**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 № 1 з дисципліни

«Сучасні технології розробки WEB-застосувань на платформі Microsoft.NET»

«Узагальнені типи (Generic) з підтримкою подій. Колекції»

**Виконав**

(шифр, прізвище, ім'я, по батькові)

*ІП-15, Тонконог В.В.*

**Перевірив**

(прізвище, ім'я, по батькові)

*Бардін В.*

Київ 2023

Варіант 8

Завдання



Код виконання:

Мій кільцевий список:

public sealed class MyLinkedList<T> : ICollection<T>

{

public MyLinkedListNode<T>? Head { get; private set; }

public MyLinkedListNode<T>? Tail { get; private set; }

public event EventHandler<MyLinkedListEventArgs<T>> AddedNode;

public event EventHandler<MyLinkedListEventArgs<T>> RemovedNode;

private int \_count;

public MyLinkedList()

{

\_count = 0;

}

public MyLinkedList(IEnumerable<T> list)

{

if (list == null)

throw new ArgumentNullException("list can`t be null");

foreach (var item in list)

{

Add(item);

}

}

public int Count => \_count;

public bool IsReadOnly => false;

#region Addings

public void Add(T item)

{

ArgumentNullException.ThrowIfNull(item);

if (\_count <= 0)

{

AddToEmptyList(item);

}

else

{

var newNode = new MyLinkedListNode<T>(Head!, Tail!, item);

Tail!.Next = newNode;

Tail = newNode;

Head!.Previous = Tail;

\_count++;

AddedNode?.Invoke(this, new MyLinkedListEventArgs<T>(newNode.Value));

}

}

public void Add(MyLinkedListNode<T> node)

{

Add(node.Value);

}

public void AddFirst(T item)

{

AddFirst(new MyLinkedListNode<T>(item));

}

public void AddFirst(MyLinkedListNode<T> node)

{

if (node == null || node.Value == null)

throw new ArgumentNullException();

if (\_count > 0)

{

MyLinkedListNode<T>? second = Head;

second!.Previous = node;

if(\_count == 1)

{

Tail = second;

}

var newNode = new MyLinkedListNode<T>(Head!, Tail!, node.Value);

Head!.Previous = newNode;

Head = newNode;

Tail!.Next = Head;

\_count++;

AddedNode?.Invoke(this, new MyLinkedListEventArgs<T>(node.Value));

}

else

{

AddToEmptyList(node.Value);

}

}

private void AddToEmptyList(T value)

{

Head = new MyLinkedListNode<T>(value);

Head.Next = Head;

Head.Previous = Head;

Tail = Head;

\_count = 1;

AddedNode?.Invoke(this, new MyLinkedListEventArgs<T>(value));

}

#endregion

#region Removings

public bool Remove(T item)

{

var node = new MyLinkedListNode<T>(item);

return Remove(node);

}

public bool Remove(MyLinkedListNode<T> item)

{

var node = Find(item.Value);

if (node == null)

return false;

if(\_count == 1)

{

Clear();

RemovedNode?.Invoke(this, new MyLinkedListEventArgs<T>(node.Value));

return true;

}

if (node == Head)

Head = Head.Next;

if (node == Tail)

Tail = Tail.Previous;

node.Previous.Next = node.Next;

node.Next.Previous = node.Previous;

\_count--;

RemovedNode?.Invoke(this, new MyLinkedListEventArgs<T>(node.Value));

return true;

}

public void Clear()

{

Head = null;

Tail = null;

\_count = 0;

}

#endregion

#region Findings

public bool Contains(T item)

{

return Find(item) != null;

}

public MyLinkedListNode<T>? Find(T values)

{

MyLinkedListNode<T>? current = Head;

foreach(var value in this)

{

if (value!.Equals(values))

{

return current;

}

current = current!.Next;

}

return null;

}

#endregion

#region Copy

public void CopyTo(T[] array, int arrayIndex)

{

ArgumentNullException.ThrowIfNull(array);

if (arrayIndex < 0 || arrayIndex > array.Length)

{

throw new ArgumentOutOfRangeException();

}

if (array.Length - arrayIndex < \_count)

{

throw new ArgumentException();

}

foreach (var item in this)

{

array[arrayIndex++] = item;

}

}

#endregion

#region Enumerations

public IEnumerator<T> GetEnumerator()

{

if (Head == null)

yield break;

MyLinkedListNode<T> current = Head;

do

{

yield return current!.Value;

current = current.Next!;

}

while (current != Head);

}

IEnumerator IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

#endregion

}

public sealed class MyLinkedListNode<T>

{

public MyLinkedListNode<T> Next { get; set; }

public MyLinkedListNode<T> Previous { get; set; }

public T Value { get; set; }

public MyLinkedListNode(T value)

{

Value = value;

Next = Previous = null!;

}

public MyLinkedListNode(MyLinkedListNode<T> next, MyLinkedListNode<T> previous, T value) : this(value)

{

Next = next;

Previous = previous;

}

}

Приклад виконання дій з списком:

