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»

“Модульне тестування. Ознайомлення з засобами та практиками

модульного тестування”

Виконав(ла): Маланічев Д., ІС-13

Перевірив(ла): Бардін В.

Київ 2023

Мета лабораторної роботи – навчитися створювати модульні тести для вихідного коду розроблювального програмного забезпечення.

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.

Хід роботи

Варіант:

9	Динамічний масив з довільним діапазоном індексу	Див. List<T>	Збереження даних за допомогою вектору

Лістинг коду:

DynamicArray.cs

```
using MyCollection;

namespace DynamicArray.Tests;

public class DynamicArrayTests
{
    [Fact]
    public void IndexerReturnsItem_IfItemExists()
    {
        // Arrange
        const int expectedItem1 = 1;
        const int expectedItem2 = 2;
        const int expectedItem3 = 3;
        var dynArray = new DynamicArray<int> { expectedItem1, expectedItem2,
expectedItem3 };

        // Act
        var item1 = dynArray[0];
        var item2 = dynArray[1];
        var item3 = dynArray[2];

        // Assert
        Assert.Equal(expectedItem1, item1);
        Assert.Equal(expectedItem2, item2);
        Assert.Equal(expectedItem3, item3);
    }

    [Fact]
    public void IndexerThrowsIndexOutOfRangeException_IfItemDoesntExist()
    {
        // Arrange
        var dynArray = new DynamicArray<int> { 1, 2, 3 };

        // Act & Assert
        Assert.Throws<IndexOutOfRangeException>(() => dynArray[-1]);
        Assert.Throws<IndexOutOfRangeException>(() => dynArray[300]);
    }

    [Fact]
    public void IndexerSetsItem_IfIndexIsValid()
    {
        // Arrange
        const int expectedValue = 100;
        var dynArray = new DynamicArray<int> { 1 };

        // Act
```

```
        dynArray[0] = expectedValue;
        var assignedValue = dynArray[0];

        // Assert
        Assert.Equal(expectedValue, assignedValue);
    }

    [Fact]
    public void SetterThrowsArgumentException_IfItemDoesntExist()
    {
        // Arrange
        var dynArray = new DynamicArray<int> { 1, 2, 3 };

        // Act & Assert
        Assert.Throws<ArgumentException>(() => dynArray[10] = 5);
        Assert.Throws<ArgumentException>(() => dynArray[-10] = 5);
    }

    [Fact]
    public void ConstructorInitializeDynamicArray_IfCapacityIsValid()
    {
        // Arrange
        const int capacity = 5;

        // Act
        var dynArray = new DynamicArray<int>(capacity);
        int factCapacity = 0;
        dynArray.DynamicArrayResized += (sender, args) => factCapacity =
args.OldCapacity;
        dynArray.AddRange(new[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 });

        // Assert
        Assert.Equal(capacity, factCapacity);
    }

    [Fact]
    public void
ConstructorThrowsArgumentOutOfRangeException_IfCapacityIsNegative()
    {
        // Arrange & Act & Assert
        Assert.Throws<ArgumentOutOfRangeException>(() => new
DynamicArray<int>(-100));
    }

    [Fact]
    public void ConstructorInitializeDynamicArray_IfIEnumerablePassed()
    {
        // Arrange
        var nums = new[] { 1, 2, 3, 4, 5 };

        // Act
        var dynArray = new DynamicArray<int>(nums);

        // Assert
        Assert.Equal(nums.Length, dynArray.Count);
        Assert.Equal(nums[0], dynArray[0]);
        Assert.Equal(nums[1], dynArray[1]);
        Assert.Equal(nums[2], dynArray[2]);
        Assert.Equal(nums[3], dynArray[3]);
        Assert.Equal(nums[4], dynArray[4]);
    }

    [Fact]
    public void
```

```
ConstructorThrowsArgumentNullException_IfIEnumerablePassedIsNull()
    {
        // Arrange & Act & Assert
        Assert.Throws<ArgumentNullException>(() => new
DynamicArray<int>(null!));
    }

    [Fact]
    public void CountReturnValidNumOfElements()
    {
        // Arrange
        var nums = new[] { 1, 2, 3, 4, 5 };

        // Act
        var dynArray = new DynamicArray<int>(nums) { 3 };
        dynArray.Add(10);
        dynArray.AddRange(new[] { 1, 2, 3 });

        // Assert
        Assert.Equal(10, dynArray.Count);
    }

    [Fact]
    public void ClearClearsDynamicArray()
    {
        // Arrange
        var dynArray = new DynamicArray<int>(new[] { 1, 2, 3 });

        // Act
        dynArray.Clear();

        // Assert
        Assert.Empty(dynArray);
        Assert.Throws<IndexOutOfRangeException>(() => dynArray[0]);
    }

    [Fact]
    public void RemoveAt_RemovesElementGivenAtGivenIndex_IfIndexIsValid()
    {
        // Arrange
        var dynArray = new DynamicArray<int>(new[] { 1, 2, 3 });

        // Act
        dynArray.RemoveAt(2);

        // Assert
        Assert.Equal(2, dynArray.Count);
        Assert.Throws<IndexOutOfRangeException>(() => dynArray[2]);
    }

    [Fact]
    public void RemoveAt_ThrowsIndexOutOfRangeException_IfIndexIsntValid()
    {
        // Arrange
        var dynArray = new DynamicArray<int>(new[] { 1, 2, 3 });

        // Act & Assert
        Assert.Throws<IndexOutOfRangeException>(() => dynArray.RemoveAt(20));
    }

    [Fact]
    public void Insert_ThrowsIndexOutOfRangeException_IfIndexIsntValid()
    {
        // Arrange
```

```
        var dynArray = new DynamicArray<int>(new[] { 1, 2, 3 });

        // Act & Assert
        Assert.Throws<IndexOutOfRangeException>(() => dynArray.Insert(-20,
100));
        Assert.Throws<IndexOutOfRangeException>(() => dynArray.Insert(20,
100));
    }

    [Theory]
    [InlineData(new[] { 1, 2, 3 }, 1)]
    [InlineData(new[] { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
}, 1)]
    [InlineData(new[] { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
}, 16)]
    public void Insert_InsertElementAtGivenIndex_IfIndexIsValid(int[] nums,
int index)
    {
        // Arrange
        var dynArray = new DynamicArray<int>(nums);
        const int numToInsert = 100;

        // Act
        dynArray.Insert(index, numToInsert);

        // Assert
        Assert.Equal(nums.Length + 1, dynArray.Count);
        Assert.Equal(numToInsert, dynArray[index]);
    }

    [Theory]
    [InlineData(new[] { 1, 2, 3 })]
    [InlineData(new[] { 1, 2, 3, 2 })]
    public void IndexOf_ReturnsIndexOfGivenElement(int[] nums)
    {
        // Arrange
        var dynArray = new DynamicArray<int>(nums);
        const int expectedIndex = 1;

        // Act
        var index = dynArray.IndexOf(2);

        // Assert
        Assert.Equal(expectedIndex, index);
    }

    [Fact]
    public void Remove_RemovesGivenElement_IfExists()
    {
        // Arrange
        var dynArray = new DynamicArray<int>(new[] { 1, 2, 3 });

        // Act
        var isRemoved = dynArray.Remove(2);

        // Assert
        Assert.True(isRemoved);
        Assert.DoesNotContain(2, dynArray);
    }

    [Fact]
    public void Remove_ReturnsFalse_IfDoesntExist()
    {
        // Arrange
```

```
        var dynArray = new DynamicArray<int>(new[] { 1, 2, 3 });

        // Act
        var isRemoved = dynArray.Remove(200);

        // Assert
        Assert.False(isRemoved);
        Assert.DoesNotContain(200, dynArray);
    }

    [Fact]
    public void IsReadonly_IsAlwaysFalse()
    {
        // Arrange
        var dynArray = new DynamicArray<int> { 1, 2, 3};

        // Act
        var isReadOnly = dynArray.IsReadOnly;

        // Assert
        Assert.False(isReadOnly);
    }

    [Fact]
    public void CopyTo_CopiesToGivenArray_IfCapacityIsBigEnough()
    {
        // Arrange
        var dynArray = new DynamicArray<int>(new[] { 1, 2, 3 });
        var destArray = new int[3];

        // Act
        dynArray.CopyTo(destArray, 0);

        // Assert
        Assert.Equal(3, destArray.Length);
        Assert.Equal(dynArray[0], destArray[0]);
        Assert.Equal(dynArray[1], destArray[1]);
        Assert.Equal(dynArray[2], destArray[2]);
    }

    [Fact]
    public void CopyTo_ThrowsArgumentException_IfCapacityIsBigEnough()
    {
        // Arrange
        var dynArray = new DynamicArray<int>(new[] { 1, 2, 3 });
        var destArray = new int[3];

        // Act & Assert
        Assert.Throws<ArgumentException>(() => dynArray.CopyTo(destArray,
3));
    }

    [Fact]
    public void
AddRange_ThrowsArgumentNullException_IfPassedIEnumerableIsNull()
    {
        // Arrange
        var dynArray = new DynamicArray<int>();

        // Act & Assert
        Assert.Throws<ArgumentNullException>(() => dynArray.AddRange(null!));
    }

    [Fact]
```

```
    public void Contains_ReturnTrue_IfPassedItemExists()
    {
        // Arrange
        var dynArray = new DynamicArray<int> { 1, 2, 3};

        // Act
        var isContains = dynArray.Contains(2);

        // Assert
        Assert.True(isContains);
    }

    [Fact]
    public void Contains_ReturnFalse_IfPassedItemDoesntExist()
    {
        // Arrange
        var dynArray = new DynamicArray<int> { 1, 2, 3};

        // Act
        var isContains = dynArray.Contains(200);

        // Assert
        Assert.False(isContains);
    }

    [Fact]
    public void ItemAdded_EventRaised()
    {
        // Arrange
        var dynArray = new DynamicArray<int>();
        var count = 0;
        dynArray.ItemAdded += (sender, e) => count++;

        // Act
        dynArray.Add(2);

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

    [Fact]
    public void ItemRemoved_EventRaised()
    {
        // Arrange
        var dynArray = new DynamicArray<int> { 2 };
        var count = 0;
        dynArray.ItemRemoved += (sender, e) => count++;

        // Act
        dynArray.Remove(2);

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

    [Fact]
    public void ArrayResized_EventRaised()
    {
        // Arrange
        var dynArray = new DynamicArray<int>();
        var count = 0;
        dynArray.DynamicArrayResized += (sender, e) => count++;

        // Act
```

```
        dynArray.AddRange(new[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17 });

        // Assert
        Assert.Equal(1, count);
    }
}
```

Висновок: в ході виконання першої частини даної лабораторної роботи було створено власну реалізацію узагальненої колекції, а саме списку. При цьому було використано інтерфейси IList<T> та IList, а також було продемонстровано використання подій (event) та обробку виключних ситуацій. Результат роботи продемонстровано в консольному застосунку, який використовує всі методи створеної колекції.

У даній частині для вже створеної колекції були написані тести, які покривають всю логіку колекції. Виконуючи лабораторну роботу, було вивчено поняття модульне тестування, TDD, принцип “Triple A”, Mock & Stub.

Посилання на GitHub: https://github.com/DenysMalanichev/DynamicArray-.NET-Lab1-