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»

Виконав студент ІП-15, Мочалов Дмитро
(шифр, прізвище, ім'я, по батькові)

Перевірив Бардін Владислав
(прізвище, ім'я, по батькові)

Київ 2023

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.

4	Дек (черга з двома кінцями)	Див. Queue<T>	Збереження даних за допомогою динамічно зв’язаного списку

Код:

```
public class Node<T>
{
    public T Data { get; set; }
    public Node<T>? Next { get; set; }
    public Node<T>? Previous { get; set; }

    public Node(T data)
    {
        Data = data;
        Next = null;
        Previous = null;
    }
}


public class Deque<T> : IEnumerable<T>
{
    private Node<T>? _head;
    private Node<T>? _tail;

    public event EventHandler? OnPushFront;
    public event EventHandler? OnPushBack;
    public event EventHandler? OnPopFront;
    public event EventHandler? OnPopBack;
    public event EventHandler? OnClear;
    public event EventHandler? OnReverse;

    public Deque()
    {
        _head = null;
        _tail = null;
    }

    public Deque(T item)
    {
        _head = new Node<T>(item);
        _tail = _head;
    }

    public void PushFront(T item)
    {
        var node = new Node<T>(item);
        if (_head == null)
        {
            _head = node;
            _tail = node;
        }
        else
        {
            node.Next = _head;
            _head.Previous = node;
```

```
            _head = node;
        }
        OnPushFront?.Invoke(this, EventArgs.Empty);
    }

    public T PopFront()
    {
        if (_head == null)
        {
            throw new InvalidOperationException("Deque is empty");
        }
        var item = _head.Data;
        _head = _head.Next;
        if (_head != null)
        {
            _head.Previous = null;
        }
        else
        {
            _tail = null;
        }

        OnPopFront?.Invoke(this, EventArgs.Empty);
        return item;
    }

    public T PeekFront()
    {
        if(_head == null)
        {
            throw new InvalidOperationException("Deque is empty");
        }

        return _head.Data;
    }

    public void PushBack(T item)
    {
        var node = new Node<T>(item);
        if (_tail == null)
        {
            _head = node;
            _tail = node;
        }
        else
        {
            node.Previous = _tail;
            _tail.Next = node;
            _tail = node;
        }
        OnPushBack?.Invoke(this, EventArgs.Empty);
    }
```

```
public T PopBack()
{
    if (_tail == null)
    {
        throw new InvalidOperationException("Deque is empty");
    }
    var item = _tail.Data;
    _tail = _tail.Previous;
    if (_tail != null)
    {
        _tail.Next = null;
    }
    else
    {
        _head = null;
    }
    OnPopBack?.Invoke(this, EventArgs.Empty);
    return item;
}

public T PeekBack()
{
    if(_tail == null)
    {
        throw new InvalidOperationException("Deque is empty");
    }

    return _tail.Data;
}

public bool IsEmpty()
{
    return _head == null;
}

public void Clear()
{
    _head = null;
    _tail = null;
    OnClear?.Invoke(this, EventArgs.Empty);
}

public bool Contains(T item)
{
    var node = _head;
    while (node != null)
    {
        if (node.Data != null && node.Data.Equals(item))
        {
            return true;
        }
```

```
            node = node.Next;
        }
        return false;
    }

    public void Reverse()
    {
        if (_head == null)
        {
            return;
        }
        var node = _head;
        while (node != null)
        {
            var temp = node.Next;
            node.Next = node.Previous;
            node.Previous = temp;
            node = temp;
        }
        (_head, _tail) = (_tail, _head);
        OnReverse?.Invoke(this, EventArgs.Empty);
    }

    public override string ToString()
    {
        var sb = new StringBuilder();
        sb.Append("[");
        var node = _head;
        while (node != null)
        {
            sb.Append(node.Data);
            node = node.Next;
            if (node != null)
            {
                sb.Append(", ");
            }
        }
        sb.Append("]");
        return sb.ToString();
    }

    public IEnumerator<T> GetEnumerator()
    {
        return new DequeEnumerator<T>(_head);
    }

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

```
public class DequeEnumerator<T> : IEnumerator<T>
{
    private readonly Node<T>? _head;
    private Node<T>? _current = null;
    private bool isStarted = false;

    public DequeEnumerator(Node<T> head)
    {
        _head = head;
    }

    public bool MoveNext()
    {
        if (!isStarted && _head != null)
        {
            _current = _head;
            isStarted = true;
            return HasNext();
        }
        if (_current != null && HasNext())
        {
            _current = _current.Next;
            return true;
        }

        return false;
    }

    private bool HasNext()
    {
        return _current.Next != null;
    }

    public void Reset()
    {
        _current = _head;
    }

    public T Current => _current.Data ;

    object IEnumerator.Current =>  _current.Data;

    public void Dispose()
    {
    }
}
```