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”

Варіант 9

Виконав студент ІП-12 Єльчанінов Артем Юрійович

Перевірив(ла) Бардін Владислав

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.

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

Виконання:

UnitTests.cs:

```
using MyListLibrary;

namespace Lab2 {
    public class UnitTests {

        [Fact]
        public void Property_IsReadOnly_False() {

            //Arange
            MyList<int> myList = new MyList<int>(0);

            //Act
            bool myListIsReadOnly = myList.IsReadOnly;

            //Assert
            Assert.False(myListIsReadOnly);
        }

        [Fact]
        public void Constructor_With_Negative_Capacity_ThrowsException() {

            //Arange
            int capacity = -1;

            //Act & Assert
            Assert.Throws<ArgumentOutOfRangeException>(() => new MyList<int>(capacity));
        }

        [Fact]
        public void Constructor_With_Zero_Capacity_InitializesMyList() {

            //Arange
            int capacity = 0;

            //Act
            MyList<int> myList = new MyList<int>(capacity);
```

```
            //Assert
            Assert.Equal(capacity, myList.GetCapacity());
            Assert.Empty(myList);
        }

        [Fact]
        public void Construcor_With_Positive_Capacity_InitializesMyList() {

            //Arange
            int capacity = 5;

            //Act
            MyList<int> myList = new MyList<int>(capacity);

            //Assert
            Assert.Equal(capacity, myList.GetCapacity());
            Assert.Equal(0, myList.Count);
        }

        [Fact]
        public void Indexer_Returns_Item_IfItemExiscts() {

            //Arange
            int expectedItem1 = 10;
            int expectedItem2 = 20;
            int expectedItem3 = 30;
            MyList<int> myList = new MyList<int> { expectedItem1, expectedItem2,
expectedItem3 };

            //Act
            int item1 = myList[0];
            int item2 = myList[1];
            int item3 = myList[2];

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

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

    //Arange
    int expectedItem1 = 10;
    MyList<int> myList = new MyList<int> { 500 };

    //Act
    myList[0] = expectedItem1;
    int item1 = myList[0];

    //Assert
    Assert.Equal(expectedItem1, item1);
}

[Fact]
public void Indexer_Sets_InvalidIndex_ThrowsArgumentOutOfRangeException() {
    // Arrange
    MyList<int> customList = new MyList<int> { 1 };

    // Act & Assert
    Assert.Throws<IndexOutOfRangeException>(() => customList[-10] = 2);
    Assert.Throws<IndexOutOfRangeException>(() => customList[10] = 2);
}

[Fact]
public void Add_NewItem_ToFullList_RisezesAndContainsItem() {

    //Arange
    MyList<int> newList = new MyList<int> { 1 };

    //Act
    newList.Add(2);

    //Assert
    Assert.Equal(2, newList.Count);
    Assert.Contains(2, newList);
}
```

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

    //Arange
    MyList<int> newList = new MyList<int>();

    //Act
    newList.Add(5);

    //Assert
    Assert.Equal(1, newList.Count);
    Assert.Contains(5, newList);
}

[Fact]
public void Clear_EmptyList_ClearsList() {

    //Arange
    MyList<int> myList = new MyList<int>();

    //Act
    myList.Clear();

    //Assert
    Assert.Empty(myList);
}

[Fact]
public void Clear_NonEmptyList_ClearsList() {

    //Arange
    MyList<int> myList = new MyList<int> { 1, 2, 3};

    //Act
    myList.Clear();

    //Assert
    Assert.Empty(myList);
}
```

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

    //Arange
    MyList<int> myList = new MyList<int>();

    //Act & Assert
    Assert.False(myList.Contains(1));

}

[Fact]
public void Contains_NonEmptyList_ReturnsTrue() {

    //Arange
    MyList<int> myList = new MyList<int> { 1, 3, 4 };

    //Act & Assert
    Assert.True(myList.Contains(1));
}

[Fact]
public void CopyTo_NullList_ThrowsArgumentNullException() {

    //Arange
    MyList<int> list1 = new MyList<int> {1, 2, 3};
    int[]? list2 = null;

    //Act & Assert
    Assert.Throws<ArgumentNullException>(() => list1.CopyTo(list2!, 0));
}

[Fact]
public void CopyTo_InvalidIndex_ThrowsArgumentException() {

    //Arange
    MyList<int> list1 = new MyList<int> { 1, 2, 3 };
    int[] list2 = new int[list1.Count];
```

```
    //Act & Assert
    Assert.Throws<ArgumentException>(() =>  list1.CopyTo(list2, 5));
}

[Fact]
public void CopyTo_SmallDestArray_ThrowsArgumentException() {

    //Arange
    MyList<int> list1 = new MyList<int> { 1, 2, 3 };
    int[] list2 = new int[1];

    //Act & Assert
    Assert.Throws<ArgumentException>(() => list1.CopyTo(list2, 0));
}

[Fact]
public void CopyTo_ValidArraysAndParameters_CopiesItemsToDestArray() {

    //Arange
    MyList<int> list1 = new MyList<int> { 1, 2, 3 };
    int[] list2 = new int[list1.Count];

    //Act
    list1.CopyTo(list2, 0);

    //Assert
    int[] expectedList = new int[] {1, 2, 3};
    Assert.Equal(expectedList, list2);
}

[Fact]
public void IndexOf_InvalidItem_ReturnsMinusOne() {

    //Arange
    MyList<int> list1 = new MyList<int> { 1, 2, 3 };

    //Act & Assert
    Assert.Equal(-1, list1.IndexOf(500));
```

```
}

[Fact]
public void IndexOf_ValidItem_ReturnsTrueIndex() {

    //Arange
    MyList<int> list1 = new MyList<int> { 1, 2, 3 };

    //Act & Assert
    Assert.Equal(1, list1.IndexOf(2));
}

[Fact]
public void Insert_InvalidIndex_ThrowsArgumentOutOfRangeException() {

    //Arange
    MyList<int> list = new MyList<int> { 1, 2, 3 };

    //Act & Assert
    Assert.Throws<ArgumentOutOfRangeException>(() => list.Insert(-5, 5));
    Assert.Throws<ArgumentOutOfRangeException>(() => list.Insert(5, 5));
}

[Fact]
public void Insert_ValidParameters_WithResize_ResizesAndInsertItem() {

    //Arange
    MyList<int> list = new MyList<int>(3) { 1, 2, 3 };

    //Act
    list.Insert(3, 4);

    //Assert
    Assert.Equal(6, list.GetCapacity());
    Assert.Equal(4, list[3]);
}

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

```
    //Arange
    MyList<int> list = new MyList<int>(2) { 10 };

    //Act
    list.Insert(1, 2);

    //Assert
    Assert.Equal(2, list.Count);
    Assert.Equal(2, list[1]);
}

[Fact]
public void Remove_FromEmptyList_ThrowsInvalidOperationException() {

    //Arange
    MyList<int> list = new MyList<int>(0);

    //Act & Assert
    Assert.Throws<InvalidOperationException>(() => list.Remove(5));
}

[Fact]
public void Remove_From_NormalList_RemovesItem() {

    // Arange
    MyList<int> list = new MyList<int>();
    list.Add(1);
    list.Add(2);

    // Act
    bool removed = list.Remove(1);

    // Assert
    Assert.True(removed);
    Assert.False(list.Contains(1));
    Assert.Equal(1, list.Count);
}
```

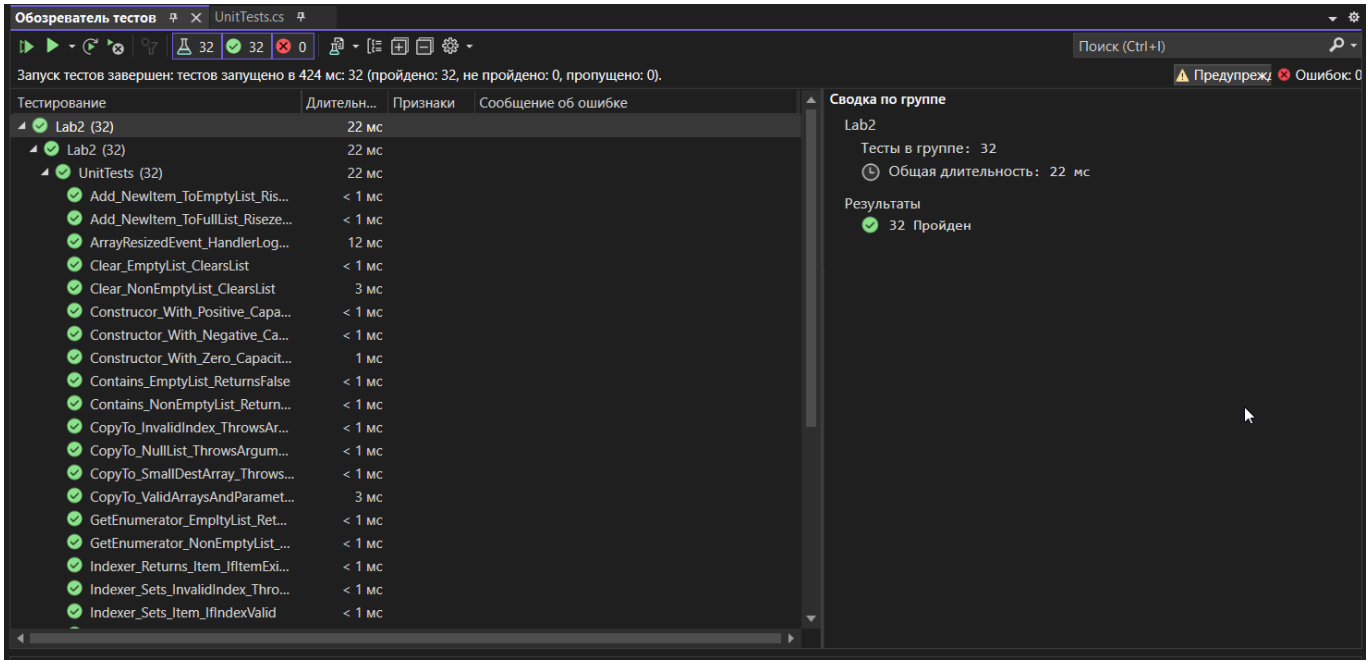
```
[Fact]
public void RemoveAt_InvalidIndex_ThrowsArgumentOutOfRangeException() {

    //Arange
    MyList<int> list = new MyList<int>(1) { 1 };

    //Act & Assert
    Assert.Throws<ArgumentOutOfRangeException>(() => list.RemoveAt(-1));
    Assert.Throws<ArgumentOutOfRangeException>(() => list.RemoveAt(2));

}

[Fact]
public void RemoveAt_ValidIndex_RemovesItem() {

    //Arange
    MyList<int> list = new MyList<int>();
    list.Add(1);
    list.Add(2);
    list.Add(3);

    // Act
    list.RemoveAt(1);

    // Assert
    Assert.Equal(2, list.Count);
    Assert.False(list.Contains(2));
}

[Fact]
public void GetEnumerator_EmpltyList_ReturnsEmptyEnumerator() {

    //Arange
    MyList<int> list = new MyList<int>() ;

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

    //Assert
```

```
    Assert.False(enumerator.MoveNext());
}

[Fact]
public void GetEnumerator_NonEmptyList_ReturnsCorrectEnumerator() {

    // Arange
    MyList<int> list1 = new MyList<int>();
    list1.Add(1);
    list1.Add(2);
    list1.Add(3);

    // Act
    var enumerator = list1.GetEnumerator();

    // Assert
    List<int> list2 = new List<int>();
    int i = 0;
    while (i < list1.Count) {
        list2.Add(enumerator.Current);
        enumerator.MoveNext();
        i++;
    }

    Assert.Equal(new[] { 1, 2, 3 }, list2);
}

[Fact]
public void ArrayResizedEvent_HandlerLogsCorrectly() {

    // Arange
    MyList<int> list = new MyList<int>(1) { 1 };
    list.ArrayResized += MyEventHandlers.ListResizedEventHandler!;

    var consoleOutput = new System.IO.StringWriter();
    Console.SetOut(consoleOutput);

    //Act
    list.Insert(0, 1);
```

```
    //Assert
    var expectedOutput = "\nEvent: change size; OldCapacity: 1, NewCapacity: 2\n";
    Assert.Contains(expectedOutput, consoleOutput.ToString());
}

[Fact]
public void ItemAddedEvent_HandlerLogsCorrectly() {

    // Arange
    MyList<int> list = new MyList<int>();
    list.ItemAdded += MyEventHandlers.ListItemChangesEventHandler!;

    var consoleOutput = new System.IO.StringWriter();
    Console.SetOut(consoleOutput);

    // Act
    list.Add(1);

    // Assert
    var expectedOutput = "\nEvent: ItemAdded; Item: 1; Index: 0\n";
    Assert.Contains(expectedOutput, consoleOutput.ToString());
}

[Fact]
public void ItemRemovedEvent_HandlerLogsCorrectly() {

    // Arange
    MyList<int> list = new MyList<int>() { 1 };
    list.ItemRemoved += MyEventHandlers.ListItemChangesEventHandler!;

    var consoleOutput = new System.IO.StringWriter();
    Console.SetOut(consoleOutput);

    // Act
    list.Remove(1);

    // Assert
    var expectedOutput = "\nEvent: ItemRemoved; Item: 1; Index: 0\n";
```

```
            Assert.Contains(expectedOutput, consoleOutput.ToString());
        }

        [Fact]
        public void MyListEnumerator_ResetTest_WorksNormal() {

            //Arange
            MyList<int> list1 = new MyList<int>();
            list1.Add(1);
            list1.Add(2);
            list1.Add(3);

            // Act
            var enumerator = list1.GetEnumerator();
            enumerator.MoveNext();
            enumerator.Reset();

            //Assert
            Assert.Equal(list1[0], enumerator.Current);
        }
    }
}
```

Приклад виконання:



Посилання на код:

[Web_.Net_Labs/Lab2 at master · LesterGaben/Web_.Net_Labs (github.com)]