**Лабоpатоpная pабота №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

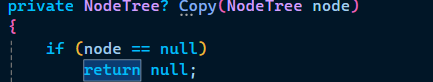
элементов дерева тремя видами обхода. Используя информацию о выведенном дереве

изобразить структуру одного из построенных деревьев в отчете. Выполнить по вариантам

следующие задания:

Вариант 17. Написать рекурсивную функцию, которая строит копию дерева

Алгоритм

1. Находим сначала крайний случай рекурсии. В данном случае, когда NodeTree является null. 
2. Создаем новый экземпляр класса NodeTree, в который будем записывать данные из node и сразу записываем значение из корня дерева. 
3. Записываем значение в правые и левые ветки result’а через рекурсию. 
4. Возвращаем result.

Код

Program.cs

using Lr2;

namespace Lr

{

internal class Program

{

static void Main()

{

BinaryTree tree = new();

var rnd = new Random();

var countElementInTree = rnd.Next(5, 10);

for (int i = 0; i < countElementInTree; i++)

tree.Add(rnd.Next(1, 100));

Console.WriteLine("Original tree:");

tree.PrintTreeS();

tree.PrintTreeR();

tree.PrintTreeC();

var copyTree = tree.Copy();

Console.WriteLine("Copy tree:");

copyTree.PrintTreeS();

copyTree.PrintTreeR();

copyTree.PrintTreeC();

}

}

}

BinaryTree.cs

using System.Threading.Channels;

using System.Xml.Linq;

namespace Lr2

{

public class BinaryTree

{

public class NodeTree(int data)

{

public int value = data;

public NodeTree? left;

public NodeTree? right;

}

public NodeTree? head;

public void Add(int data) => head = Add(head, data);

private NodeTree Add(NodeTree? tree, int data)

{

var tmp = tree;

if(tree == null)

{

return new NodeTree(data);

}

else

{

var rand = new Random();

if(rand.Next() % 2 == 0)

tmp.left = Add(tree.left, data);

else

tmp.right = Add(tree.right, data);

}

return tmp;

}

/// <summary>Прямой вывод дерева.</summary>

public void PrintTreeS()

{

PrintTreeS(head);

Console.WriteLine("\n");

}

private void PrintTreeS(NodeTree? tree)

{

if (tree == null)

return;

Console.Write($"{tree.value} ");

PrintTreeS(tree.left);

PrintTreeS(tree.right);

}

public void PrintTreeR()

{

PrintTreeR(head);

Console.WriteLine("\n");

}

private void PrintTreeR(NodeTree? tree)

{

if (tree == null)

return;

PrintTreeR (tree.left);

PrintTreeR(tree.right);

Console.Write($"{tree.value} ");

}

public void PrintTreeC()

{

PrintTreeC(head);

Console.WriteLine("\n");

}

private void PrintTreeC(NodeTree? tree)

{

if (tree == null)

return;

PrintTreeC(tree.left);

Console.Write($"{tree.value} ");

PrintTreeC(tree.right);

}

public BinaryTree Copy()

{

var copy = Copy(head);

var result = new BinaryTree();

result.head = copy;

return result;

}

private NodeTree? Copy(NodeTree? node)

{

if (node == null)

return null;

NodeTree result = new(node.value);

result.left = Copy(node.left);

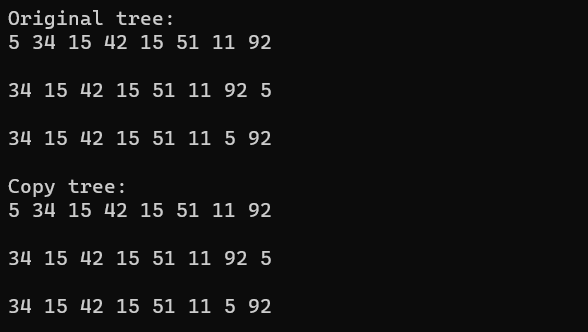
result.right = Copy(node.right);

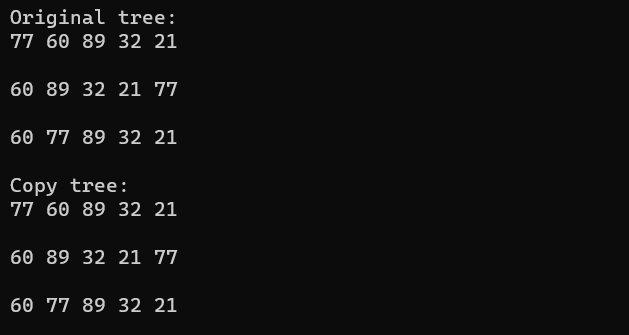
return result;

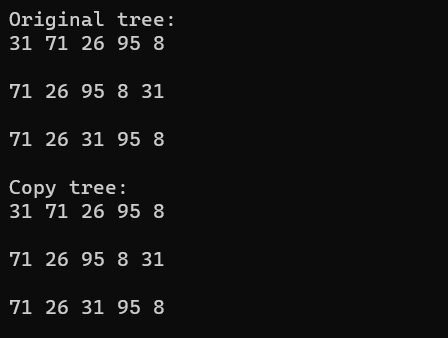
}

}

}Тест







Восстановленные деревья (порядок вывода:

прямой, обратный, симметричный).

