ЗВІТ
Основи програмування
Лабораторна робота 7
ПОБУДОВА ТА ВИКОРИСТАННЯ СТРУКТУР ДАНИХ

Виконав: Комін Іван Андрійович ІП-44

Завдання:

7	float	Односпрямований	Включення після другого елементу	1.Знайти перше значення більше за задане в 2 рази. 2.Знайти кількість елементів, більших за 3.14. 3.Отримати новий список зі значень елементів більших за заданє значення. 4.Видалити елементи, які більші за середнє значення.

Код: List.cs

```
using System.Collections;
class Node
{
    public double data;
    public Node next;

    public Node(double data)
    {
        this.data = data;
        next = null;
    }
}
class LinkedList : IEnumerable<double>
{
    public Node head;
    public LinkedList()
    {
        head = null;
    }
    public void AddFirstNode(double data)
    {
        Node newNode = new Node(data);
        newNode.next = head;
        head = newNode;
    }
    public void AddLastNode(double data)
    {
        Node newNode = new Node(data);
```

```
        if (head == null)
        {
            head = newNode;
        }
        else
        {
            Node temp = head;
            while (temp.next != null)
            {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }
    public void PrintList()
    {
        Node temp = head;
        while (temp != null)
        {
            Console.Write(temp.data + " -> ");
            temp = temp.next;
        }
        Console.WriteLine("null");
    }
    public void InsertAfterSecond(double data)
    {
        int index = 0;
        Node temp = head;
        while(temp != null && index < 1)
        {
            temp = temp.next;
            index++;
        }
        if (temp != null)
        {
            Node newNode = new Node(data);
            newNode.next = temp.next;
            temp.next = newNode;
        }
    }
    public double? FindTwiceGreater(double value)
    {
```

```
        Node temp = head;
        while (temp != null)
        {
            if (temp.data == value * 2)
            {
                return temp.data;
            }
            temp = temp.next;
        }
        return null;
    }
    public int FindGreaterThanPi()
    {
        int amount = 0;
        Node temp = head;
        while (temp != null)
        {
            if (temp.data > 3.14)
            {
                amount++;
            }
            temp = temp.next;
        }
        return amount;
    }
    public LinkedList NewList(double value)
    {
        var newList = new LinkedList();
        Node temp = head;
        while (temp != null)
        {
            if (temp.data > value)
            {
                newList.AddLastNode(temp.data);
            }
            temp = temp.next;
        }
        return newList;
    }
    public void DeleteNode(double value)
    {
        if (head == null) return;
```

```
        if (head.data == value)
        {
            head = head.next;
            return;
        }
        Node temp = head;
        while (temp.next != null)
        {
            if (temp.next.data == value)
            {
                temp.next = temp.next.next;
                return;
            }
            temp = temp.next;
        }
    }
    public void DeleteGreaterThanAverage()
    {
        if (head == null) return;
        double sum = 0;
        int count = 0;
        Node temp = head;
        while (temp != null)
        {
            sum += temp.data;
            count++;
            temp = temp.next;
        }
        double average = sum / count;
        Console.WriteLine("The average is: " + average);
        temp = head;
        while (temp != null)
        {
            if (temp.data > average)
            {
                DeleteNode(temp.data);
            }
            temp = temp.next;
        }
    }

    public IEnumerator<double> GetEnumerator()
```

```
    {
        Node temp = head;
        while (temp != null)
        {
            yield return temp.data;
            temp = temp.next;
        }
    }
    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }
    public double this[int index]
    {
        get
        {
            if (index < 0) throw new
ArgumentOutOfRangeException("Index is out of range.");
            Node current = head;
            int i = 0;
            while (current != null)
            {
                if (i == index)
                    return current.data;
                current = current.next;
                i++;
            }
            throw new ArgumentOutOfRangeException("Index is out of
range.");
        }
    }

    public void DeleteElementByIndex(int index)
    {
        if (index < 0) throw new IndexOutOfRangeException("Index is
out of range.");
        if (index == 0)
        {
            head = head.next;
            return;
        }
        Node temp = head;
```

```
        int i = 0;
        while (temp != null && i < index - 1)
        {
            temp = temp.next;
            i++;
        }
        if (temp == null) throw new IndexOutOfRangeException("Index
is out of range.");

        temp.next = temp.next.next;
    }
}
```

Program.cs:

```
class Program
{
    static void Main(string[] args)
    {
        var list = new LinkedList();
        list.AddFirstNode(1.0);
        list.AddLastNode(2.0);
        list.AddLastNode(3.0);
        list.InsertAfterSecond(4.0);
        list.InsertAfterSecond(5.0);

        Console.WriteLine("The initial list is: ");
        list.PrintList();

        double value = 1;
        double? twiceGreater = list.FindTwiceGreater(value);
        if (twiceGreater != null)
        {
            Console.WriteLine($"The first number that is twice
greater than {value} is: " + twiceGreater);
        }
        else
        {
            Console.WriteLine($"There is no number that is twice
greater than {value} in the list.");
```

```
        }
        Console.WriteLine($"The amount of elements larger than 3.14
is {list.FindGreaterThanPi()}");

        var newList = list.NewList(2.0);
        Console.WriteLine("The new list is: ");
        newList.PrintList();

        list.DeleteGreaterThanAverage();
        Console.WriteLine("The list after deleting elements greater
than the average is: ");
        list.PrintList();

        list.InsertAfterSecond(6.0);
        list.InsertAfterSecond(7.0);
        list.InsertAfterSecond(8.0);
        list.InsertAfterSecond(9.0);
        list.InsertAfterSecond(10.0);

        Console.WriteLine("Iterating using foreach: ");
        foreach(var item in list) Console.Write(item + " -> ");
        Console.WriteLine("null");
        Console.WriteLine();

        Console.WriteLine("The element at index 2 is: " + list[2]);

        list.DeleteElementByIndex(2);
        Console.WriteLine("The list after deleting the element at
index 2 is: ");
        list.PrintList();

    }
```

Висновок: У ході виконання роботи було реалізовано однозв’язний список мовою програмування C#. Реалізація включала базову функціональність додавання елементів у початок, кінець та після другого елемента списку, а також операції пошуку, створення нового списку за умовою, обчислення середнього значення та видалення елементів, більших за середнє. Додатково було реалізовано можливість доступу до елементів за індексом, видалення за індексом і підтримку конструкції foreach, що забезпечує зручний перебір елементів списку.