```
// Ðåàëèçîâàòü äåðåâî ñ îïåðàöèÿìè ïîèñêà, óäàëåíèÿ, äîáàâëåíèÿ è âñå âèäû îáõîäîâ
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Tree
{
   public class Node
   {
      public int data;
      public Node right_child;
      public Node left_child;
   }
   class BinaryTree
   {
      public Node root;

      public BinaryTree()
      {
         root = null;
      }

      private Node Search( int data)
      {
         Node current = root;

         while (current.data != data)
         {
            if (current == null)
            {
               return null;
            }
            if (data < current.data)
            {
               current=current.left_child;
            }
            else if (data > current.data)
            {
              current = current.right_child;
            }
            }

            return current;
      }
      public void Insert(int data)
      {

         Node node = new Node();
         node.data = data;
         if (root == null)
```

```
    {
        root = node;

    }
    else
    {
        Node current = root;
        Node parent;
        while (true)
        {
            parent = current;
            if (data < current.data)
            {
                current = current.left_child;
                if (current == null)
                {
                    parent.left_child = node;
                    break;
                }
            }
            else
            {
                current = current.right_child;
                if (current == null)
                {
                    parent.right_child = node;
                    break;
                }
            }
        }
    }
}

private void PrintNode(Node root)
{
    Console.Write(root+ " ");
}

public void InOrder(Node root)
{
    if(root != null)
    {
        InOrder(root.left_child);
        PrintNode(root);
        InOrder(root.right_child);

    }

}

public void PreOrder(Node root)
{
```

```
            if (root != null)
            {
                PrintNode(root);
                PreOrder(root.left_child);
                PreOrder(root.right_child);

            }

        }

        public void PostOrder(Node root)
        {
            if (root != null)
            {
                PreOrder(root.left_child);
                PreOrder(root.right_child);
                PrintNode(root);

            }

        }

        public void BreadthFirstSearch(Node root)
        {
            Queue<Node> queue = new Queue<Node>();

            queue.Enqueue(root);

            while (queue != null)
            {
                Node node = queue.Dequeue();
                PrintNode(node);
                if (node.left_child != null)
                {
                    queue.Enqueue(node.left_child);
                }
                if (node.right_child != null)
                {
                    queue.Enqueue(node.right_child);
                }
            }
        }
    }
}
```