КПІ ім. Ігоря Сікорського

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт до комп'ютерного практикуму з курсу

"Основи програмування. Частина 2"

Прийняв
Ст. викладач кафедри ІПІ
Крамар Ю.М.
"14" квітня 2025 р.

Виконав
Студент групи ІП-41
Сокуренко Б.В.

Київ 2024

Комп’ютерний практикум №7

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

Завдання:

Написати програму мовою C#, де описати власну структуру даних згідно з варіантом (Табл.1), створити 2 проекти, в одному – має бути функціонал списку, в другому - його використання. Виведення на консоль даних зі списку вважати використанням списку, а не його функціоналом! В списку потрібно передбачити крім функціональності, заданої варіантом, можливість отримати з нього значення, наприклад, стандартним оператором foreach. Додати до списку функцію індексації (елементи списку мають бути доступні на читання за індексом). Додати до списку операцію видалення елементу за номером (номер елементу задається параметром). Єдиний стиль найменувань обов'язковий. Застосовувати в коді лише XML-коментарі.

Продемонструвати функціональність розробленої структури шляхом застосування всіх її операцій (створити декілька об’єктів структур, додати в них певну кількість значень, зчитати значення зі списку, викликати операції над значеннями відповідно варіанту).

В завданні, де вказується на літерал цілого типу, дійсного, символьного та ін. (наприклад, «!»: знайти перше входження символу «!»), значення літералу не “зашивати” у код операції, а передавати як параметр (тобто передбачити можливість виконання операції з різними значеннями):

11	float	Односпрямований	Включення після другого елементу списку	1.Знайти перший від’ємний елемент списку. 2.Знайти суму елементів більших за середнє значення. 3.Отримати новий список зі значень позитивних елементів поточного списку. 4. Видалити всі від’ємні елементи поточного списку.

Текст програмної реалізації класу Node(вузол)

```
public class Node
{
   public float Data { get; set; }
   public Node? Next { get; set; }

   public Node(float data)
   {
      Data = data;
      Next = null;
   }
```

```
    }
```

Текст програмної реалізації класу myLinkedList(зв'язний список)

```
public class LinkedList
{
    private Node? head;
    private int number;

    public void AddToEnd(float value)
    {
        Node newNode = new Node(value);
        if (head == null)
        {
            head = newNode;
        }
        else
        {
            Node current = head;
            while (current.Next != null)
            {
                current = current.Next;
            }
            current.Next = newNode;
        }
        number++;
    }

    public void AddAfterSecond(float value)
    {
        if (head == null || head.Next == null)
        {
            Console.WriteLine("The list has less than two elements");
            return;
        }

        Node newNode = new Node(value);
        Node second = head.Next;

        newNode.Next = second.Next;
        second.Next = newNode;
        number++;
    }
    public int Size
    {
        get { return number; }
    }
```

```
public float this[int index]
{
   get
   {
      if (index < 0 || index >= number)
      {
         throw new IndexOutOfRangeException("Index is out of range.");
      }
      Node current = head;
      for (int i = 0; i < index; i++)
      {
         current = current.Next;
      }
      return current.Data;
   }
   set
   {
      if (index < 0 || index >= number)
      {
         throw new IndexOutOfRangeException("Index is out of range.");
      }
      Node current = head;
      for (int i = 0; i < index; i++)
      {
         current = current.Next;
      }
      current.Data = value;
   }
}



public bool deleteElement(int elNum)
{
   if (elNum >= 0 && elNum < Size)
   {
      if (elNum == 0)
      {
         head = head.Next;
      }
      else
      {
         Node current = head;
         Node previous = null;
         for (int i = 0; i < elNum; i++)
         {
            previous = current;
```

```
                current = current.Next;
            }
            previous.Next = current.Next;

        }
        number--;
        return true;
    }
    else
    {
        Console.WriteLine("Invalid index");
        return false;
    }
}

public float? FirstNegativeEl()
{
    Node current = head;
    while (current != null)
    {
        if (current.Data < 0)
        {
            return current.Data;
        }
        current = current.Next;
    }
    return null;
}


public float SumOfElementsGreaterThanMedian()
{

    Node current = head;
    float sumOfAll = 0;
    while (current != null)
    {

        sumOfAll += current.Data;
        current = current.Next;
    }

    float Median = sumOfAll / number;


    float SumOfElemenetsGreaterThanMedian = 0;
    current = head;
    while(current != null)
```

```
        {
            if(current.Data > Median)
            {
                SumOfElemenetsGreaterThanMedian += current.Data;
            }
            current = current.Next;
        }

        return SumOfElemenetsGreaterThanMedian;
    }

    public LinkedList PostitiveElementsList()
    {
        LinkedList newList = new LinkedList();
        Node current = head;
        while (current != null)
        {
            if (current.Data > 0)
                newList.AddToEnd(current.Data);
            current = current.Next;
        }

        return newList;

    }

    public void deleteAllNegativeElements()
    {
        int i = 0;
        while (i < Size)
        {
            if (this[i] < 0)
            {
                deleteElement(i);
            }
            else
            {
                i++;
            }
        }
    }


}
```

Текст програми, що використана для демонстрації можливостей описаних класів

```
using System.ComponentModel;
using System;
```

```
using System.Collections.Generic;
using System.Text;
using System.Linq;
using LinkedListLibrary;

class Program
{
   static void Main()
   {
      var list = new LinkedList();
      list.AddToEnd(1.1f);
      list.AddToEnd(2.2f);
      list.AddToEnd(3.3f);
      list.AddToEnd(-3.3f);

      list.AddAfterSecond(9.9f);


      list.AddAfterSecond(11.1f);
      PrintList(list);

      bool isDeleted = false;
      int input = 0;
      while (!isDeleted)
      {
         Console.WriteLine("Choose the number of element to delete");
         input = int.Parse(Console.ReadLine()) - 1;
         isDeleted = list.deleteElement(input);
      }


      Console.WriteLine($"\nAfter deleting element {input+1}:");
      PrintList(list);

      float? result = list.FirstNegativeEl();

      if (result.HasValue)
         Console.WriteLine("\nFirst negative element: " + result.Value);
      else
         Console.WriteLine("\nNo negative elements found.");


      float greaterSum = list.SumOfElementsGreaterThanMedian();


      Console.WriteLine("\nThe sum of elements greater than median:" + greaterSum);
```

```
        LinkedList newList = list.PostitiveElementsList();

        Console.WriteLine("\nThe new list of postive values:");
        PrintList(newList);

        list.deleteAllNegativeElements();

        Console.WriteLine("\nThe old list of postive values:");
        PrintList(list);
    }
    static void PrintList(LinkedList list)
    {
        for (int i = 0; i < list.Size; i++)
        {
            if (i > 0)
                Console.Write(" -> ");
            Console.Write(list[i]);
        }
        Console.WriteLine();
    }

}
```

Приклад роботи