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 26

Виконав студент: ІП-15 Поліщук Валерій Олександрович

(шифр, прізвище, ім'я, по батькові)

Перевірила: Вечерковська Анастасія Сергіївна

(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота № 6

Дерева

Варіант 26

Мета — вивчити особливості організації і обробки дерев.

Постановка задачі

26. Побудувати двійкове дерево пошуку, в вершинах якого знаходяться слова з текстового файлу. Визначити кількість вершин дерева, що містять слова, які починаються на зазначену букву.

Розв’язання

C#

Program.cs

```
using System.Text;
using System.Threading.Tasks;

namespace Last_lab.classes
{
    internal class Program
    {
        static void Main(string[] args)
        {
            BinaryTree binaryTree = new BinaryTree();
            binaryTree.ReadFromFile();
            binaryTree.TraverseInOrder();

            #region get and parse letter
            Console.WriteLine();
            Console.WriteLine();
            Console.WriteLine("Enter a letter for search : ");

            char letter;

            while (!char.TryParse(Console.ReadLine(), out letter) || char.IsDigit(letter))
            {
                Console.WriteLine();
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("Incorrect format! Try again ");
                Console.WriteLine();
                Console.ForegroundColor = ConsoleColor.White;
                Console.WriteLine("Enter a letter for search : ");
            }
            Console.WriteLine();
            #endregion

            int count = binaryTree.FindCount(letter);

            #region output result
            Console.WriteLine();
            Console.WriteLine();
            Console.WriteLine("Total count : {0}", count);
            #endregion
        }
    }
}
```

BinaryTree.cs

```
7   namespace Last_lab.classes
8   {
9       public class BinaryTree
10      {
11
12          private class BinaryTreeNode
13          {
14              public BinaryTreeNode(string value)
15              {
16                  Value = value;
17              }
18
19              public BinaryTreeNode? Left { get; set; }
20
21              public BinaryTreeNode? Right { get; set; }
22
23              public string Value { get; private set; }
24
25          }
26
27
28          private BinaryTreeNode? _head;
29
30          private int _count;
31
32          public int Count
33          {
34              get
35              {
36                  return _count;
37              }
38          }
39
40          public void Clear()
41          {
42              _head = null;
43              _count = 0;
44          }
```

```
46          public void ReadFromFile(string path = "input.txt")
47          {
48              List<string> words;
49              using (StreamReader reader = new StreamReader(path, System.Text.Encoding.Default))
50              {
51                  string text = reader.ReadToEnd();
52                  words = text.Split(new char[] { ' ', '\r', '\n', ',', '.', '!', '?' }, StringSplitOptions.RemoveEmptyEntries).ToList();
53              }
54
55              foreach (var word in words)
56              {
57                  Add(word);
58              }
59
60          }
61
62          public int FindCount(char letter)
63          {
64              BinaryTreeNode? current = _head;
65
66
67              while (current != null)
68              {
69                  int result = current.Value[0].CompareTo(letter);
70
71                  if (result > 0)
72                  {
73
74                      current = current.Left;
75                  }
76                  else if (result < 0)
77                  {
78
79                      current = current.Right;
80                  }
81                  else
82                  {
83                      break;
84                  }
85              }
86
87              if (current == null)
88              {
89                  return 0;
90              }
91              else
92              {
93                  int count = 0;
94                  TraverseInOrderForSearch(current, ref count, letter);
95                  return count;
96              }
97
98
99          }
```

```
101         public void Add(string value)
102         {
103             if (_head == null)
104             {
105                 _head = new BinaryTreeNode(value);
106             }
107 
108             else
109             {
110                 AddTo(_head, value);
111             }
112 
113             _count++;
114         }
115 
116         private void AddTo(BinaryTreeNode node, string value)
117         {
118 
119             if (value.CompareTo(node.Value) < 0)
120             {
121 
122                 if (node.Left == null)
123                 {
124                     node.Left = new BinaryTreeNode(value);
125                 }
126                 else
127                 {
128                     AddTo(node.Left, value);
129                 }
130             }
131             else
132             {
133 
134                 if (value.CompareTo(node.Value) == 0)
135                 {
136 
137                 }
138                 else
139                 {
140 
141                     if (node.Right == null)
142                     {
143                         node.Right = new BinaryTreeNode(value);
144                     }
145                     else
146                     {
147                         AddTo(node.Right, value);
148                     }
149                 }
150             }
151         }
152 
```

```
153         private void TraverseInOrderForSearch(BinaryTreeNode parent, ref int count, char letter)
154         {
155             if (parent != null)
156             {
157                 TraverseInOrderForSearch(parent.Left, ref count, letter);
158                 if (parent.Value[0] == letter)
159                 {
160                     Console.Write("{0})" + parent.Value + "      ", ++count);
161                 }
162 
163                 TraverseInOrderForSearch(parent.Right, ref count, letter);
164             }
165         }
166 
167         private void TraverseInOrder(BinaryTreeNode parent)
168         {
169             if (parent != null)
170             {
171                 TraverseInOrder(parent.Left);
172                 Console.Write(parent.Value + " ");
173                 TraverseInOrder(parent.Right);
174             }
175         }
176 
177         public void TraverseInOrder()
178         {
179             if (_head != null)
180             {
181                 TraverseInOrder(_head);
182             }
183             else
184             {
185                 throw new InvalidOperationException("Tree is empty");
186             }
187 
188         }
189 
190     }
191 }
```

Тестування програми

C#

```
Microsoft Visual Studio Debug Console
adorable arrange betray boat bury contribute crib damp detect error event eyes fan frame fuzzy gain gash grade greasy guitar heal indent
inspire invincible last milk mom occupy overconfident plain price punish renew renounce retain rotten sanctify smell solicit song squeeze
 statement step support swim tasteful tongue trains tree waste worried yarn

Enter a letter for search :
f

1)fan      2)frame      3)fuzzy

Total count : 3
```

Висновок

Я вивчив особливості організації і обробки дерев.