**Лабораторная работа №2**

**Бинарные деревья**

Написать функцию формирования бинарного дерева, состоящего из целых чисел. Для представления дерева использовать динамические структуры данных. Количество элементов дерева, а также его вид задаются случайным образом. Произвести вывод элементов дерева тремя видами обхода. Используя информацию о выведенном дереве изобразить структуру одного из построенных деревьев в отчете.

Выполнить следующее задание: написать рекурсивную функцию, которая строит копию дерева.

**Текст программы**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* left;

Node\* right;

};

void add(int x, Node\*& node) {

if (node == NULL) {

node = new Node;

node->data = x;

node->left = NULL;

node->right = NULL;

}

else {

if (x < node->data) {

add(x, node->left);

}

else {

add(x, node->right);

}

}

}

void printTree1(Node\*& node) { //прямой обход

if (node != NULL) {

cout << node->data << " ";

printTree1(node->left);

printTree1(node->right);

}

}

void printTree2(Node\*& node) { //симметричный обход

if (node != NULL) {

printTree2(node->left);

cout << node->data << " ";

printTree2(node->right);

}

}

void printTree3(Node\*& node) { //обратный обход

if (node != NULL) {

printTree3(node->left);

printTree3(node->right);

cout << node->data << " ";

}

}

Node\* copyTree(Node\*& original) {

if (original == NULL)

return NULL;

else {

Node\* node2 = new Node;

node2->data = original->data;

node2->left = copyTree(original->left);

node2->right = copyTree(original->right);

return node2;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(0));

Node\* root = NULL;

int elements = rand() % 11 + 10;

for (int i = 1; i <= elements; i++)

add(rand() % 100, root);

cout << "Оригинальное дерево" << endl << "Прямой обход: ";

printTree1(root);

cout << endl << "Симметричный обход: ";

printTree2(root);

cout << endl << "Обратный обход: ";

printTree3(root);

cout << endl << endl;

Node\* root2 = copyTree(root);

cout << "Копия" << endl << "Прямой обход: ";

printTree1(root2);

cout << endl << "Симметричный обход: ";

printTree2(root2);

cout << endl << "Обратный обход: ";

printTree3(root2);

cout << endl;

system("pause");

return 0;

}

**Описание программы**

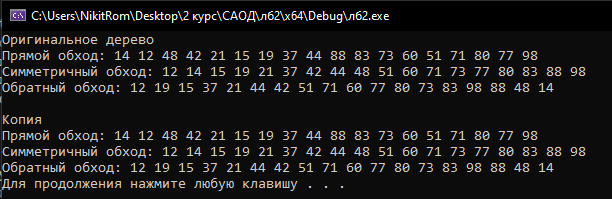
В начале программы задаются структуры и функции для работы с деревьями:

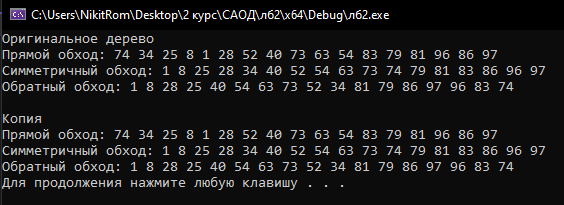
* **Node** – узел бинарного дерева
* **add** – добавление нового узла
* **printTree1** – прямой обход дерева
* **printTree2** – симметричный обход дерева
* **printTree3** – обратный обход дерева
* **copyTree** – копирование дерева

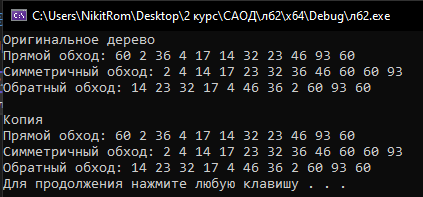
Содержание функции main:

1. Инициализируется структура **root**, ей присваивается значение **NULL**
2. Инициализируется переменная **elements** (количество элементов дерева), ей присваиваивается рандомное значение от 10 до 20
3. Дерево **root** заполняется рандомными числами
4. На экран выводятся результаты прямого обхода дерева **root**
5. На экран выводятся результаты симметричного обхода дерева **root**
6. На экран выводятся результаты обратного обхода дерева **root**
7. Инициализируется структура **root2**
8. Применяется функция **copyTree**, дерево **root** копируется в дерево **root2**
9. На экран выводятся результаты прямого обхода дерева **root2**
10. На экран выводятся результаты симметричного обхода дерева **root2**
11. На экран выводятся результаты обратного обхода дерева **root2**

**Результаты тестов**







**Пример дерева**

Для примера №3 построено дерево



**Вывод**

Мною была написана программа, формирующая бинарное дерево.

Программа выводит дерево с помощью 3 обходов: прямого, симметричного и обратного.

Также программа имеет возможность копировать деревья. Этот функционал реализован с помощью рекурсивной функции.