Binární stromy jsou v programování široce využívány pro ukládání a manipulaci s daty. V C# můžete implementovat binární strom pomocí tříd a rekurzivních funkcí. Zde jsou příklady vytvoření jednoduchého binárního stromu a několik operací s ním:

Definice třídy pro uzly binárního stromu:

```
public class Node
{
    public int Data { get; set; }
    public Node Left { get; set; }
    public Node Right { get; set; }

    public Node(int data)
    {
        Data = data;
        Left = null;
        Right = null;
    }
}
```

Definice třídy pro binární strom:

```
public class BinaryTree
{
    public Node Root { get; set; }
    public BinaryTree()
    {
        Root = null;
    }
    // Přidání uzlu do stromu
    public void Insert(int data)
    {
        Root = InsertRec(Root, data);
    }
    private Node InsertRec(Node root, int data)
    {
        if (root == null)
        {
            root = new Node(data);
            return root;
        }
        if (data < root.Data)
            root.Left = InsertRec(root.Left, data);
        else if (data > root.Data)
            root.Right = InsertRec(root.Right, data);
        return root;
    }
    // In-order procházení stromem (levý podstrom, kořen, pravý
podstrom)
    public void InOrderTraversal(Node root)
    {
        if (root != null)
```

```
        {
            InOrderTraversal(root.Left);
            Console.Write(root.Data + " ");
            InOrderTraversal(root.Right);
        }
    }
}
```

Použití:

```
class Program
{
    static void Main()
    {
        BinaryTree tree = new BinaryTree();

        // Přidání uzlů
        tree.Insert(50);
        tree.Insert(30);
        tree.Insert(20);
        tree.Insert(40);
        tree.Insert(70);
        tree.Insert(60);
        tree.Insert(80);

        // In-order procházení a výpis uzlů
        Console.WriteLine("In-order traversal:");
        tree.InOrderTraversal(tree.Root);
    }
}
```

Binární stromy mohou být využity v mnoha praktických situacích. Jedním z příkladů může být implementace slovníku nebo stromu pro vyhledávání a manipulaci s daty. Níže uvádím příklad projektu, který vytváří jednoduchý slovník pomocí binárního stromu v C#.

```
using System;

public class Node
{
    public int Key { get; set; }
    public string Value { get; set; }
    public Node Left { get; set; }
    public Node Right { get; set; }

    public Node(int key, string value)
    {
        Key = key;
        Value = value;
        Left = null;
        Right = null;
    }
}
```

```
public class BinarySearchTree
{
    public Node Root { get; set; }

    public BinarySearchTree()
    {
        Root = null;
    }

    public void Insert(int key, string value)
    {
        Root = InsertRec(Root, key, value);
    }

    private Node InsertRec(Node root, int key, string value)
    {
        if (root == null)
        {
            root = new Node(key, value);
            return root;
        }

        if (key < root.Key)
            root.Left = InsertRec(root.Left, key, value);
        else if (key > root.Key)
            root.Right = InsertRec(root.Right, key, value);

        return root;
    }

    public string Search(int key)
    {
        Node result = SearchRec(Root, key);
        return result != null ? result.Value : "Key not found";
    }

    private Node SearchRec(Node root, int key)
    {
        if (root == null || root.Key == key)
            return root;

        if (key < root.Key)
            return SearchRec(root.Left, key);

        return SearchRec(root.Right, key);
    }
}

class Program
```

```
{
    static void Main()
    {
        BinarySearchTree dictionary = new BinarySearchTree();

        // Přidání klíčů a hodnot do slovníku
        dictionary.Insert(50, "apple");
        dictionary.Insert(30, "banana");
        dictionary.Insert(70, "cherry");
        dictionary.Insert(20, "grape");
        dictionary.Insert(40, "lemon");
        dictionary.Insert(60, "orange");
        dictionary.Insert(80, "pear");

        // Vyhledání hodnoty pro daný klíč
        Console.WriteLine("Value for key 40: " +
dictionary.Search(40));
        Console.WriteLine("Value for key 90: " +
dictionary.Search(90));
    }
}
```

V tomto projektu je vytvořen jednoduchý slovník, kde klíče jsou celá čísla a hodnoty jsou řetězce. Projekt zahrnuje přidávání prvků do slovníku (metoda Insert) a vyhledávání hodnoty pro zadaný klíč (metoda Search).

Program vytváří jednoduchý slovník pomocí binárního vyhledávacího stromu. Zde je stručný popis fungování programu:

1. **Definice třídy `Node`:**
   - Tato třída představuje uzel binárního stromu.
   - Každý uzel má klíč (Key), hodnotu (Value), a odkazy na levého a pravého potomka (**`Left`** a **`Right`**).
2. **Definice třídy `BinarySearchTree`:**
   - Tato třída představuje samotný binární vyhledávací strom.
   - Obsahuje metody pro vložení uzlu (**`Insert`**) a pro vyhledání hodnoty podle klíče (**`Search`**).
3. **Hlavní metoda `Main`:**
   - Vytvoření instance **`BinarySearchTree`** nazvané **`dictionary`**.
   - Přidání několika klíčů a hodnot do stromu pomocí metody **`Insert`**.
   - Vytištění hodnoty pro určitý klíč pomocí metody **`Search`**.
4. **Vložení prvků:**
   - Při vložení prvku se provede porovnání klíče s aktuálním uzlem.
   - Pokud je klíč menší než klíč aktuálního uzlu, prvky jsou vkládány do levého podstromu; pokud je klíč větší, prvky jsou vkládány do pravého podstromu.
   - Tato operace je prováděna rekurzivně, dokud není nalezen vhodný uzel pro vložení nového prvku.
5. **Vyhledávání hodnoty:**
   - Při vyhledávání se klíč porovnává s klíči uzlů ve stromu.

- Pokud se klíč shoduje s klíčem aktuálního uzlu, je vrácena odpovídající hodnota.
- Pokud je klíč menší než klíč aktuálního uzlu, vyhledávání pokračuje v levém podstromu; pokud je klíč větší, vyhledávání pokračuje v pravém podstromu.
- Tato operace je také prováděna rekurzivně, dokud není nalezen hledaný klíč nebo není dosažen konec stromu.

6. **Výstup:**
   - Program vytiskne hodnotu pro klíč 40 (hrozno) a klíč 90 (není ve slovníku).

Tímto způsobem program demonstruje základní operace s binárním vyhledávacím stromem: vkládání prvků a vyhledávání hodnot podle klíče.

Zadání

Název projektu: Binární strom - Implementace slovníku

Cíle projektu:

1. Vytvořte třídu **Node**, která představuje uzel binárního stromu. Každý uzel by měl obsahovat klíč (**Key**), hodnotu (**Value**), a odkazy na levého a pravého potomka (**Left** a **Right**).
2. Implementujte třídu **BinarySearchTree**, která bude obsahovat metody pro vkládání uzlů do stromu a vyhledávání hodnoty podle klíče.
3. Vytvořte hlavní program, který:
   - Vytvoří instanci **BinarySearchTree**.
   - Přidá několik klíčů a hodnot do stromu.
   - Vytiskne hodnoty pro určité klíče pomocí metody pro vyhledávání.

Požadavky na implementaci:

1. Ujistěte se, že vložení uzlů do stromu probíhá správně a že strom zůstává vyvážený.
2. Implementujte metodu pro vyhledávání hodnoty podle klíče a zajistěte, aby vrátila správný výsledek pro existující a neexistující klíče.
3. Zajistěte, aby váš kód byl dobře zdokumentován pomocí komentářů ve zdrojovém kódu.
4. Implementujte metodu pro odstraňování uzlů ze stromu.
1. Vytvořte metodu pro in-order procházení stromu a výpis uzlů ve správném pořadí.
2. Umožněte uživateli zadávat vstupní data pro klíče a hodnoty, a nechte program dynamicky vytvářet binární strom.

Odevzdání: Odevzdejte kompletní zdrojový kód projektu v jazyce C# spolu s popisem ve formě dokumentace nebo komentářů ve zdrojovém kódu. Tato dokumentace by měla obsahovat popis implementace, rozhodnutí při návrhu a popis testování aplikace.