



Міністерство освіти і науки України КПІ ім. Ігоря Сікорського

Факультет Інформатики та Обчислювальної Техніки

ЗВІТ

Лабораторна робота №1 з дисципліни

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

Перевірив:

Викладач кафедри ІСТ

ФІОТ

Бардін В.

Виконала:

Жеглова Євгенія

гр. ІК-13

Узагальнені типи (Generic) з підтримкою подій. Колекції

Мета лабораторної роботи – навчитися проектувати та реалізовувати узагальнені типи, а також типи з підтримкою подій.

Завдання:

1. Розробити клас власної узагальненої колекції, використовуючи стандартні інтерфейси колекцій із бібліотек System.Collections та System.Collections.Generic. Стандартні колекції при розробці власної не застосовувати. Для колекції передбачити методи внесення даних будь-якого типу, видалення, пошуку та ін. (відповідно до типу колекції).
2. Додати до класу власної узагальненої колекції підтримку подій та обробку виключних ситуацій.
3. Опис класу колекції та всіх необхідних для роботи з колекцією типів зберегти у динамічній бібліотеці.
4. Створити консольний додаток, в якому продемонструвати використання розробленої власної колекції, підписку на події колекції.

8	Кільцевий список	Див. List<T>, LinkedList<T>	Збереження даних за допомогою динамічно зв'язаного списку
---	------------------	-----------------------------	---

Код програми

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

public class MyLinkedList<T> : IEnumerable<T>
{
    private Node<T> head;
    private Node<T> tail;
    private int count;

    public MyLinkedList()
    {
        head = null;
        tail = null;
        count = 0;
    }

    public int Count => count;

    private class Node<TNode>
    {
        public TNode Data { get; }
        public Node<TNode> Next { get; set; }
    }
}
```

```

public Node(TNode data)
{
    Data = data;
    Next = null;
}
}
public void Add(T item)
{
    if (item == null)
    {
        throw new ArgumentNullException($"{typeof(T)} {nameof(item)} is null");
    }

    Node<T> newNode = new Node<T>(item);
    if (head == null)
    {
        head = newNode;
        tail = newNode;
        tail.Next = head;
    }
    else
    {
        newNode.Next = head;
        tail.Next = newNode;
        tail = newNode;
    }
    count++;
    OnItemAdded(item);
}

public bool Remove(T item)
{
    if (item == null)
    {
        throw new ArgumentNullException($"{typeof(T)} {nameof(item)} is null");
    }

    if (head == null)
        return false;

    Node<T> current = head;
    Node<T> previous = null;

    do
    {
        if (current.Data.Equals(item))
        {
            if (previous != null)
            {
                previous.Next = current.Next;
                if (current == head)
                    head = current.Next;
                if (current == tail)
                    tail = previous;
            }
            else
            {
                head = head.Next;
                tail.Next = head;
            }
            count--;
            OnItemRemoved(item);
            return true;
        }
    }

    previous = current;

```

```

        current = current.Next;
    } while (current != head);

    return false;
}

public bool Contains(T item)
{
    if (head == null)
        return false;

    Node<T> current = head;

    do
    {
        if (current.Data.Equals(item))
        {
            OnItemContain(item);
            return true;
        }

        current = current.Next;
    } while (current != head);

    return false;
}

public void Clear(T item)
{
    head = null;
    tail = null;
    count = 0;
    OnItemClearUp(item);
}

public bool InsertAfter(T existingItem, T newItem)
{
    Node<T> newNode = new Node<T>(newItem);

    if (head == null)
        return false;

    Node<T> current = head;

    do
    {
        if (current.Data.Equals(existingItem))
        {
            newNode.Next = current.Next;
            current.Next = newNode;
            if (current == tail)
                tail = newNode;
            count++;
            OnItemAdded(newItem);
            return true;
        }

        current = current.Next;
    } while (current != head);

    return false;
}

public void AddFirst(T item)
{
    Node<T> newNode = new Node<T>(item);
    if (head == null)

```

```

    {
        head = newNode;
        tail = newNode;
        tail.Next = head;
    }
    else
    {
        newNode.Next = head;
        tail.Next = newNode;
        head = newNode;
    }
    count++;
    OnItemAddedFirst(item);
}

public event Action<T> ItemAdded;

public event Action<T> ItemRemoved;

public event Action<T> Contain;

public event Action<T> ClearUp;

public event Action<T> ItemAddedFirst;

protected virtual void OnItemAdded(T item)
{
    ItemAdded?.Invoke(item);
}

protected virtual void OnItemRemoved(T item)
{
    ItemRemoved?.Invoke(item);
}

protected virtual void OnItemContain(T item)
{
    Contain?.Invoke(item);
}

protected virtual void OnItemClearUp(T item)
{
    ClearUp?.Invoke(item);
}

protected virtual void OnItemAddedFirst(T item)
{
    ItemAddedFirst?.Invoke(item);
}

public IEnumerator<T> GetEnumerator()
{
    if (head != null)
    {
        Node<T> current = head;
        do
        {
            yield return current.Data;
            current = current.Next;
        } while (current != head);
    }
}

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

```

```

}

class Program
{
    static void Main()
    {
        MyLinkedList<int> list = new MyLinkedList<int>();

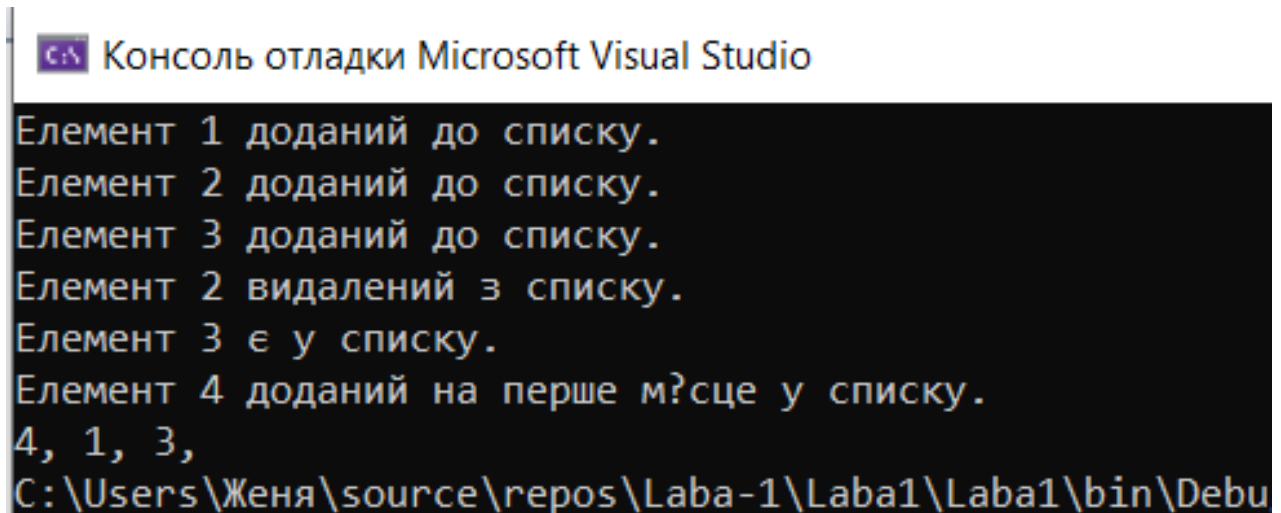
        list.ItemAdded += item => Console.WriteLine($"Елемент {item} доданий до списку.");
        list.ItemRemoved += item => Console.WriteLine($"Елемент {item} видалений з списку.");
        list.Contain += item => Console.WriteLine($"Елемент {item} є у списку.");
        list.ClearUp += item => Console.WriteLine($"Список чистий.");
        list.ItemAddedFirst += item => Console.WriteLine($"Елемент {item} доданий на перше місце у списку.");

        list.Add(1);
        list.Add(2);
        list.Add(3);
        list.Remove(2);
        list.Contains(3);
        list.AddFirst(4);

        foreach (var item in list)
        {
            Console.Write(item + ", ");
        }
    }
}

```

Результати роботи програми



```

C# Консоль отладки Microsoft Visual Studio
Елемент 1 доданий до списку.
Елемент 2 доданий до списку.
Елемент 3 доданий до списку.
Елемент 2 видалений з списку.
Елемент 3 є у списку.
Елемент 4 доданий на перше місце у списку.
4, 1, 3,
C:\Users\Женя\source\repos\Laba-1\Laba1\Laba1\bin\Debu

```