Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної

техніки Кафедра ІПІ

Звіт

з лабораторної роботи № 7 з дисципліни

«Основи програмування (частина 2). Методології програмування.»

„ **Побудова та використання структур даних**”

Виконав(ла) ІП-43 Ткачук Дмитро Вікторович
(шифр, прізвище, ім'я, по батькові)

Перевірив Куценко Микита Олександрович
(прізвище, ім'я, по батькові)

Київ 2025

Лабораторна робота 7

ПОБУДОВА ТА ВИКОРИСТАННЯ СТРУКТУР ДАНИХ

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

Варіант: 8

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).

Таблиця 1 – Завдання 8-го варіанту.

№	Тип даних елементів	Тип списку	Спосіб додавання елементу списку	Операції зі списком
8	int	Двоспрямований	Включення в кінець списку	1.Знайти перше входження заданого елементу. 2. Знайти суму елементів, які розташовані на парних позиціях списку. 3.Отримати новий список зі значень елементів, значення яких більші за задане значення. 4.Видалити елементи, які більші за середнє значення.

Код програми

Код бібліотеки класів

Файл ListNode.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DoublyLinkedListLibrary
{
   public class ListNode
   {
      private int value;
      private ListNode previous;
      private ListNode next;

      public ListNode Previous { get { return previous; } set { previous = value; } }
      public ListNode Next { get { return next; } set { next = value; } }
      public int Value { get { return value; } set { this.value = value; } }
```

```
        public ListNode(int value, ListNode previous, ListNode next)
        {
            this.value = value;
            this.previous = previous;
            this.next = next;
        }
    }
}
```

Файл DoublyLinkedList.cs:

```
using System;
using System.CodeDom;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DoublyLinkedListLibrary
{
    public class DoublyLinkedList : IEnumerable<int>
    {
        private ListNode head;
        private ListNode tail;
        private int length;

        public ListNode Head {  get { return head; } }
        public ListNode Tail { get { return tail; } }
        public int Length { get { return length; } }

        public DoublyLinkedList()
        {
            length = 0;
        }

        public int this[int nodeIndex]
        {
            get
            {
                IndexValidation(nodeIndex);
                ListNode node = GetNodeByTheIndex(nodeIndex);
                return node.Value;
            }
            set
            {
                IndexValidation(nodeIndex);
                ListNode node = GetNodeByTheIndex(nodeIndex);
                node.Value = value;
            }
        }

        public void AddNode(int nodeValue)
        {
            ListNode node;
            if(tail is ListNode)
            {
                node = new ListNode(nodeValue, tail, null);
                tail.Next = node;
                tail = node;
            }
            else
            {
                node = new ListNode(nodeValue, null, null);
                head = node;
                tail = node;
```

```
    }
    length++;
}

public void RemoveNode(int nodeIndex)
{
    IndexValidation(nodeIndex);
    ListNode node = GetNodeByTheIndex(nodeIndex);
    ListNode previousNode = node.Previous;
    ListNode nextNode = node.Next;

    if (previousNode != null)
    {
        previousNode.Next = nextNode;
    }
    else
    {
        head = nextNode;
        if (nextNode != null)
            nextNode.Previous = null;
    }

    if (nextNode != null)
    {
        nextNode.Previous = previousNode;
    }
    else
    {
        tail = previousNode;
        if (previousNode != null)
            previousNode.Next = null;
    }

    length--;
}

public int FirstEntry(int value)
{
    int counter = 0;
    foreach (var nodeValue in this)
    {
        if(nodeValue == value)
        {
            return counter;
        }
        counter++;
    }
    return -1;
}

public int FindSumOnOddPositions()
{
    int sum = 0;
    int position = 1;
    foreach(var nodeValue in this)
    {
        if(position % 2 != 0)
        {
            sum += nodeValue;
        }
        position++;
    }
    return sum;
}

public DoublyLinkedList GetNewListWithHigherElements(int value)
```

```
{
    DoublyLinkedList newList = new DoublyLinkedList();
    foreach (var nodeValue in this)
    {
        if(nodeValue > value)
        {
            newList.AddNode(nodeValue);
        }
    }
    return newList;
}

public void RemoveElementsHigherAverageValue()
{
    if (length == 0)
        return;

    double averageValue = 0.0;
    int elementsSum = 0;

    foreach (var nodeValue in this)
    {
        elementsSum += nodeValue;
    }

    averageValue = (double)elementsSum / length;

    for (int i = length - 1; i >= 0; i--)
    {
        if (this[i] > averageValue)
        {
            RemoveNode(i);
        }
    }
}

private void IndexValidation(int index)
{
    if (index >= length || index < 0)
    {
        throw new IndexOutOfRangeException();
    }
}

private ListNode GetNodeByTheIndex(int nodeIndex)
{
    ListNode node = head;
    for (int i = 0; i < nodeIndex; i++)
    {
        node = node.Next;
    }
    return node;
}

public IEnumerator<int> GetEnumerator()
{
    ListNode current = head;
    while (current != null)
    {
        yield return current.Value;
        current = current.Next;
    }
}

IEnumerator IEnumerable.GetEnumerator()
{
```

```
            return GetEnumerator();
        }
    }
}
```

Код консольного додатку

Файл Program.cs:

```
using System;
using System.Collections.Generic;
using System.Diagnostics.Eventing.Reader;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using DoublyLinkedListLibrary;


namespace ConsoleApp3
{
    internal class Program
    {

        enum UserMenuOptions {ShowMenu, ShowList, AddElement, RemoveElement, ChangeElement, FindFirstEntry,
FindSumOnOddPositions,
            GetNewListWithHigherElements, RemoveElementsHigherThanAverageValue, Exit};

        const int MIN_ELEMENT_VALUE = -1000000;
        const int MAX_ELEMENT_VALUE = 1000000;

        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.UTF8;
            Console.InputEncoding = Encoding.UTF8;

            Console.WriteLine("Лабораторна робота 7 - Робота з двозв'язним списком.\n");
            DoublyLinkedList doublyLinkedList = new DoublyLinkedList();

            bool isTerminate = false;
            ShowMenu();

            do
            {
                int option = OptionChoosing();
                switch (option)
                {
                    case (int)UserMenuOptions.ShowMenu:
                        ShowMenu();
                        break;
                    case (int)UserMenuOptions.ShowList:
                        Console.WriteLine("\n--- Виведення списку ---");
                        ShowList(doublyLinkedList);
                        break;
                    case (int)UserMenuOptions.AddElement:
                        Console.WriteLine("\n--- Додавання елементу в список ---");
                        AddElementToTheList(doublyLinkedList);
                        break;
                    case (int)UserMenuOptions.RemoveElement:
                        Console.WriteLine("\n--- Видалення елементу зі списку ---");
                        RemoveElement(doublyLinkedList);
                        break;
                    case (int)UserMenuOptions.ChangeElement:
                        Console.WriteLine("\n--- Зміна значення елементу списку ---");
                        ChangeElement(doublyLinkedList);
```

```
                break;
            case (int)UserMenuOptions.FindFirstEntry:
                Console.WriteLine("\n--- Пошук першого входження елементу у список ---");
                FindFirstEntry(doublyLinkedList);
                break;
            case (int)UserMenuOptions.FindSumOnOddPositions:
                Console.WriteLine("\n--- Розрахунок суми елементів на непарних позиціях ---");
                FindSumOnOddPositions(doublyLinkedList);
                break;
            case (int)UserMenuOptions.GetNewListWithHigherElements:
                Console.WriteLine("\n--- Виведення списку елементів, значення яких вище зазначеного ---");
                GetNewListWithHigherElements(doublyLinkedList);
                break;
            case (int)UserMenuOptions.RemoveElementsHigherThanAverageValue:
                Console.WriteLine("\n--- Видалення елементів списку, які вище середнього значення ---");
                RemoveElementsHigherThanAverageValue(doublyLinkedList);
                break;
            case (int)UserMenuOptions.Exit:
                isTerminate = true;
                break;
        }
        Console.WriteLine("\n");
    } while(!isTerminate);
}

static void ShowMenu()
{
    Console.WriteLine($"Оберіть дію, яку бажаєте зробити:" +
        $"\n {(int)UserMenuOptions.ShowMenu} - Вивести меню;" +
        $"\n {(int)UserMenuOptions.ShowList} - Вивести список в консоль;" +
        $"\n {(int)UserMenuOptions.AddElement} - Додати елемент вкінець списку;" +
        $"\n {(int)UserMenuOptions.RemoveElement} - Видалити елемент зі списку;" +
        $"\n {(int)UserMenuOptions.ChangeElement} - Змінити значення елемента;" +
        $"\n {(int)UserMenuOptions.FindFirstEntry} - Знайти перше входження елементу в список;" +
        $"\n {(int)UserMenuOptions.FindSumOnOddPositions} - Знайти суму елементів на непарних позиціях;" +
        $"\n {(int)UserMenuOptions.GetNewListWithHigherElements} - Вивести список з елементів, вище заданого
значення;" +
        $"\n {(int)UserMenuOptions.RemoveElementsHigherThanAverageValue} - Видалити елементи, значення
яких більше за середнє;" +
        $"\n {(int)UserMenuOptions.Exit} - Завершити виконання програми.");
}

static int OptionChoosing()
{
    int optionNumber = -1;
    bool isError = false, res = false;
    do {
        Console.Write("Обріть дію, яку бажаєте виконати до списку(введіть числовий номер дії): ");
        string userInput = Console.ReadLine();
        res = int.TryParse(userInput, out optionNumber);

        if (!res)
        {
            Console.WriteLine("Помилковий ввід. Очікувалось ціле число. Повторіть спробу.");
            isError = true;
        }
        else if (optionNumber < (int)UserMenuOptions.ShowMenu || optionNumber > (int)UserMenuOptions.Exit)
        {
            Console.WriteLine("Помилковий ввід. Немає даної дії. Повторіть спробу.");
            isError = true;
        }
        else
        {
            isError = false;
        }
    }
```

```
        while( isError );
        return optionNumber;
    }

    static void ShowList(DoublyLinkedList doublyLinkedList)
    {
        if(doublyLinkedList.Length != 0)
        {
            foreach (var listElement in doublyLinkedList)
            {
                Console.Write($"{listElement}    ");
            }
        }
        else
        {
            Console.WriteLine("Наразі список порожній");
        }
    }

    static void AddElementToTheList(DoublyLinkedList doublyLinkedList)
    {
        int number = ElementValueInput();
        doublyLinkedList.AddNode(number);
        Console.WriteLine("Елемент був вдало доданий до списку.");
    }

    static void RemoveElement(DoublyLinkedList doublyLinkedList)
    {
        if( doublyLinkedList.Length == 0)
        {
            Console.WriteLine("Список порожній.");
            return;
        }
        int index = -1;
        bool isError = false, res = false;
        do
        {
            Console.Write("Введіть індекс елемента, який потрібно видалити: ");
            string userInput = Console.ReadLine();
            res = int.TryParse(userInput, out index);
            if (!res || index < 0)
            {
                Console.WriteLine("Помидковий ввід. Очікувалось ціле додатнє число. Повторіть спробу.");
                isError = true;
            }

            try
            {
                doublyLinkedList.RemoveNode(index);
                Console.WriteLine("Елемент було успішно видалено.");
            }
            catch (IndexOutOfRangeException e)
            {
                Console.WriteLine("Помилковий ввід. Введений індекс не входить до коректного діапазону. Повторіть
спробу.");
                isError = true;
            }

        }
        while(isError);
    }

    static void ChangeElement(DoublyLinkedList doublyLinkedList)
    {
        if (doublyLinkedList.Length == 0)
        {
```

```
            Console.WriteLine("Список порожній.");
            return;
        }
        int index = -1;
        bool isError = false, res = false;
        do
        {
            Console.Write("Введіть індекс елемента, який потрібно змінити: ");
            string userInput = Console.ReadLine();
            res = int.TryParse(userInput, out index);
            if (!res || index < 0)
            {
                Console.WriteLine("Помидковий ввід. Очікувалось ціле додатнє число. Повторіть спробу.");
                isError = true;
            }
            try
            {
                int newValue  = ElementValueInput();
                doublyLinkedList[index] = newValue;
                Console.WriteLine($"Значення елементу {index} було успішно змінено.");
                isError = false;
            }
            catch (IndexOutOfRangeException e)
            {
                Console.WriteLine("Помилковий ввід. Введений індекс не входить до коректного діапазону. Повторіть
спробу.");
                isError = true;
            }

        }
        while (isError);
    }

    static void FindFirstEntry(DoublyLinkedList doublyLinkedList)
    {
        int nuber = ElementValueInput();
        int elementIndex = doublyLinkedList.FirstEntry(nuber);
        if(elementIndex != -1)
        {
            Console.WriteLine($"Індекс першого входження: {elementIndex}");
        }
        else
        {
            Console.WriteLine($"Елементу не має у списку");
        }
    }

    static void FindSumOnOddPositions(DoublyLinkedList doublyLinkedList)
    {
        if(doublyLinkedList.Length != 0)
        {
            Console.WriteLine($"Сума елементів, які розташовані на непарних позиціях у списку:
{doublyLinkedList.FindSumOnOddPositions()}");
        }
        else
        {
            Console.WriteLine("Список порожній.");
        }
    }

    static void GetNewListWithHigherElements(DoublyLinkedList doublyLinkedList)
    {
        int number = ElementValueInput();
        DoublyLinkedList resultList = doublyLinkedList.GetNewListWithHigherElements(number);
        ShowList(resultList);
    }
```

```
        static void RemoveElementsHigherThanAverageValue(DoublyLinkedList doublyLinkedList)
        {
            doublyLinkedList.RemoveElementsHigherAverageValue();
            ShowList(doublyLinkedList);
        }

        static int ElementValueInput()
        {
            int number;
            bool isError = false, res = false;
            do
            {
                Console.Write("Введіть числове значення: ");
                string userInput = Console.ReadLine();
                res = int.TryParse(userInput, out number);
                if (!res)
                {
                    Console.WriteLine("Помидковий ввід. Очікувалось ціле число. Повторіть спробу.");
                    isError = true;
                }
                else if (number < MIN_ELEMENT_VALUE || number > MAX_ELEMENT_VALUE)
                {
                    Console.WriteLine($"Помилковий ввід. Число повинно бути в діаіпазоні [{MIN_ELEMENT_VALUE},
{MAX_ELEMENT_VALUE}]");
                    isError = true;
                }
                else
                {
                    isError = false;
                }
            }
            while (isError);
            return number;
        }
    }
}
```

Висновок: Під час виконання лабораторної роботи було реалізовано двозв’язний список — структуру даних, у якій кожен вузол містить значення та посилання на попередній і наступний елементи. Було створено основні операції: додавання, видалення, доступ і зміна значення за індексом, а також реалізовано додаткові методи згідно з варіантом завдання, зокрема пошук першого входження, створення нового списку за умовою та видалення елементів, що перевищують середнє значення. Робота списку та його операцій була протестована за допомогою створеного консольного додатку.