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 node {

string key;

struct node \*left, \*right;

};

struct node \*newNode(string item) {

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->key = item;

temp->left = temp->right = NULL;

return temp;

}

void inorder(struct node \*root) {

if (root != NULL) {

inorder(root->left);

cout << root->key << " -> ";

inorder(root->right);

}

};

int search(struct node \*root,std::string k) {

if(k==root->key){

return 1;

}

if (root == NULL){

return 0;}

if(k<root->key){

search(root->left,k);}

else{

search(root->right,k);}

}

struct node \*insert(struct node \*node,string key) {

if (node == NULL) return newNode(key);

if (key < node->key)

node->left = insert(node->left, key);

else

node->right = insert(node->right, key);

return node;

}

struct node \*minValueNode(struct node \*node) {

struct node \*current = node;

while (current && current->left != NULL)

current = current->left;

return current;

}

struct node \*deleteNode(struct node \*root,string key) {

if (root == NULL) return root;

if (key < root->key)

root->left = deleteNode(root->left, key);

else if (key > root->key)

root->right = deleteNode(root->right, key);

else {

if (root->left == NULL) {

struct node \*temp = root->right;

free(root);

return temp;

} else if (root->right == NULL) {

struct node \*temp = root->left;

free(root);

return temp;

}

struct node \*temp = minValueNode(root->right);

root->key = temp->key;

root->right = deleteNode(root->right, temp->key);

}

return root;

}

int main() {

struct node \*root = NULL;

root = insert(root, "380997528538");

root = insert(root, "594359345935");

root = insert(root, "294858392920");

root = insert(root, "495384583848");

root = insert(root, "898345843583");

cout << "Отсортированный обход: ";

inorder(root);

cout << "\nПосле удаления 495384583848\n";

root = deleteNode(root, "495384583848");

cout << "Отсортированный обход: ";

inorder(root);

cout<<endl<<"Поиск нереального числа 9 = "<<search(root,"9")<<endl;

cout<<"Поиск реального числа 380997528538 = "<<search(root,"380997528538")<<endl;

}

**Моя консоль**

