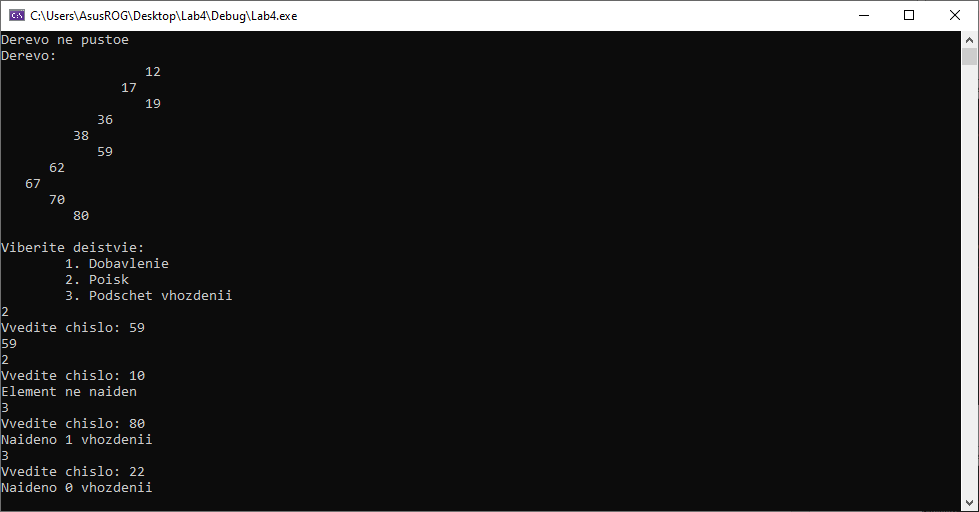


Рисунок 3 – Подсчёт кол-ва вхождений элемента

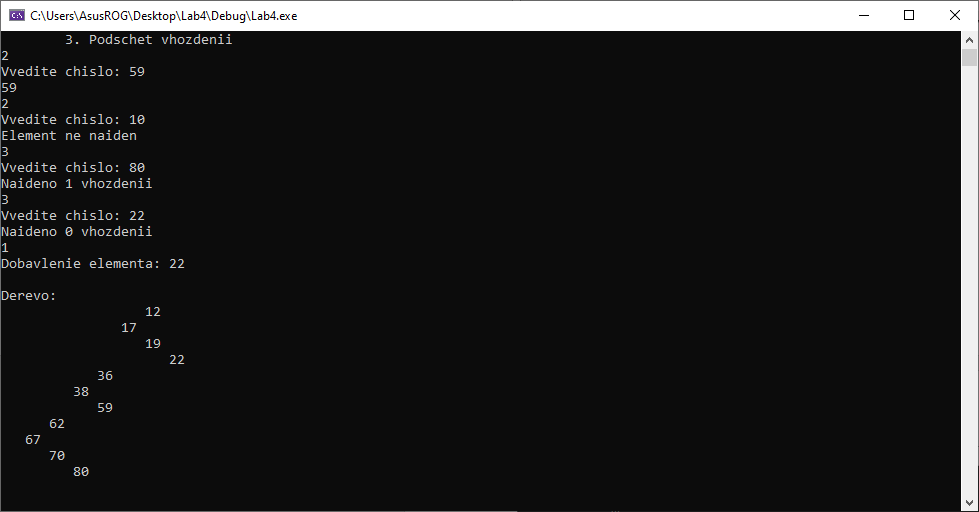


Рисунок 4 – Добавление нового элемента

Вывод

Изучил бинарное дерево поиска, изучил методы создания и работы с бинарным деревом поиска.