

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 № 7 з дисципліни
«Основи програмування. Частина 2.
Методології програмування»

«Побудова та використання структур даних»

Варіант 7

Виконав студент ІІІ-43
Гребняк Вадим Крістіанович

Перевірила
Вітковська Ірина Іванівна

Київ 2024

ПОБУДОВА ТА ВИКОРИСТАННЯ СТРУКТУР ДАНИХ

Мета: дослідити типи лінійних та нелінійних структур даних, навчитись користуватись бібліотечними реалізаціями структур даних та будувати власні.

Опис завдання: Створити зв'язаний список зі включенням після другого елемента. Додати основні функції списку та можливість отримати з нього елемент. Використовуючи створений список виконати наступні завдання:

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 задане.
4. Видалити елементи, які більші за середнє значення.

Вихідний код програми:

Файл LinkedListMod.cs

```
using System;
using System.Collections;
using System.Collections.Generic;

namespace linked_list
{
    public class LinkedListMod : IEnumerable<float>
    {
        private class Node
        {
            public float data;
            public Node next;

            public Node(float data, Node next)
            {
                this.data = data;
                this.next = next;
            }
        }

        private int _insertionPosition = 2;
        private int _elementsCount = 0;
        private Node _head = null;
```

```

private float FindElementByIndex(int index)
{
    if (_elementsCount == 0 || index < 0 || index >= _elementsCount)
        throw new IndexOutOfRangeException();

    Node currentNode = _head;
    int currentIndex = 0;

    do
    {
        if (currentIndex == index)
            return currentNode.data;

        currentNode = currentNode.next;
        currentIndex++;
    } while (currentNode != null);

    return -1f;
}

public float this[int index] => FindElementByIndex(index);

public int Count() { return _elementsCount; }

public void Add(float data)
{
    if (_elementsCount == 0)
    {
        Node headNode = new Node(data, null);
        _head = headNode;
    }
    else if (_elementsCount <= _insertionPosition)
    {
        Node currentNode = _head;

        while (currentNode.next != null)
        {

```

```

        currentNode = currentNode.next;
    }

    currentNode.next = new Node(data, null);
}
else
{
    Node currentNode = _head;

    for (int i = 1; i < _insertionPosition; i++)
    {
        currentNode = currentNode.next;
    }

    Node tempNode = currentNode.next;

    currentNode.next = new Node(data, tempNode);
}

_elementsCount++;
}

```

```

public void Remove(float data)
{
    if (_elementsCount == 0)
        throw new InvalidOperationException();

    if (_head.data == data)
    {
        if (_elementsCount == 1)
            _head = null;
        else
            _head = _head.next;

        _elementsCount--;
        return;
    }
}

```

```

Node currentNode = _head;

```

```

        while (currentNode.next != null)
        {
            if (currentNode.next.data == data)
            {
                currentNode.next = currentNode.next.next;
                _elementsCount--;
                return;
            }

            currentNode = currentNode.next;
        }

        throw new InvalidOperationException();
    }

    public bool Contains(float data)
    {
        if (_elementsCount == 0)
            return false;

        foreach (float listElement in this)
            if (listElement == data)
                return true;

        return false;
    }

    public IEnumerator<float> GetEnumerator()
    {
        return new ListEnumerator(this);
    }

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

```

Файл ListEnumerator.cs

```
using System.Collections;
using System.Collections.Generic;

namespace linked_list
{
    public class ListEnumerator : IEnumerator<float>
    {
        private LinkedListMod _list;
        private int _index;

        public ListEnumerator(LinkedListMod list)
        {
            _list = list;
            _index = -1;
        }
        public float Current => _list[_index];

        object IEnumerator.Current => _list[_index];

        public void Dispose()
        {
        }

        public bool MoveNext()
        {
            _index++;

            if (_list.Count() != 0 && _index < _list.Count())
                return true;
            else
                return false;
        }

        public void Reset()
        {
            _index = -1;
        }
    }
}
```

```
}
```

Файл Program.cs

```
using System;
```

```
namespace linked_list
```

```
{
```

```
    public class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            LinkedListMod list = [1f, 7f, 9f, 3f, 5f];
```

```
            float n;
```

```
            Console.WriteLine("Initial list:");
```

```
            PrintList(list);
```

```
            Console.WriteLine("\n1-st task");
```

```
            Console.WriteLine("Enter the value to find the first value in the list 2 times  
bigger:");
```

```
            EnterFloat(out n);
```

```
            FindX2Value(n, list);
```

```
            Console.WriteLine($"2-nd task\nThere are {AmountOfBiggerElements(3.14f,  
list)} elements in the list bigger than 3.14");
```

```
            Console.WriteLine("\n3-rd task\nEnter the value to get a list of elements bigger  
than that value:");
```

```
            EnterFloat(out n);
```

```
            LinkedListMod bigList = GetArrayOfBiggerElements(n, list);
```

```
            Console.WriteLine($"List of elements bigger than {n}:");
```

```
            PrintList(bigList);
```

```
            Console.WriteLine("\n4-th task\nElements bigger than average removed:");
```

```
            RemoveElementsMoreThanAverage(list);
```

```

        PrintList(list);
    }

    static void EnterFloat(out float value)
    {
        while (!float.TryParse(Console.ReadLine(), out value))
        {
            Console.WriteLine("Invalid input. Try again.");
        }
    }

    static void PrintList(LinkedListMod linkedList)
    {
        foreach (var item in linkedList)
        {
            Console.Write(item + " ");
        }
        Console.WriteLine();
    }

    static void FindX2Value(float value, LinkedListMod linkedList)
    {
        foreach (float item in linkedList)
        {
            if (item >= 2f * value)
            {
                Console.WriteLine($"Value {item} inside the list is more than 2 times
bigger than {value}.");
                return;
            }
        }

        Console.WriteLine($"There is no value 2 times bigger than {value} inside the
list");
    }

    static int AmountOfBiggerElements(float value, LinkedListMod linkedList)
    {
        int count = 0;

```



```

        foreach (float item in linkedList)
            if (item > value)
                count++;

        return count;
    }

    static LinkedListMod GetArrayOfBiggerElements(float value, LinkedListMod
linkedList)
    {
        LinkedListMod newList = new LinkedListMod();

        foreach (float item in linkedList)
            if (item > value)
                newList.Add(item);

        return newList;
    }

    static void RemoveElementsMoreThanAverage(LinkedListMod linkedList)
    {
        if (linkedList.Count() == 0)
            return;

        float average = 0;

        foreach (float item in linkedList)
            average += item;

        average /= linkedList.Count();

        foreach (float item in linkedList)
            if (item > average)
                linkedList.Remove(item);
    }
}

```

Тестування програми:

```
Initial list:
1 7 5 3 9

1-st task
Enter the value to find the first value in the list 2 times bigger:
2,2
Value 7 inside the list is more than 2 times bigger than 2,2.

2-nd task
There are 3 elements in the list bigger than 3.14

3-rd task
Enter the value to get a list of elements bigger than that value:
3,56
List of elements bigger than 3,56:
7 5 9

4-th task
Elements bigger than average removed:
1 5 3
```

Висновок: Під час виконання лабораторної роботи було досліджено типи лінійних та нелінійних структур даних та отримано навички користування бібліотечними реалізаціями структур даних та побудови власних.