ЗВІТ
Основи програмування
Лабораторна робота 7
ПОБУДОВА ТА ВИКОРИСТАННЯ СТРУКТУР ДАНИХ

Виконав: Радченко Микола Сергійович ІП-44

Завдання:

9	double	Двоспря-мований	Включення до початку	1.Знайти перше входження елементу меншого за середнє значення. 2.Знайти суму елементів, які розташовані після максимального елементу. 3.Отримати новий список зі значень елементів більших за задане значення. 4.Видалити елементи, які розташовані до максимального елементу.

Код:

BiDirectionalList.cs

```
1   using System;
2   using System.Collections;
3   using System.Collections.Generic;
4
5   namespace List
6   {
        6 references | Windsurf: Refactor | Explain
7       class BiDirectionalList : IEnumerable<double>
8       {
            8 references
9           private Node? head;
            4 references
10          private Node? tail;
11
12
13
            4 references | Windsurf: Refactor | Explain | Generate Documentation
14          public void AddToBeginning(double data)
15          {
16              Node newNode = new Node(data);
17              if (head != null)
18              {
19                  newNode.Next = head;
20                  head.Prev = newNode;
21              }
22              head = newNode;
23              if (tail == null)
24              {
25                  tail = newNode;
26              }
27          }
28
            2 references | Windsurf: Refactor | Explain | Generate Documentation
29          public double FindAvarage(){
30              if(this == null) throw new EmptyListException();
31              int count = 0;
32              double sum = 0;
33              foreach (var item in this){
34                  count++;
35                  sum += item;
36              }
```

```
37          return sum/count;
38      }
39
        1 reference | Windsurf: Refactor | Explain | Generate Documentation
40      public int FindFirstEntryOfElementBelowAvarage(){
41          if(this == null) throw new EmptyListException();
42          var current = head;
43          if(current == null) throw new EmptyListException();
44          int index = 0;
45          while(current != null){
46              if(current.Data < FindAvarage()){
47                  return index;
48              }
49              current = current.Next;
50              index++;
51          }
52          return index;
53      }
54
        4 references | Windsurf: Refactor | Explain | Generate Documentation
55      public double FindMaxElement(){
56          if(this == null) throw new EmptyListException();
57          var current = head;
58          if(current == null) throw new EmptyListException();
59          double max = 0;
60          while(current != null){
61              if(current.Data > max){
62                  max = current.Data;
63              }
64              current = current.Next;
65          }
66          return max;
67      }
68
        1 reference | Windsurf: Refactor | Explain | Generate Documentation
69      public double FindSumAfterMaxElement(){
70          if(this == null) throw new EmptyListException();
71          var current = tail;
72          double result = 0;
73          while(current != null){
```

```
74              if(current.Data == FindMaxElement()){
75                  return result;
76              }
77              result += current.Data;
78              current = current.Prev;
79          }
80          return result;
81      }
82
83
        1 reference | Windsurf: Refactor | Explain | Generate Documentation
84      public BiDirectionalList NewListGreaterThanNumber(double number){
85          if(this == null) throw new EmptyListException();
86          var newBiDirectionalList = new BiDirectionalList();
87          var current = head;
88          while(current != null){
89              if(current.Data > number){
90                  newBiDirectionalList.AddToBeginning(current.Data);
91              }
92              current = current.Next;
93          }
94
95          return newBiDirectionalList;
96      }
97
        1 reference | Windsurf: Refactor | Explain | Generate Documentation
98      public BiDirectionalList NewListWithoutElementsBeforeMax(){
99          if(this == null) throw new EmptyListException();
100         var newBiDirectionalList = new BiDirectionalList();
101         var current = tail;
102         while(current.Data != FindMaxElement()){
103             newBiDirectionalList.AddToBeginning(current.Data);
104             current = current.Prev;
105         }
106         newBiDirectionalList.AddToBeginning(FindMaxElement());
107         return newBiDirectionalList;
108     }
109
```

```
110         public IEnumerator<double> GetEnumerator()
111         {
112             var current = head;
113             while (current != null)
114             {
115                 yield return current.Data;
116                 current = current.Next;
117             }
118         }
119
            0 references
120         IEnumerator IEnumerable.GetEnumerator() => GetEnumerator();
121     }
122
        0 references | Windsurf: Refactor | Explain
123     class Program
124     {
            0 references | Windsurf: Refactor | Explain | Generate Documentation
125         static void Main(string[] args)
126         {
127             BiDirectionalList list = new BiDirectionalList();
128
129             Random rng = new Random();
130
131             for (int i = 0; i < 5; i++)
132             {
133                 list.AddToBeginning(Math.Round(rng.NextDouble() * 10, 2));
134             }
135
136             int index = list.FindFirstEntryOfElementBelowAvarage();
137
138             foreach (double item in list)
139             {
140                 if(index == 0) {
141                     Console.WriteLine(item + " ← first element below avarage =)");
142                 }
143                 else{
144                     Console.WriteLine(item);
145                 }
```

```
            index--;
        }

        Console.WriteLine("Avarage: " + Math.Round(list.FindAvarage(), 2));
        Console.WriteLine("Max element: " + Math.Round(list.FindMaxElement(), 2));
        Console.WriteLine("Sum after max element: " + Math.Round(list.FindSumAfterMaxElement(), 2));



        Console.WriteLine("Enter the number to create a new list from the first one but without " +
        "elements below it: ");
        double yourNumber = Convert.ToDouble(Console.ReadLine());

        var newList = list.NewListGreaterThanNumber(yourNumber);
        Console.WriteLine("New list with elements greater than " + yourNumber + ": ");

        foreach (double item in newList)
        {
            Console.WriteLine(item);
        }

        var newList2 = list.NewListWithoutElementsBeforeMax();
        Console.WriteLine("New list without elements before max element: ");

        foreach (double item in newList2)
        {
            Console.WriteLine(item);
        }
    }
}
```

Node.cs

```
using System;

namespace List;

9 references | Windsurf: Refactor | Explain
public class Node
{
    3 references
    private Node? next;
    3 references
    private Node? prev;
    3 references
    private double data;

    10 references
    public double Data {
        get => data;
        set => data = value;
    }
    5 references
    public Node? Next {
        get => next;
        set => next = value;
    }
    3 references
    public Node? Prev {
        get => prev;
        set => prev = value;
    }

    1 reference | Windsurf: Refactor | Explain | Generate Documentation
    public Node(double data)
    {
        this.data = data;
        next = null;
        prev = null;
    }
}
```

EmptyListException.cs

```
using System;

namespace List;

11 references | Windsurf: Refactor | Explain
public class EmptyListException : Exception
{
    8 references
    public EmptyListException() { }
    0 references
    public EmptyListException(string message) : base(message) { }
    0 references
    public EmptyListException(string message, Exception inner) : base(message, inner) { }
}
```