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 14

Подготовил: Зайцев Е. А.

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:** На основе операции обхода сверху вниз реализовать операцию, определяющую, являются ли два бинарных дерева зеркально подобными (два бинарных дерева зеркально подобны, если они оба либо пусты, либо левое поддерево каждого дерева подобно правому дереву другого).

Код программы приведен ниже:

#include <iostream>

using namespace std;

struct tnode {

int field;

tnode\* left;

tnode\* right;

tnode(int value) : field(value), left(nullptr), right(nullptr) {}

};

// Обход дерева сверху вниз

void treeprint(tnode\* tree) {

if (tree != NULL) {

cout << tree->field << " ";

treeprint(tree->left);

treeprint(tree->right);

}

}

// Заполнение дерева (слева меньшее значение)

tnode\* insert(tnode\* root, int value) {

if (root == nullptr) {

return new tnode(value);

}

if (value < root->field) {

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

}

else {

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

}

return root;

}

// Заполнение дерева (слева большее значение)

tnode\* insert(tnode\* root, int value, int choise) {

if (root == nullptr) {

return new tnode(value);

}

if (value > root->field) { // Значения больше идут в левое поддерево

root->left = insert(root->left, value, choise);

}

else { // Значения меньше или равные идут в правое поддерево

root->right = insert(root->right, value, choise);

}

return root;

}

// Проверка на зеркальность деревьев

bool checker(tnode\* root, tnode\* root\_one) {

if (root == nullptr && root\_one == nullptr) {

return true; // Оба дерева пустые — зеркальны

}

if (root == nullptr || root\_one == nullptr) {

return false; // Одно из деревьев пустое — не зеркальны

}

return (root->field == root\_one->field &&

checker(root->left, root\_one->right) &&

checker(root->right, root\_one->left));

}

int main() {

setlocale(LC\_ALL, "Russian");

tnode\* root = nullptr;

tnode\* root\_one = nullptr;

// Создаем первое дерево (слева меньшее значение)

root = insert(root, 10);

root = insert(root, 5);

root = insert(root, 1);

root = insert(root, 3);

root = insert(root, 7);

// Создаем второе дерево (слева большее значение)

root\_one = insert(root\_one, 10, 1);

root\_one = insert(root\_one, 5, 1);

root\_one = insert(root\_one, 15, 1);

root\_one = insert(root\_one, 3, 1);

root\_one = insert(root\_one, 7, 1);

// Печатаем деревья

cout << "Первое дерево: ";

treeprint(root);

cout << endl;

cout << "Второе дерево: ";

treeprint(root\_one);

cout << endl;

// Проверяем зеркальность

if (checker(root, root\_one)) {

cout << "Заданные деревья зеркальны" << endl;

}

else {

cout << "Заданные деревья не зеркальны" << endl;

}

return 0;

}

Результат работающей программы представлен на рисунке 1:

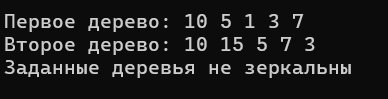


Рисунок 1 – Результат работы программы

Блок-схема работы программы представлена на рисунке 2-5.

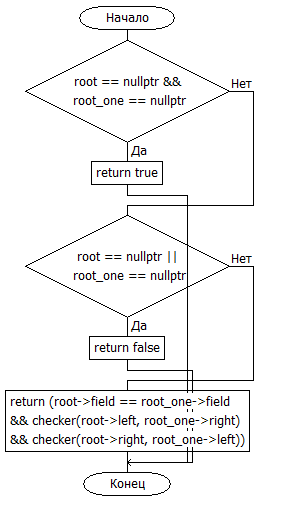
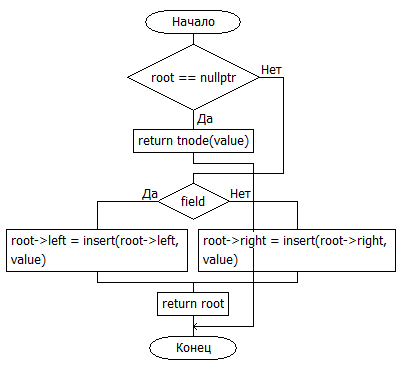
 

Рисунок 2 - checker() Рисунок 3 - insert()

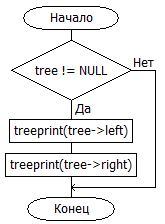


Рисунок 4 – treeprint()

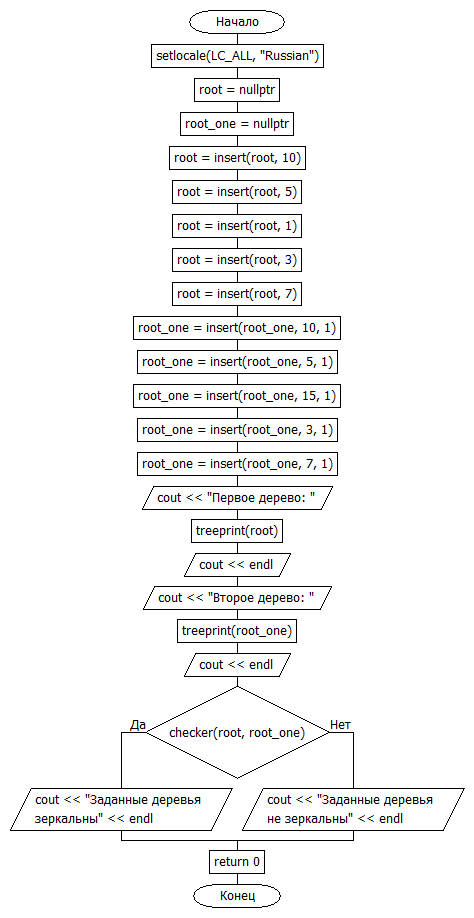
****

Рисунок 5 – main()

**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.