Министерство образования Республики Беларусь

г. Минск

Государственное учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

**Лабораторная работа №7**

**“ Бинарные деревья ”**

**Учебная группа 230501**

Выполнил: Кочеров Роман Сергеевич

Проверил: Селезнев Александр Игоревич

2023 год

**Вариант 4**

1. **Написать программу для создания произвольного бинарного дерева.**
2. **Произвести удаление элементов дерева, выбираемых пользователем.**
3. Написать функцию, которая определяет число вхождений элемента x в бинарное дерево.
4. Добавить визуализацию полученного дерева.

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node\* left;

struct node\* right;

};

struct node\* createNode(int data) {

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

if (newNode == NULL)

{

printf("Error");

exit(1);

}

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

struct node\* insert(struct node\* root, int data) {

if (root == NULL) {

root = createNode(data);

return root;

}

else if (data <= root->data) {

root->left = insert(root->left, data);

}

else {

root->right = insert(root->right, data);

}

return root;

}

void treeprint(struct node\* tree) {

if (tree != NULL) //Пока не встретится пустой узел

{

printf("%d ", tree->data); //Отображаем корень дерева

treeprint(tree->left); //Рекурсивная функция для левого поддерева

treeprint(tree->right); //Рекурсивная функция для правого поддерева

}

}

void poick(struct node\* tree, int \*kolvo\_chicla\_v\_tree, int x) {

if (tree != NULL) //Пока не встретится пустой узел

{

if (tree->data == x)

{

(\*kolvo\_chicla\_v\_tree)++;

}

poick(tree->left, kolvo\_chicla\_v\_tree, x); //Рекурсивная функция для левого поддерева

poick(tree->right, kolvo\_chicla\_v\_tree, x); //Рекурсивная функция для правого поддерева

}

}

struct node\* findMinValueNode(struct node\* root) {

while (root->left != NULL) {

root = root->left;

}

return root;

}

struct node\* deleteNode(struct node\* root, int value) {

if (root == NULL) {

return root;

}

if (value < root->data) {

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

}

else if (value > root->data) {

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

}

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 = findMinValueNode(root->right);

root->data = temp->data;

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

}

return root;

}

int vvod()

{

int var;

while ((scanf\_s("%d", &var) == 0) || getchar() != '\n')

{

printf("wrong input! try again\n");

rewind(stdin);

}

return var;

}

int main() {

struct node\* root = NULL;

int kolvo\_chicel, x, kolvo\_chicla\_v\_tree = 0;

srand(time(NULL));

kolvo\_chicel = 20 + rand() % (30 - 20 + 1);

x = rand() % 100;

//printf("koren - %d\n", x);

root = insert(root, x);

for (int i = 0; i < kolvo\_chicel; i++)

{

x = rand() % 100;

insert(root, x);

}

printf("Tree traversal in prefix form:\n");

treeprint(root);

printf("\nWrite elemtnr for delete: \n");

x=vvod();

root = deleteNode(root, x);

treeprint(root);

printf("\nWrite your number: ");

x = vvod();

poick(root, &kolvo\_chicla\_v\_tree, x);

printf("The number of occurrences of a number in the tree: %d\n", kolvo\_chicla\_v\_tree);

return 0;

}

}