Національний технічний університет України «КПІ»
Факультет інформатики та обчислювальної техніки
Кафедра Інформаційних систем та технологій

Лабораторна робота №2

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

на тему: «Модульне тестування. Ознайомлення з засобами та практиками модульного тестування»

Виконала:
студентка гр. ІО-15
Григор’єв К. С.
Викладач:
Бардін В.

2023 рік

Мета: 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.

Посилання на код GitHub: https://github.com/Grugoriev180/DotNetLabs

Код тестів:

DynamicArrayTests.cs

```
using Xunit;

namespace DotNetLab1.Tests
{
    public class DynamicArrayTests
    {

        #region ConstructorTests



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

            // 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]);
        }

        #endregion
```

```
        #region IndexerTesting

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

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

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

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

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

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

            // Act
            dynamicArray[0] = expectedValue;
            var assignedValue = dynamicArray[0];

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

        #endregion

        #region AddTests

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

            // Act
            dynamicArray.Add(2);
```

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

    [Fact]
    public void Add_NewElement_CountIncrements()
    {
        // Arrange
        var rdynamicArray = new DynamicArray<int>();
        int defaultCount = rdynamicArray.Count;

        // Act
        rdynamicArray.Add(0);

        // Assert
        Assert.Equal(1, rdynamicArray.Count - defaultCount);
    }

    [Fact]
    public void Add_NullElement_ThrowsArgumentNullException()
    {
        // Arrange
        var dynamicArray = new DynamicArray<string>();

        // Act & Assert
        Assert.Throws<ArgumentNullException>(() => dynamicArray.Add(null));
    }

    #endregion

    #region ContainsTests

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

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

        // Assert
        Assert.True(isContains);
    }

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

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

        // Assert
        Assert.False(isContains);
    }


    #endregion

    #region CopyToTests

    [Fact]
```

```
        public void CopyTo_CorrectArrayAndIndex_SuccessfullCopying()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int>{ 1, 2, 3 };
            var destArray = new int[3];

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

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

        [Fact]
        public void CopyTo_ThrowsArgumentException()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int>{ 1, 2, 3 };
            var destArray = new int[4];

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

        [Fact]
        public void CopyTo_ThrowsArgumentNullException()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int>();
            int[] arrayCopyTo = null;
            int indexCopyTo = 0;

            // Act & Assert
            Assert.Throws<ArgumentNullException>(()
=>dynamicArray.CopyTo(arrayCopyTo, indexCopyTo));

        }

        #endregion

        #region IndexOfTests

        [Fact]
        public void IndexOf_NullElement_ThrowsArgumentNullException()
        {
            // Arrange
            var dynamicArray = new DynamicArray<string>();

            // Act & Assert
            Assert.Throws<ArgumentNullException>(() =>
dynamicArray.IndexOf(null));
        }

        [Fact]
        public void IndexOf_ElementDoesNotExist_ReturnsDefaultIndex()
        {
            // Arrange
            var collection = new DynamicArray<int>() { 1, 2, 3 };
            int element = 4;
            int defaultIndex = -1;
```

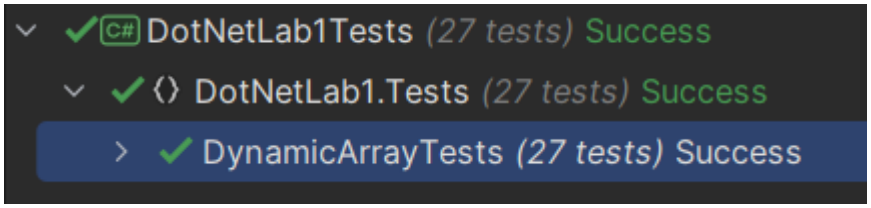
```
            // Act
            int actualIndex = collection.IndexOf(element);

            // Assert
            Assert.Equal(defaultIndex, actualIndex);
        }

        [Fact]
        public void IndexOf_ElementExists_ReturnsElementsIndex()
        {
            // Arrange
            var collection = new DynamicArray<int>() { 1, 2, 3 };
            int element = 2;
            int expectedIndex = 1;

            // Act
            int actualIndex = collection.IndexOf(element);

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

        #endregion

        #region InsertTests

        [Fact]
        public void Insert_ProperElement_SuccessfullInsertion()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int>() { 1, 2, 4, 5 };
            int elementToInsert = 6;
            int indexToInsert = 1;

            int defaultCount = dynamicArray.Count;

            // Act
            dynamicArray.Insert(indexToInsert, elementToInsert);

            // Assert
            Assert.Equal(elementToInsert, dynamicArray[indexToInsert]);
            Assert.Equal(1, dynamicArray.Count - defaultCount);
        }

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

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

        [Fact]
        public void Insert_NullElement_ThrowsArgumentNullException()
        {
            // Arrange
            var dynamicArray = new DynamicArray<string>();
```

```
            // Act & Assert
            Assert.Throws<ArgumentNullException>(() => dynamicArray.Insert(0,
null));
        }

        #endregion

        #region RemoveTests

        [Fact]
        public void Remove_Element_EventRaised()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int> { 1, 2, 3 };
            var count = 0;
            dynamicArray.ItemRemoved += (sender, e) => count++;

            // Act
            dynamicArray.Remove(2);

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

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

            // Act
            var isRemoved = dynamicArray.Remove(12);

            // Assert
            Assert.False(isRemoved);
            Assert.DoesNotContain(12, dynamicArray);
        }

        [Fact]
        public void Remove_ElementDoesNotExist_ReturnsFalse()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int>() { 1, 2, 3 };
            var elementToRemove = 4;

            // Act
            var result = dynamicArray.Remove(elementToRemove);

            // Assert
            Assert.False(result);
        }


        #endregion

        #region RemoveAtTests

        [Fact]
        public void RemoveAt_IndexPassed_SuccessfullRemoving()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int>() { 1, 2, 3 };
            var indexToRemove = 1;
            var elementToRemove = dynamicArray[indexToRemove];
```

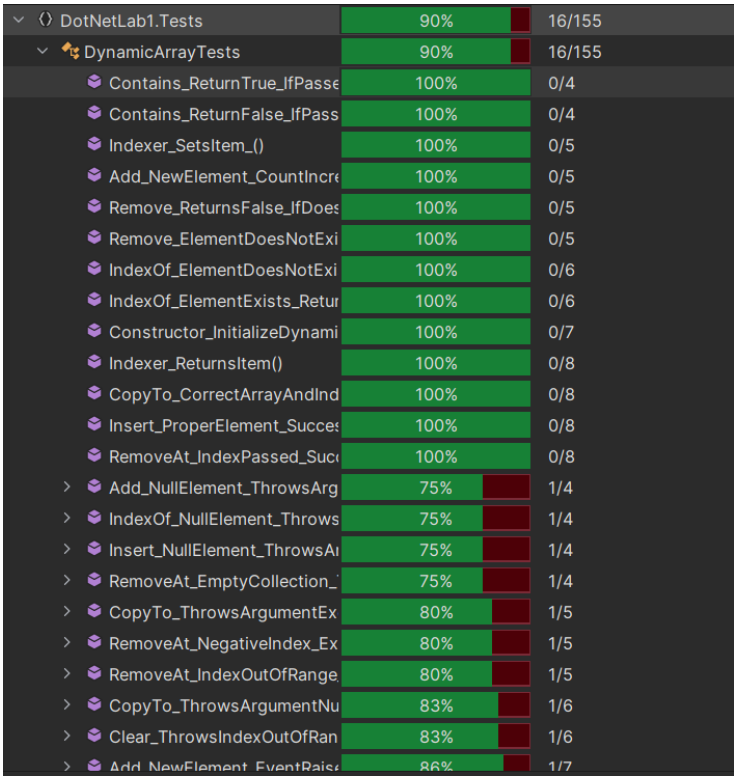
```
            var defaultCount = dynamicArray.Count;

            // Act
            dynamicArray.RemoveAt(indexToRemove);

            // Assert
            Assert.Equal(1, defaultCount - dynamicArray.Count);
            Assert.DoesNotContain(elementToRemove, dynamicArray);
        }

        [Fact]
        public void RemoveAt_NegativeIndex_ExceptionThrown()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int>() { 1, 2, 3 };
            var indexToRemove = -1;

            // Act and Assert
            Assert.Throws<ArgumentOutOfRangeException>(() =>
dynamicArray.RemoveAt(indexToRemove));
        }

        [Fact]
        public void RemoveAt_IndexOutOfRange_ExceptionThrown()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int>() { 1, 2, 3 };
            var indexToRemove = 4;

            // Act and Assert
            Assert.Throws<ArgumentOutOfRangeException>(() =>
dynamicArray.RemoveAt(indexToRemove));
        }

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

            // Act & Assert
            Assert.Throws<ArgumentOutOfRangeException>(() =>
dynamicArray.RemoveAt(0));
        }

        #endregion

        #region ClearTests

        [Fact]
        public void Clear_EventRaised()
        {
            // Arrange
            var dynamicArray = new DynamicArray<int>() { 1, 2, 3 };
            var eventRaised = false;
            dynamicArray.ArrayCleared += (sender, e) => eventRaised = true;

            // Act
            dynamicArray.Clear();

            // Assert
            Assert.True(eventRaised);
        }
```

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

            // Act
            dynamicArray.Clear();

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

        #endregion



    }
}
```

Результат виконання:



Покриття тестів:



Висновок:

В ході виконання цієї лабораторної роботи було досягнуто важливого практичного результату, пов'язаного з розробкою модульних тестів для вихідного коду розроблювального програмного забезпечення. Основною метою лабораторної роботи було навчитися створювати модульні тести.

Під час створення модульних тестів були перевірені різні аспекти функціоналу колекції, такі як конструктор, індексатори, додавання, видалення, перевірка наявності елементів тощо. Це дозволило переконатися в коректності роботи колекції та її методів. Деякі помилки були виправлені після виконання юніт тестів.

Крім того, для оцінки ступеня покриття модульними тестами вихідного коду колекції був використаний інструмент dotCover. Завдяки цьому інструменту було можливо визначити, наскільки велика частина коду була покрита модульними тестами.