**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 Виконав

Ст. викладач кафедри ІПІ Студент групи ІП-41

Вітковська І.І. Коломієць М.О.

“16” квітня 2025 р.

**Київ 2024**

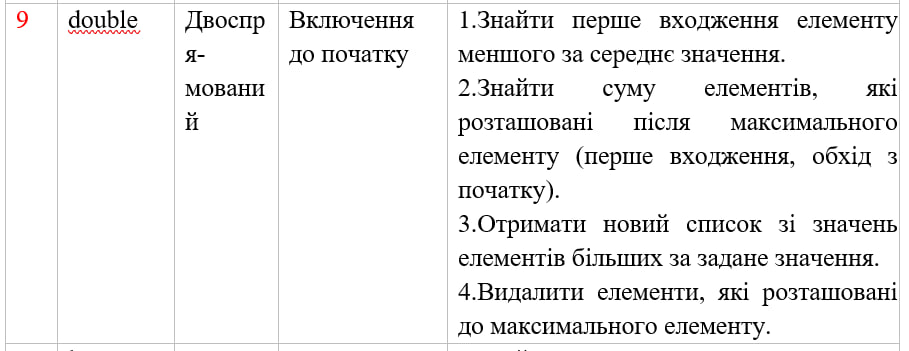
**Лабораторна  робота 7  
ПОБУДОВА ТА ВИКОРИСТАННЯ СТРУКТУР ДАНИХ**

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

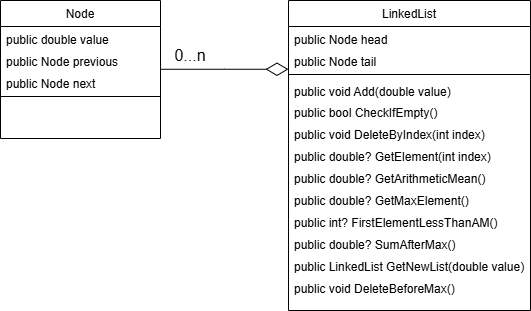
**Завдання:** Написати програму мовою С#, де описати власну структуру даних згідно з варіантом, створити 2 проекти, в одному – має бути функціонал списку, в другому - його використання. В списку потрібно передбачити крім функціональності, заданої варіантом, можливість отримати з нього значення. Додати 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).

Продемонструвати функціональність розробленої структури шляхом застосування всіх її операцій.

**Варіант:**



**Діаграма класів**

****

**Програмна реалізація**

**Клас Node:**

class Node

{

public double value;

public Node previous;

public Node next;

public Node(double val)

{

value = val;

previous = null;

next = null;

}

}

**Клас LinkedList:**

class LinkedList

{

public Node head;

public Node tail;

public LinkedList()

{

head = null;

tail = null;

}

public void Add(double value)

{

Node node = new Node(value);

if (head == null)

{

head = node;

tail = node;

}

else

{

head.previous = node;

node.next = head;

head = node;

}

}

public bool CheckifEmpty()

{

if (this.head == null)

{

return true;

}

return false;

}

public void DeleteByIndex(int index)

{

if (this.CheckifEmpty() || index < 0)

{

return;

}

else if (index == 0)

{

if (this.head.next != null)

{

this.head = this.head.next;

this.head.previous = null;

}

else

{

this.head = null;

this.tail = null;

}

return;

}

else

{

Node node = this.head;

int i = 0;

while (node != null)

{

if (i == index)

{

if(node.next != null)

{

node.previous.next = node.next;

node.next.previous = node.previous;

}

else

{

node.previous.next = null;

}

node = null;

return;

}

i++;

node = node.next;

}

return;

}

}

public double? GetElement(int index)

{

if(index < 0 || this.CheckifEmpty())

{

return null;

}

Node node = this.head;

int i = 0;

while (node != null)

{

if (i == index)

{

return node.value;

}

i++;

node = node.next;

}

return null;

}

public double? this[int index]

{

get { return GetElement(index); }

}

public double? GetArithmeticMean()

{

if(this.CheckifEmpty())

{

return null;

}

else

{

Node node = this.head;

double sum = 0;

int n = 0;

while (node!=null)

{

sum += node.value;

n++;

node= node.next;

}

return sum / n;

}

}

public double? GetMaxElement()

{

if (this.CheckifEmpty())

{

return null;

}

else

{

Node node = this.head;

double max = node.value;

node = node.next;

while (node != null)

{

if (node.value > max)

{

max = node.value;

}

node = node.next;

}

return max;

}

}

public int? FirstElementLessThanAM()

{

if (this.CheckifEmpty() || this.head.next == null)

{

return null;

}

else

{

double? arithmeticmean = this.GetArithmeticMean();

Node node = this.head;

int n = 0;

while (node != null)

{

if (node.value < arithmeticmean)

{

return n;

}

node = node.next;

n++;

}

return null;

}

}

public double? SumAfterMax()

{

if (this.CheckifEmpty() || this.head.next == null)

{

return null;

}

else

{

double? max = this.GetMaxElement();

double sum=0;

Node node = this.head;

while (node.value!=max)

{

node = node.next;

}

node= node.next;

while(node != null)

{

sum+= node.value;

node = node.next;

}

return sum;

}

}

public LinkedList GetNewList(double value)

{

LinkedList list = new LinkedList();

if (this.CheckifEmpty())

{

return list;

}

Node node = this.head;

while (node != null)

{

if(node.value > value)

{

list.Add(node.value);

}

node= node.next;

}

return list;

}

public void DeleteBeforeMax()

{

if (this.CheckifEmpty() || this.head.next==null)

{

return;

}

double? max = this.GetMaxElement();

Node node = this.head;

while (node.value != max)

{

node= node.next;

this.DeleteByIndex(0);

}

}

}

**Тестування програми**

