Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики

и радиоэлектроники»

Специальность «Программная инженерия»

Кафедра инженерной психологии и эргономики

Учебная дисциплина «Основы алгоритмизации и программирования»

Отчет

по лабораторной работе №14

«Бинарное дерево»

Вариант 26

Подготовила: Сёмчена Д. В.

Проверил: Усенко Ф. В.

Минск 2025

**Цель работы**: сформировать знания и умения по работе с подпрограммами, приобрести навыки написания программ с использованием бинарных деревьев.

Задание: Построить дерево поиска с заданными числовыми значениями. С использованием операций Addr и Father найти узел с заданным значением и удалить из дерева поиска отца этого узла.

Листинг кода:

#include <iostream>

#include<Windows.h>

using namespace std;

struct TreeNode {

int value;

TreeNode\* left;

TreeNode\* right;

TreeNode(int val) : value(val), left(nullptr), right(nullptr) {}

};

// Вставка

TreeNode\* insert(TreeNode\* root, int val) {

if (!root)

return new TreeNode(val);

if (val < root->value)

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

else

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

return root;

}

// Обход

void inOrder(TreeNode\* root) {

if (!root) return;

inOrder(root->left);

cout << root->value << " ";

inOrder(root->right);

}

// Поиск

TreeNode\* Addr(TreeNode\* root, int val) {

if (!root) return nullptr;

if (root->value == val) return root;

if (val < root->value) return Addr(root->left, val);

else return Addr(root->right, val);

}

// Поиск отца

TreeNode\* Father(TreeNode\* root, int val, TreeNode\* parent = nullptr) {

if (!root) return nullptr;

if (root->value == val) return parent;

if (val < root->value) return Father(root->left, val, root);

else return Father(root->right, val, root);

}

// Минимум в поддереве

TreeNode\* minValue(TreeNode\* node) {

TreeNode\* current = node;

while (current && current->left)

current = current->left;

return current;

}

// Удаление

TreeNode\* deleteNode(TreeNode\* root, int key) {

if (!root) return nullptr;

if (key < root->value) {

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

}

else if (key > root->value) {

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

}

else {

// Нет потомков или один

if (!root->left) {

TreeNode\* temp = root->right;

delete root;

return temp;

}

if (!root->right) {

TreeNode\* temp = root->left;

delete root;

return temp;

}

// Два потомка

TreeNode\* temp = minValue(root->right);

root->value = temp->value;

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

}

return root;

}

int main() {

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

TreeNode\* root = nullptr;

int values[] = { 50, 30, 70, 20, 40, 60, 80 };

for (int val : values)

root = insert(root, val);

cout << "Заданное дерево: ";

inOrder(root);

cout << endl;

int val;

cout << "Ищем узел со значением: ";

cin >> val;

TreeNode\* node = Addr(root, val);

if (node) {

TreeNode\* parent = Father(root, val);

if (parent) {

cout << "Отец данного узла - узел со значением: " << parent->value << endl;

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

cout << "Отец узла удален." << endl;

}

else {

cout << "Узел — корень, отца нет." << endl;

}

}

else {

cout << "Узел не найден!" << endl;

}

cout << "Дерево после удаления: ";

inOrder(root);

cout << endl;

return 0;

}

Результат работы программы представлен на рисунке 1.

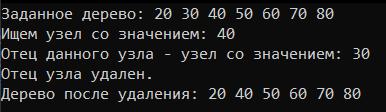
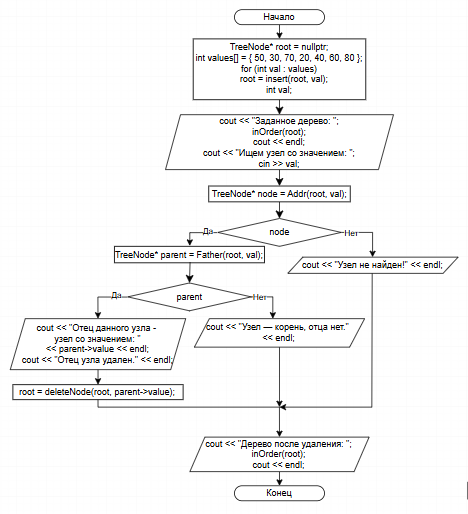


Рисунок 1 – Результат работы программы

Блок-схемы работы программы представлена на рисунках 2-6.



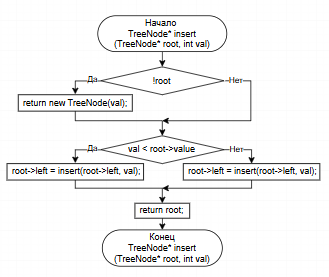
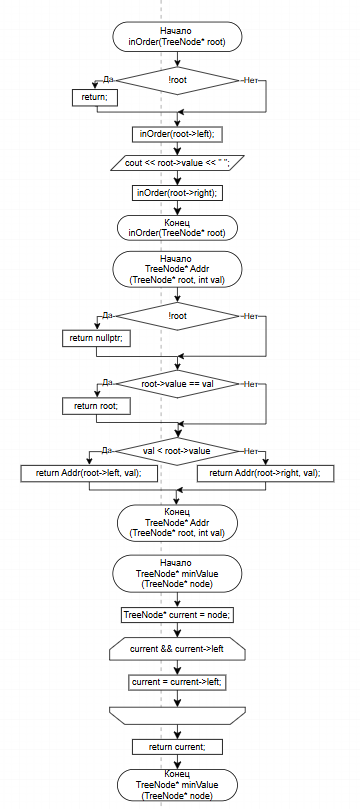
Рисунок 2

Рисунок 3

Рисунок 4

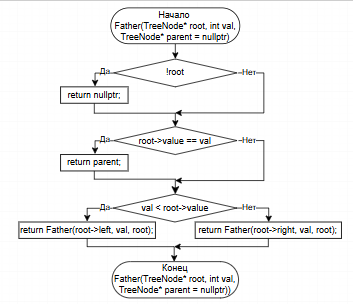
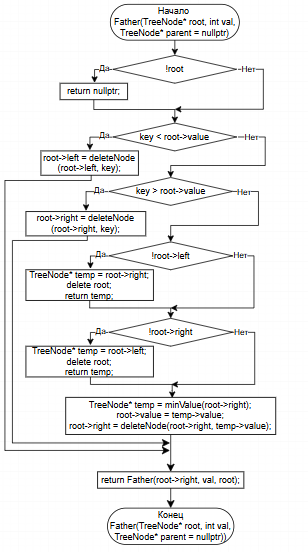


Рисунок 5

Рисунок 6

**Вывод:** В ходе выполнения работы была достигнута цель данной лабораторной работы: сформировать знания и умения по работе с подпрограммами, приобрести навыки написания программ с использованием бинарных деревьев.