ЗВІТ

Основи Програмування

Лабораторна робота 7

ПОБУДОВА ТА ВИКОРИСТАННЯ СТРУКТУР ДАНИХ

Виконав: Максименко Роман ІП-44

Завдання

8	int	Двоспря-мований	Включення в кінець списку	1.Знайти перше входження заданого елементу. 2. Знайти суму елементів, які розташовані на не парних позиціях списку (нумерація починається з голови списку). 3.Отримати новий список зі значень елементів значення яких більші за задане значення. 4.Видалити елементи, які більші за середнє значення.

Код:

Node.cs

```
namespace DoublyLinkedListLibrary;

public class Node
{
    private Node? previous;
    private int data;
    private Node? next;

    public Node? Previous
    {
        get => previous;
        set => previous = value;
    }

    public int Data
    {
        get => data;
        set => data = value;
    }

    public Node? Next
    {
        get => next;
        set => next = value;
    }

    public Node(int value)
    {
        data = value;
        previous = null;
        next = null;
    }
}
```

DoublyLinkedList.cs

```
using System.Collections;
using DoublyLinkedListLibrary.Exceptions;

namespace DoublyLinkedListLibrary;

public class DoublyLinkedList : IEnumerable<int>
{
    private Node? head;
    private Node? tail;

    public void InsertAtTheEnd(int value)
    {
        var node = new Node(value);

        if (tail != null)
        {
            tail.Next = node;
            node.Previous = tail;
        }

        tail = node;

        if (head == null)
        {
            head = node;
        }
    }

    public int? FindFirstElementEntry(int value)
    {
        if (head == null) throw new EmptyListException("List is empty");
        var curr = head;
        int index = 1;
        while (curr != null)
        {
            if (curr.Data == value)
            {
                return index;
            }

            curr = curr.Next;
            index++;
        }

        return null;
    }

    public int FindSumOfOddElements()
    {
        if (head == null) throw new EmptyListException("List is empty");
        var curr = head;
        int index = 0;
        int sum = 0;
        while (curr != null)
        {
            if (index % 2 == 0)
            {
                sum += curr.Data;
            }

            curr = curr.Next;
            index++;
        }
```

```
        return sum;
    }

    public DoublyLinkedList GetListOfElementsGreaterThan(int value)
    {
        if (head == null) throw new EmptyListException("List is empty");
        var list = new DoublyLinkedList();
        var curr = head;
        while (curr != null)
        {
            if (curr.Data > value)
            {
                list.InsertAtTheEnd(curr.Data);
            }

            curr = curr.Next;
        }

        return list;
    }

    public void DeleteElementsGreaterThanAverage()
    {
        if (head == null) throw new EmptyListException("List is empty");
        var curr = head;
        int sum = 0;
        int count = 0;
        while (curr != null)
        {
            sum += curr.Data;
            count++;
            curr = curr.Next;
        }

        decimal average = (decimal)sum / count;
        curr = head;
        while (curr != null)
        {
            if (curr.Data > average)
            {
                if(curr.Previous == null)
                {
                    head = curr.Next;
                    curr.Next!.Previous = null;
                }
                else if (curr.Next == null)
                {
                    tail = curr.Previous;
                    curr.Previous!.Next = null;
                }
                else
                {
                    curr.Previous!.Next = curr.Next;
                    curr.Next!.Previous = curr.Previous;
                }
            }
            curr = curr.Next;
        }
    }

    public IEnumerator<int> GetEnumerator()
    {
        var current = head;
        while (current != null)
        {
            yield return current.Data;
```

```
            current = current.Next;
        }
    }

    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }
}
```

EmptyListException.cs

```
namespace DoublyLinkedListLibrary.Exceptions;

class EmptyListException : Exception
{
    public EmptyListException()
    {
    }

    public EmptyListException(string message) : base(message)
    {
    }

    public EmptyListException(string message, Exception inner) : base(message,
inner)
    {
    }
}
```