Составить программу для обхода (просмотра) всех вершин двоичного дерева. Двоичное дерево реализовать как статическую структуру в виде одномерного массива. Три наиболее часто используемых способа обхода (прямой, обратный и симметричный обход) запрограммировать как отдельные функции. В качестве входного параметра, функции принимают одномерный массив (двоичное дерево).

Примечание: двоичное дерево может иметь произвольное количество узлов. Значения узлов заполнить с клавиатуры.

using System;

using System.Collections;

using System.Collections.Generic;

namespace asd9

{

class Program

{

public class Node

{

public int value { get; set; }

public Node left { get; set; }

public Node right { get; set; }

public Node parent { get; set; }

Node root;

public void printPreorder(Node node)

{

if (node == null)

return;

Console.Write(node.parent + " ");

printPreorder(node.left);

printPreorder(node.right);

}

void printPreorder() { printPreorder(root); }

static void Main(string[] args)

{

int[] arr = new int[] { 1, 2, 4, 0, 0, 5, 0, 0, 3 };

Node[] tree = new Node[arr.Length];

Node parent = null;

for (int i = 0; i < arr.Length; i++)

{

tree[i] = new Node();

var node = tree[i];

node.value = arr[i];

node.parent = parent;

parent = node.value == 0 ? node.parent : node;

if (node.parent != null)

{

bool flag = true;

while (flag)

{

if (node.parent.left == null)

{

node.parent.left = node;

flag = false;

}

else if (node.parent.right == null)

{

node.parent.right = node;

flag = false;

}

else

{

node.parent = node.parent.parent;

}

}

}

}

tree.printPreorder();

// Console.WriteLine(tree.Display());

foreach (var item in tree)

{

//Console.WriteLine("{0} left {1}, right {2} parent {3}", item.value, item.left, item.right, item.parent);

}

}

}

}

}