

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 № 5 з дисципліни
«Основи програмування 2. Модульне програмування»

«Дерева»

Варіант 35

Виконав студент ІП-15, Шабанов Метін Шаміль огли
(шифр, прізвище, ім'я, по батькові)

Перевірів Вечерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 5

Дерева Варіант 35 (5)

Завдання:

5. Побудувати двійкове дерево, елементами якого є символи. Визначити, чи знаходиться у цьому дереві елемент, значення якого вводиться з клавіатури. Якщо елемент знайдений, то підрахувати число його входжень.

Виконання:

C#:

```
Program.cs | InputReceiver.cs | Tree.cs | Node.cs | TreeBuilder.cs | Printer.cs
1 namespace Lab6_Trees
2 {
3     class Program
4     {
5         static void Main(string[] args)
6         {
7             char[] symbols = InputReceiver.RecieveTreeData();
8             Tree symbolTree = TreeBuilder.BuildTree(symbols);
9             Printer.PrintTree(symbolTree);
10            char searched = InputReceiver.ReceiveChar();
11            Printer.PrintResults(searched, symbolTree);
12        }
13    }
14 }

Program.cs | InputReceiver.cs | Tree.cs | Node.cs | TreeBuilder.cs | Printer.cs
1 using System;
2 using System.Text.RegularExpressions;
3
4 namespace Lab6_Trees
5 {
6     public class InputReceiver
7     {
8         public static char[] RecieveTreeData()
9         {
10             Console.WriteLine("Enter the size of tree: ");
11             int size = _checkOnNumber(input: Console.ReadLine());
12             var treeData = new char[size];
13
14             for (int i = 0; i < size; i++)
15             {
16                 treeData[i] = ReceiveChar();
17             }
18
19             return treeData;
20         }
21
22         public static char ReceiveChar()
23         {
24             Console.WriteLine("Enter the symbol: ");
25             string input = Console.ReadLine();
26             while (input.Length != 1)
27             {
28                 Console.WriteLine("Wrong input, try again: ");
29                 input = Console.ReadLine();
30             }
31             return input[0];
32         }
33     }
34 }
```

```

30     }
31
32     return Convert.ToChar(input);
33 }
34
35 [1 usage]
36 private static int _checkOnNumber(string input)
37 {
38     while (!Regex.IsMatch(input, pattern:@"^\d+$"))
39     {
40         Console.WriteLine("Not a number! Try again: ");
41         input = Console.ReadLine();
42     }
43
44     return int.Parse(input);
45 }

```

C# Program.cs × C# InputReceiver.cs × C# Tree.cs × C# Node.cs × C# TreeBuilder.cs × C# Printer.cs ×

```

1  using System;
2  using System.Linq;
3
4  namespace Lab6_Trees
5  {
6      [6 usages] [1 exposing API]
7      public class Tree
8      {
9
10         [1 usage]
11         public Tree(char rootSym)
12         {
13             Node root = new Node(rootSym);
14             _root = root;
15         }
16
17         [2 usages]
18         public Node Root => _root;
19
20         [1 usage]
21         public void AddNode(char insertedSym)
22         {
23             Node newNode = new Node(insertedSym);
24             bool continueSearch = true;
25             Node current = _root;
26             while (continueSearch)
27             {
28                 Node tempParent = current;
29                 if (newNode.Data < current.Data)
30                 {
31                     current = current.Left;
32                 }
33                 else
34                 {
35                     current = current.Right;
36                 }
37             }
38             tempParent.Right = newNode;
39         }
40     }
41 }

```

C# Program.cs × C# InputReceiver.cs × C# Tree.cs × C# Node.cs × C# TreeBuilder.cs × C# Printer.cs ×

```
29         if (current == null)
30         {
31             tempParent.Left = newNode;
32             continueSearch = false;
33         }
34     }
35     else
36     {
37         current = current.Right;
38         if (current == null)
39         {
40             tempParent.Right = newNode;
41             continueSearch = false;
42         }
43     }
44 }
45 }
46
```

3 usages

```
47 public bool CheckElementPresence(Node current, char checkedElement)
48 {
49     bool presenceLeft = false;
50     bool presenceRight = false;
51     if (current.Data == checkedElement)
52         return true;
53     if (current.Left != null)
54         presenceLeft = CheckElementPresence(current.Left, checkedElement);
55     if (current.Right != null)
56         presenceRight = CheckElementPresence(current.Right, checkedElement);
57     return presenceLeft || presenceRight;
58 }
```

C# Program.cs × C# InputReceiver.cs × C# Tree.cs × C# Node.cs × C# TreeBuilder.cs × C# Printer.cs ×

```
60 public int CountElementEntries(Node current, char checkedElement)
61 {
62     if (current.Data == checkedElement)
63         return _checkElementChildren(current, checkedElement, entry: 1);
64
65     return _checkElementChildren(current, checkedElement, entry: 0);
66 }
67
68 2 usages
69 private int _checkElementChildren(Node current, char checkedElement, int entry)
70 {
71     if (current.Left != null && current.Right != null)
72         return entry + CountElementEntries(current.Left, checkedElement) + CountElementEntries(current.Right, checkedElement);
73     if (current.Left != null && current.Right == null)
74         return entry + CountElementEntries(current.Left, checkedElement);
75     if (current.Right != null && current.Left == null)
76         return entry + CountElementEntries(current.Right, checkedElement);
77     return entry;
78 }
79
80 5 usages
81 public string PreOrderTraversal(Node current)
82 {
83     string spaces = String.Concat(Enumerable.Repeat("\t", _checkLevelOfNode(currentParent: _root, searched: current, level: 0)));
84     if (current.Left == null && current.Right == null)
85         return $"{current.Data}\n";
86
87     if (current.Left == null && current.Right != null)
88         return $"{current.Data}-----{PreOrderTraversal(current.Right)}";
89     else if (current.Left != null && current.Right == null)
90         return $"{current.Data}\n{spaces}\\\\-----{PreOrderTraversal(current.Left)}";
91     else
```

```

90         return
91         $"{current.Data}-----{PreOrderTraversal(current.Right)}{spaces}\\-----{PreOrderTraversal(current.Left)}";
92     }
93
94     [3 usages]
95     private int _checkLevelOfNode(Node currentParent, Node searched, int level)
96     {
97         if (currentParent == null)
98             return 0;
99
100         if (currentParent == searched)
101             return level;
102
103         int downLevel = _checkLevelOfNode(currentParent.Left, searched, level: level + 1);
104         if (downLevel != 0)
105             return downLevel;
106
107         downLevel = _checkLevelOfNode(currentParent.Right, searched, level: level + 1);
108         return downLevel;
109     }
110
111     public override string ToString()
112     {
113         string tree = PreOrderTraversal(_root);
114         return tree;
115     }
116 }



```

```

1 namespace Lab6_Trees
2
3 [18 usages] [5 exposing APIs]
4 public class Node
5 {
6     private char _data;
7     private Node _left;
8     private Node _right;
9
10    [2 usages]
11    public Node(char data)
12    {
13        _data = data;
14    }
15
16    [8 usages]
17    public char Data
18    {
19        get => _data;
20        set => _data = value;
21    }
22
23    [15 usages]
24    public Node Left
25    {
26        get => _left;
27        set => _left = value;
28    }
29
30    [15 usages]
31    public Node Right
32    {
33        get => _right;
34        set => _right = value;
35    }
36 }

```

C# Program.cs × C# InputReceiver.cs × C# Tree.cs × C# Node.cs × C# TreeBuilder.cs × C# Printer.cs ×

```
1 namespace Labó_Trees
2 {
3      1 usage
4     public class TreeBuilder
5     {
6          1 usage
7         public static Tree BuildTree(char[] elements)
8         {
9             Tree symbolTree = new Tree(elements[0]);
10            for (int i = 1; i < elements.Length; i++)
11                symbolTree.AddNode(elements[i]);
12
13            return symbolTree;
14        }
15    }
16 }
```

C# Program.cs × C# InputReceiver.cs × C# Tree.cs × C# Node.cs × C# TreeBuilder.cs × C# Printer.cs ×

```
1 using System;
2
3 namespace Labó_Trees
4 {
5      2 usages
6     public class Printer
7     {
8          1 usage
9         public static void PrintTree(Tree tree)
10        {
11            Console.WriteLine("The tree is: \n");
12            Console.WriteLine(tree);
13        }
14
15         1 usage
16        public static void PrintResults(char checkedElement, Tree tree)
17        {
18            if (tree.CheckElementPresence(tree.Root, checkedElement))
19            {
20                int entries = tree.CountElementEntries(tree.Root, checkedElement);
21                Console.WriteLine($"The element is in the tree, number of entries: {entries}.");
22            }
23            else
24                Console.WriteLine("No such element in the tree.");
25        }
26    }
27 }
```

Тестування:

C#:

Enter the size of tree: 10

Enter the symbol: r

Enter the symbol: y

Enter the symbol: a

Enter the symbol: v

Enter the symbol: o

Enter the symbol: b

Enter the symbol: a

Enter the symbol: p

Enter the symbol: w

Enter the symbol: a

The tree is:

```
r-----y
      \-----v-----w
     \-----a-----o-----p
          \-----b
              \-----a-----a
```

Enter the symbol: a

The element is in the tree, number of entries: 3.