

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського» Факультет
інформатики та обчислювальної техніки Кафедра
інформатики та програмної інженерії

Звіт

З лабораторної роботи №6 з дисципліни

«Основи програмування 2.

Модульне програмування»

«Дерева»

Варіант 27

Виконав студент

ІП-11 Савенко Олексій Андрійович

(шифр, прізвище, ім'я, по батькові)

Перевірів

Вітковська І.І.

(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 6

Дерева

Мета роботи - вивчити особливості організації і обробки дерев.

Варіант 27

27. Побудувати дерево, елементами якого є цілі числа. Визначити кількість вузлових вершин даного дерева та надрукувати їх координати (номер рівня та номер гілки).

Постановка завдання

Створити клас бінарного Дерева(**Tree**), а також клас його Вершини(**Node**), прописати конструктори для ініціалізації даних вершини – цілим числом, та кореня дерева відповідно. У класі вершини створити атрибути його нащадків – лівого та правого відповідно, а також атрибут даних цілого числа, яке зберігається у відповідному вузлі. Реалізувати у класі дерева методи визначення загальної кількості вузлів, виведення вершин та виведення вершин з їх координатами(**номер рівня, номер гілки**), а також додавання нової вершини до вже існуючого дерева.

Програмний код на мові C#:

Node.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab60PSharp
{
    public class Node //Клас вершини
    {
        public Node(int value) // Конструктор вершини та онулення вершин нащадків
        {
            Value = value;
            Left = Right = null;
        }

        public int Value { get; } //Властивість значення вершини

        //Гілки вершини
        public Node? Left { get; set; }
        public Node? Right { get; set; }
    }
}
```

```

        public override string ToString() => $"[{Value}]"; //Переведення значення
        вершини у строку
    }

```

```

}

```

Tree.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab60PSharp
{
    public class Tree
    {
        public Tree(int value) // Утворення кореня дерева та задання йому значення за
        допомогою реалізованого конструктора
        {
            Root = new Node(value);
        }

        public Node Root { get; } //Корінь дерева

        public void Insert(int value) => InsertHelper(value, Root);

        private void InsertHelper(int value, Node root) //Додавання нової вершини у
        дерево згідно до правил упорядкованості у бінарному дереві
        {
            if (value < root.Value)
            {
                if (root.Left == null)
                    root.Left = new Node(value);
                else InsertHelper(value, root.Left);
            }
            else if (value > root.Value)
            {
                if (root.Right == null)
                    root.Right = new Node(value);
                else InsertHelper(value, root.Right);
            }
        }

        public string PreorderTraverse() => string.Join("\n",
        PreorderTraverseHelper(Root, new List<Node>()));
    }
}

```

```

        private List<Node> PreorderTraverseHelper(Node node, List<Node> nodes)
//Виведення дерева та його вершин без координат
        {
            nodes.Add(node);
            if (node.Left != null)
                PreorderTraverseHelper(node.Left, nodes);
            if (node.Right != null)
                PreorderTraverseHelper(node.Right, nodes);
            return nodes;
        }

        public int CountNodes() //Підрахунок кількості вершин дерева
        {
            int count = 0;
            CountNodesHelper(ref count, Root);
            return count;
        }

        private void CountNodesHelper(ref int count, Node? node) //Підрахунок
кількості за рахунок обходу вершин
        {
            if (node == null) return;
            count++;
            CountNodesHelper(ref count, node.Left);
            CountNodesHelper(ref count, node.Right);
        }

        public string PreorderTraverseWithCoordinates() //Виведення вершин з
координатами(рівень, гілка)
        {
            var nodes = PreorderTraverseHelper(Root, new List<Node>()); //Масив
вершин отриманий з методу звичайного виведення
            StringBuilder res = new StringBuilder();
            foreach (Node node in nodes)
            {
                int depth = 0, branch = 0;
                GetNodesCoordinate(ref depth, ref branch, node, Root);
                res.Append($"({node}, {depth}, {branch})\n");
            }

            return res.ToString();
        }

        private void GetNodesCoordinate(ref int depth, ref int branch, Node
nodeToFind, Node? root) //Отримання координат вершини
        {
            if (root == null || root.Value == nodeToFind.Value) return;
            depth++;
            if (nodeToFind.Value < root.Value)
            {
                GetNodesCoordinate(ref depth, ref branch, nodeToFind, root.Left);
            }
            else

```

```

        {
            branch++;
            GetNodesCoordinate(ref depth, ref branch, nodeToFind, root.Right);
        }
    }
}

```

Worker.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab60PSharp
{
    internal class Worker
    {
        public Tree GetTree() // Метод створення дерева
        {
            Tree tree = RootCreating();

            tree = AddNodes(tree);

            return tree;
        }

        private Tree RootCreating() // Отримання кореню дерева
        {
            Console.WriteLine("Enter value for root of tree - ");
            int valueRoot = Convert.ToInt32(Console.ReadLine());

            Tree tree = new Tree(valueRoot);
            return tree;
        }

        private Tree AddNodes(Tree tree) // Додавання вершин до дерева
        {
            int[] values = GetValuesForTreeNode();

            foreach(int value in values)
            {
                tree.Insert(value);
            }

            return tree;
        }
    }
}

```

```

private int[] GetValuesForTreeNodes() // Отримання значень вершин дерева
{
    Console.WriteLine("Please enter amount of nodes you want add to your tree - ");

    int amount = Convert.ToInt32(Console.ReadLine());

    Random random = new Random();

    int[] values = new int[amount];

    for (int i = 0; i < amount; i++)
    {
        values[i] = random.Next(-100,101);
    }
    return values;
}

}
}

```

Program.cs

```

namespace Lab6OPSharp
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Worker worker = new Worker();

            Tree tree = worker.GetTree();
            Console.WriteLine("-----");
            Console.WriteLine($"Output of tree nodes:\n{tree.PreorderTraverse()}\n");
            //виведення без координат

            Console.WriteLine("-----");

            Console.WriteLine($"Output of tree nodes with
coordinates\n(depth,branch):\n{tree.PreorderTraverseWithCoordinates()}\n"); //виведення
з координатами

            Console.WriteLine("-----");

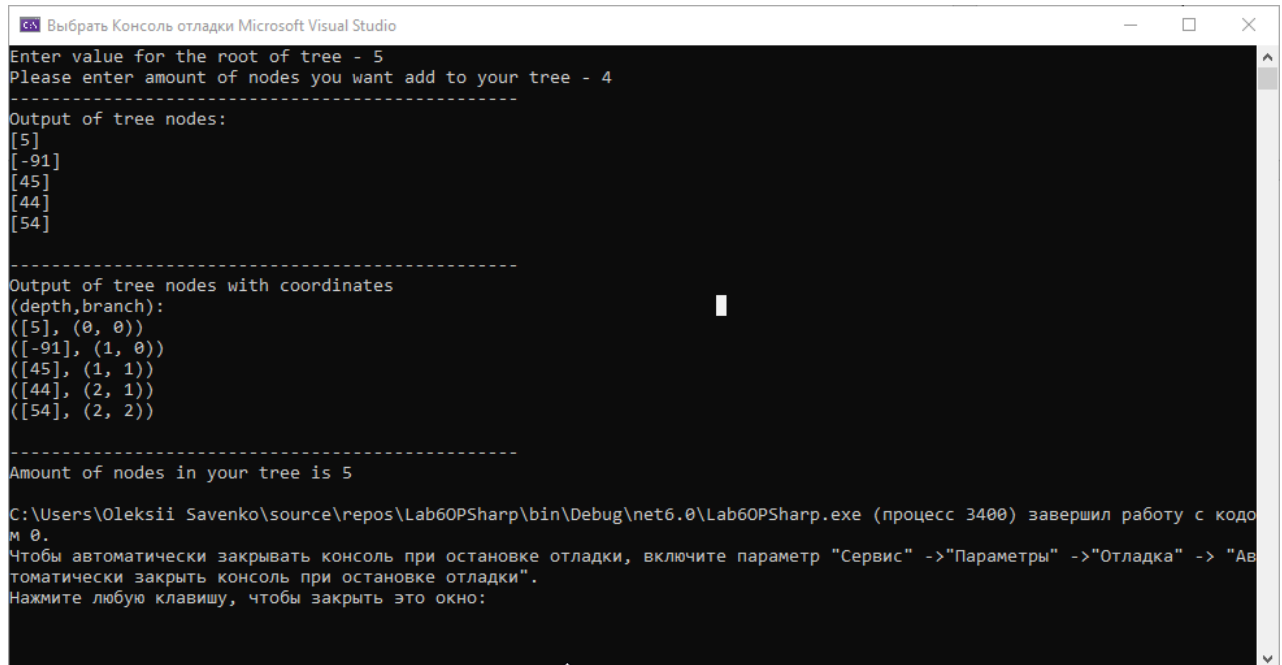
            Console.WriteLine($"Amount of nodes in your tree is
{tree.CountNodes()}");

        }
    }
}

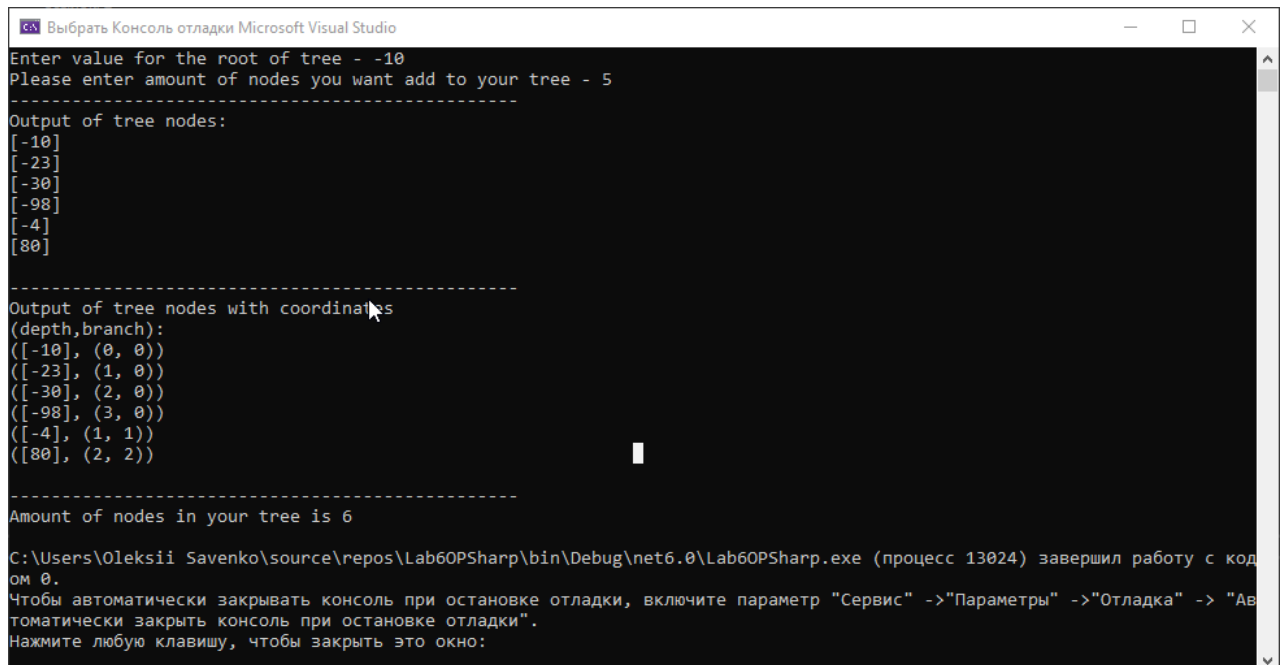
```

```
}  
}
```

Виконання програми на мові C#:



```
Выбрать Консоль отладки Microsoft Visual Studio  
Enter value for the root of tree - 5  
Please enter amount of nodes you want add to your tree - 4  
-----  
Output of tree nodes:  
[5]  
[-91]  
[45]  
[44]  
[54]  
-----  
Output of tree nodes with coordinates  
(depth,branch):  
([5], (0, 0))  
([-91], (1, 0))  
([45], (1, 1))  
([44], (2, 1))  
([54], (2, 2))  
-----  
Amount of nodes in your tree is 5  
  
C:\Users\Oleksii Savenko\source\repos\Lab60PSharp\bin\Debug\net6.0\Lab60PSharp.exe (процесс 3400) завершил работу с кодом 0.  
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".  
Нажмите любую клавишу, чтобы закрыть это окно:
```



```
Выбрать Консоль отладки Microsoft Visual Studio  
Enter value for the root of tree - -10  
Please enter amount of nodes you want add to your tree - 5  
-----  
Output of tree nodes:  
[-10]  
[-23]  
[-30]  
[-98]  
[-4]  
[80]  
-----  
Output of tree nodes with coordinates  
(depth,branch):  
([-10], (0, 0))  
([-23], (1, 0))  
([-30], (2, 0))  
([-98], (3, 0))  
([-4], (1, 1))  
([80], (2, 2))  
-----  
Amount of nodes in your tree is 6  
  
C:\Users\Oleksii Savenko\source\repos\Lab60PSharp\bin\Debug\net6.0\Lab60PSharp.exe (процесс 13024) завершил работу с кодом 0.  
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".  
Нажмите любую клавишу, чтобы закрыть это окно:
```

Висновок

Отже, я вивчив особливості організації і обробки дерев. Мною було розроблено алгоритм на основі поданого завдання, за його рахунок було написано програмне забезпечення на мові C#, при написанні було забезпечено відповідність нормам модульного підходу, були розроблені відповідні класи Дерева, Вершини, а також класу для утворення їхнього утворення. У програмному забезпеченні також присутні методи обробки даних дерева та його вершин, а саме – визначення загальної кількості вершин, виведення вершин за допомогою обходу з координатами та без них, і додавання нових вузлів до вже існуючого дерева. У результаті було розроблено ПЗ, яке виконує поставлене завдання та забезпечує інкапсуляцію даних у класах та основні принципи ООП. Програма була протестована, виявлені недоліки були виправлені задля вірного функціонування ПЗ.