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 № 2 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. Додати до проекту власної узагальненої колекції (застосувати виконану лабораторну роботу No1) проект модульних тестів, використовуючи певний фреймворк (Nunit, Xunit, тощо).

2. Розробити модульні тести для функціоналу колекції.

3. Дослідити ступінь покриття модульними тестами вихідного коду колекції, використовуючи, наприклад, засіб AxoCover.

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

Код:

```
public class DequeTests
{
    [Fact]
    public void TestPushFront()
    {
        // Arrange
        var deque = new Deque<int>();

        // Act
        deque.PushFront(1);
        deque.PushFront(2);
        deque.PushFront(3);

        // Assert
        Assert.Equal("[3, 2, 1]", deque.ToString());
    }
```

```
    [Fact]
    public void TestPushBack()
    {
        // Arrange
        var deque = new Deque<int>();

        // Act
        deque.PushBack(1);
        deque.PushBack(2);
        deque.PushBack(3);

        // Assert
        Assert.Equal("[1, 2, 3]", deque.ToString());
    }

    [Fact]
    public void TestPopFront()
    {
        // Arrange
        var deque = new Deque<int>();
        deque.PushBack(1);
        deque.PushBack(2);
        deque.PushBack(3);

        // Act
        var popped = deque.PopFront();

        // Assert
        var expectedDeque = new Deque<int>();
        expectedDeque.PushBack(2);
        expectedDeque.PushBack(3);
        Assert.Equal(1, popped);
        Assert.Equal("[2, 3]", deque.ToString());
        Assert.Equal(expectedDeque,deque);
    }

    [Fact]
    public void TestPopBack()
    {
        // Arrange
        var deque = new Deque<int>();
        deque.PushBack(1);
        deque.PushBack(2);
        deque.PushBack(3);

        // Act
        var popped = deque.PopBack();

        // Assert
        Assert.Equal(3, popped);
        Assert.Equal("[1, 2]", deque.ToString());
```

```
    }

    [Fact]
    public void TestPeekFront()
    {
        // Arrange
        var deque = new Deque<int>();
        deque.PushBack(1);
        deque.PushBack(2);

        // Act
        var peeked = deque.PeekFront();

        // Assert
        Assert.Equal(1, peeked);
    }

    [Fact]
    public void TestPeekBack()
    {
        // Arrange
        var deque = new Deque<int>();
        deque.PushBack(1);
        deque.PushBack(2);

        // Act
        var peeked = deque.PeekBack();

        // Assert
        Assert.Equal(2, peeked);
    }

    [Fact]
    public void TestIsEmpty()
    {
        // Arrange
        var emptyDeque = new Deque<int>();
        var nonEmptyDeque = new Deque<int>();
        nonEmptyDeque.PushFront(1);

        // Act & Assert
        Assert.True(emptyDeque.IsEmpty());
        Assert.False(nonEmptyDeque.IsEmpty());
    }

    [Fact]
    public void TestClear()
    {
        // Arrange
        var deque = new Deque<int>();
        deque.PushBack(1);
        deque.PushBack(2);
```

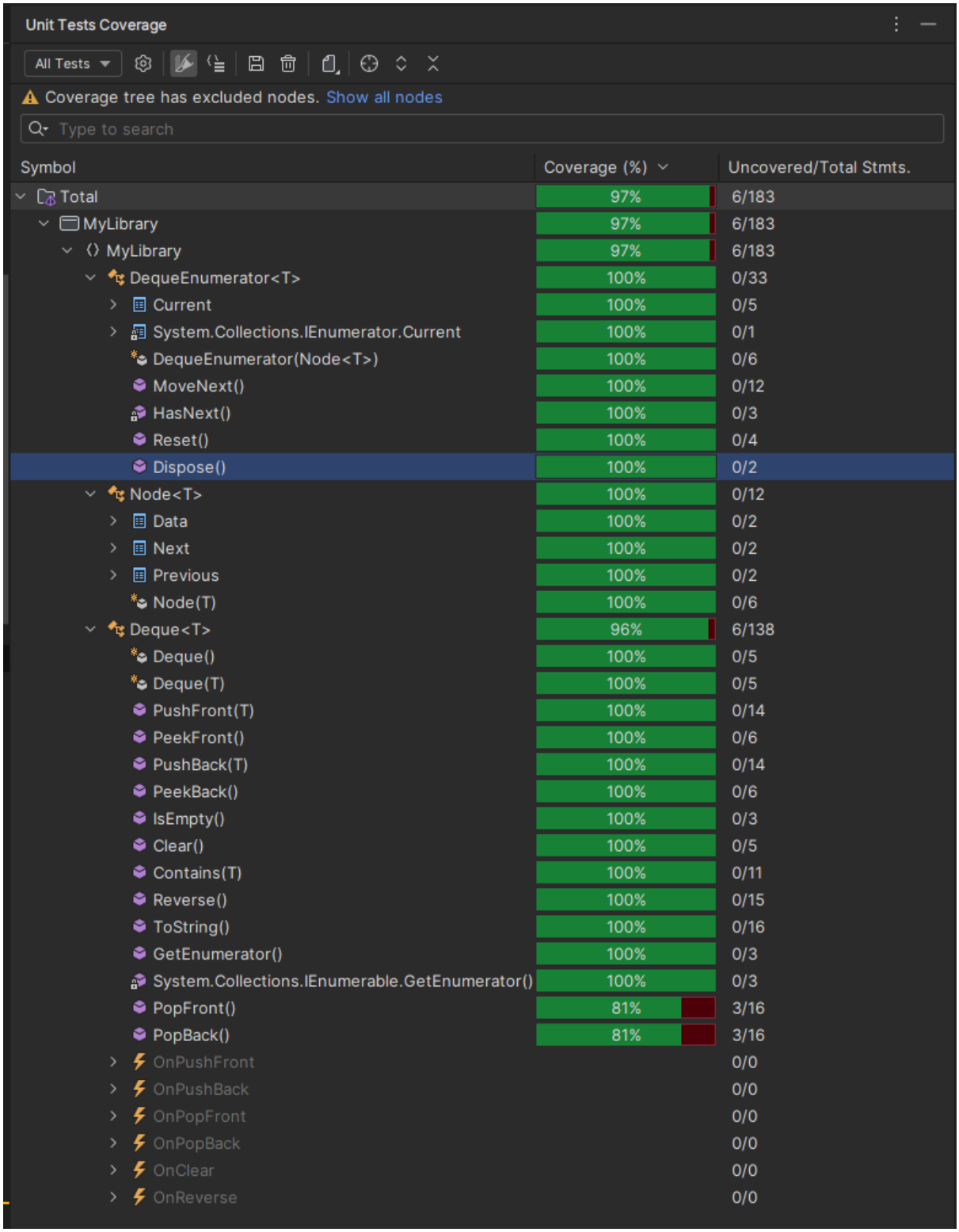
```
        // Act
        deque.Clear();

        // Assert
        Assert.True(deque.IsEmpty());
    }

    [Fact]
    public void TestContains()
    {
        // Arrange
        var deque = new Deque<string>();
        deque.PushBack("apple");
        deque.PushBack("banana");
        deque.PushBack("cherry");

        // Act & Assert
        Assert.True(deque.Contains("banana"));
        Assert.False(deque.Contains("grape"));
    }

    [Fact]
    public void TestReverse()
    {
        // Arrange
        var deque = new Deque<int>();
        deque.PushBack(1);
        deque.PushBack(2);
        deque.PushBack(3);

        // Act
        deque.Reverse();

        // Assert
        Assert.Equal("[3, 2, 1]", deque.ToString());
    }

     [Fact]
    public void TestDequeueEmptyDeque()
    {
        // Arrange
        var deque = new Deque<int>();

        // Act & Assert
        Assert.Throws<InvalidOperationException>(() => deque.PopFront());
        Assert.Throws<InvalidOperationException>(() => deque.PopBack());
    }

    [Fact]
    public void TestPeekEmptyDeque()
    {
```

```
        // Arrange
        var deque = new Deque<int>();

        // Act & Assert
        Assert.Throws<InvalidOperationException>(() => deque.PeekFront());
        Assert.Throws<InvalidOperationException>(() => deque.PeekBack());
    }

    [Fact]
    public void TestReverseEmptyDeque()
    {
        // Arrange
        var deque = new Deque<int>();

        // Act
        deque.Reverse();

        // Assert
        Assert.True(deque.IsEmpty());
    }

    [Fact]
    public void TestReverseSingleItemDeque()
    {
        // Arrange
        var deque = new Deque<int>();
        deque.PushBack(1);

        // Act
        deque.Reverse();

        // Assert
        Assert.Equal("[1]", deque.ToString());
    }

    [Fact]
    public void TestGetEnumerator()
    {
        // Arrange
        var deque = new Deque<int>();
        deque.PushBack(1);
        deque.PushBack(2);
        deque.PushBack(3);

        // Act
        var enumerator = deque.GetEnumerator();

        // Assert
        Assert.True(enumerator.MoveNext());
        Assert.Equal(1, enumerator.Current);
        Assert.True(enumerator.MoveNext());
        Assert.Equal(2, enumerator.Current);
```

```
        Assert.True(enumerator.MoveNext());
        Assert.Equal(3, enumerator.Current);
        Assert.False(enumerator.MoveNext());
    }

    [Fact]
    public void TestEnumeratorReset()
    {
        // Arrange
        var deque = new Deque<int>();
        deque.PushBack(1);
        deque.PushBack(2);

        // Act
        var enumerator = deque.GetEnumerator();
        enumerator.MoveNext();
        enumerator.Reset();

        // Assert
        enumerator.MoveNext();
        Assert.Equal(1, enumerator.Current);
    }

    [Fact]
    public void TestEnumeratorCurrentBeforeEnumeration()
    {
        // Arrange
        var deque = new Deque<int>();

        // Act & Assert
        var enumerator = deque.GetEnumerator();
        Assert.Throws<InvalidOperationException>(() => enumerator.Current);
    }

    [Fact]
    public void TestCurrentFromEnumeratorWithoutMoveNext()
    {
        // Arrange
        var deque = new Deque<int>(1);

        // Act
        var enumerator = deque.GetEnumerator();

        // Assert
        Assert.Throws<InvalidOperationException>(() => enumerator.Current);
    }

    [Fact]
    public void TestCurrentFromEnumeratorAfterMoveNext()
    {
        // Arrange
        var deque = new Deque<int>();
```

```
        deque.PushBack(2);
        deque.PushBack(3);

        // Act
        var enumerator = deque.GetEnumerator();
        enumerator.MoveNext();
        var current = enumerator.Current;

        // Assert
        Assert.Equal(2, current);
    }
}
```

Ступінь покриття модульними тестами отримав за допомогою dotCover:


Unit Tests Coverage
All Tests
Coverage tree has excluded nodes. Show all nodes
Type to search
Symbol
Coverage (%)
Uncovered/Total Stmts.
Total 97% 6/183
MyLibrary 97% 6/183
MyLibrary 97% 6/183
DequeEnumerator<T> 100% 0/33
Current 100% 0/5
System.Collections.IEnumerator.Current 100% 0/1
DequeEnumerator(Node<T>) 100% 0/6
MoveNext() 100% 0/12
HasNext() 100% 0/3
Reset() 100% 0/4
Dispose() 100% 0/2
Node<T> 100% 0/12
Data 100% 0/2
Next 100% 0/2
Previous 100% 0/2
Node(T) 100% 0/6
Deque<T> 96% 6/138
Deque() 100% 0/5
Deque(T) 100% 0/5
PushFront(T) 100% 0/14
PeekFront() 100% 0/6
PushBack(T) 100% 0/14
PeekBack() 100% 0/6
IsEmpty() 100% 0/3
Clear() 100% 0/5
Contains(T) 100% 0/11
Reverse() 100% 0/15
ToString() 100% 0/16
GetEnumerator() 100% 0/3
System.Collections.IEnumerable.GetEnumerator() 100% 0/3
PopFront() 81% 3/16
PopBack() 81% 3/16
OnPushFront 0/0
OnPushBack 0/0
OnPopFront 0/0
OnPopBack 0/0
OnClear 0/0
OnReverse 0/0