МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КІБЕРБЕЗПЕКИ

ЗВІТ ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №4

із дисципліни «Алгоритми та структури даних»

за темою: «Структура даних БІНАРНЕ ДЕРЕВО ПОШУКУ, основні операції»

Виконала студент групи Ін-13

Шеліхов Д.Ю.

Варіант 20

Суми – 2022

ЛАБОРАТОРНА РОБОТА №4

**Мета:** Навчитися використовувати структуру даних бінарне дерево пошуку в розв’язанні завдань.

Завдання:

1. Реалізувати структуру даних "бінарне дерево пошуку", виконавши на ній операції вставки, видалення та пошуку елементів. Написати програму (функцію main), що містить опис бінарного дерева пошуку з заданими операціями.

2. Створити на обраній структурі даних довідник телефонів студентів групи.

3. Продемонструвати виконання всіх запрограмованих операцій.

Виконання:

Бінарне дерево пошуку (BST), яке також називають упорядкованим або відсортованим двійковим деревом, є структурою даних двійкового дерева, особливість якого полягає в тому, що будь-який внутрішній його вузел *х зберігає ключ (значення,, інформацію) більший за всі ключі, що зберігаються в лівому піддереві цього вузла і менший за ключі, що зберігаються у вузлах правого піддерева. Така власт*ивість називається *впорядкованістю ключів у дереві.*

**C++**

#include <iostream>

#include <string>

using namespace std;

struct Tree {

string val;

string name;

Tree\* left;

Tree\* right;

};

int tabs;

Tree\* InsertNode(Tree\* node, string val, string name);

Tree\* DeleteNode(Tree\* node, string val);

void SeachNode(const Tree\* node, string val, string& name);

void ClearNode(Tree\* node);

int c;

int main(void) {

Tree\* tree = NULL;

int fun = -1;

while(fun != 0) {

cout << "0. Exit\n1. Cin element\n2. Delete element\n3. Seach element\n--> "; cin >> fun;

cout << endl;

if (fun == 1) { //ввод

string buf = "1";

string bufName{};

while (buf != "0") {

cout << "Cin id (0 - exit): ";

cin >> buf;

if (buf != "0") {

cout << "Cin name: ";

cin >> bufName;

tree = InsertNode(tree, buf, bufName);

}

}

cout << endl << endl;

}

if (fun == 2) { //удаления

string buf{};

cout << "Cin element to delete: "; cin >> buf;

tree = DeleteNode(tree, buf);

cout << endl << endl;

}

if (fun == 3) {

string buf{}; string name{};

cout << "Cin id to seach: "; cin >> buf;

SeachNode(tree, buf, name);

if(c==1){

cout << "\nThis element in " << buf << " id is " << name;

cout << endl << endl;

c = 0;}

else{

cout << "\nThis element in " << buf << " not found"<< endl << endl;

}}

}

ClearNode(tree); //конец кода, очистка дерева

return 0;

}

//вставка

Tree\* InsertNode(Tree\* node, string val, string name) {

if (node == NULL) {

node = new (std::nothrow) Tree();

if (node != NULL) {

node->val = val;

node->name = name;

node->left = node->right = NULL;

}

return node;

}

if (val < node->val)

node->left = InsertNode(node->left, val, name);

else

node->right = InsertNode(node->right, val, name);

return node;

}

//удаление

Tree\* DeleteNode(Tree\* node, string val) {

if (node == NULL)

return node;

if (val == node->val) {

Tree\* tmp;

if (node->right == NULL)

tmp = node->left;

else {

Tree\* ptr = node->right;

if (ptr->left == NULL) {

ptr->left = node->left;

tmp = ptr;

}

else {

Tree\* pmin = ptr->left;

while (pmin->left != NULL) {

ptr = pmin;

pmin = ptr->left;

}

ptr->left = pmin->right;

pmin->left = node->left;

pmin->right = node->right;

tmp = pmin;

}

}

delete node;

return tmp;

}

else if (val < node->val)

node->left = DeleteNode(node->left, val);

else

node->right = DeleteNode(node->right, val);

return node;

}

void SeachNode(const Tree\* node, string val, string &name) {

if (node->val == val) {

if (c == 0) {

name = node->name;

c++;

}

return;

}

else if (val > node->val) {

SeachNode(node->right, val, name);

}

else if (val < node->val) {

SeachNode(node->left, val, name);

}

return;

}

//удаление всего

void ClearNode(Tree\* node) {

if (node != NULL) {

if (node->left != NULL)

ClearNode(node->left);

if (node->right != NULL)

ClearNode(node->right);

delete node;

}

}}

**Моя консоль**



