Міністерство освіти і науки України КПІ ім. Ігоря Сікорського

Факультет Інформатики та Обчислювальної Технiки

**ЗВІТ**

**до лабораторної роботи №2 з модуля**

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

|  |  |
| --- | --- |
| Перевірив:  Викладач кафедри ІСТ  ФІОТ  Бардін В. | Виконав:  Поліщук Владислав  гр. ІК-13 |

2023

**Модульне тестування.**

**Ознайомлення з засобами та практиками модульного тестування**

**Мета лабораторної роботи** – навчитися створювати модульні тести для

вихідного коду розроблювального програмного забезпечення.

**Завдання:**

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.

Код програми

public class MyDeqPrimaryTests

{

[Fact]

public void EnqueueItemAtStart\_ShouldAddItem()

{

// Arrange

var myDeq = new MyDeq<int>();

// Act

myDeq.EnqueueItemAtStart(1);

myDeq.EnqueueItemAtStart(1);

// Assert

Assert.Equal(2, myDeq.Count);

}

[Theory]

[ClassData(typeof(MyDeqTestData))]

public void EnqueueItemAtStart\_WhenSizeMoreDeqCapacity\_ShouldResizeDeq<T>(MyDeq<T> values)

{

// Arrange

var myDeq = new MyDeq<T>();

// Act

foreach (var value in values)

{

myDeq.EnqueueItemAtStart(value);

}

// Assert

Assert.Equal(values.Count, myDeq.Count);

}

[Fact]

public void EnqueueItemAtEnd\_ShouldAddItem()

{

// Arrange

var myDeq = new MyDeq<int>();

// Act

myDeq.EnqueueItemAtEnd(1);

myDeq.EnqueueItemAtEnd(1);

// Assert

Assert.Equal(2, myDeq.Count);

}

[Theory]

[ClassData(typeof(MyDeqTestData))]

public void EnqueueItemAtEnd\_WhenSizeMoreDeqCapacity\_ShouldResizeDeq<T>(MyDeq<T> values)

{

// Arrange

var myDeq = new MyDeq<T>();

// Act

foreach (var value in values)

{

myDeq.EnqueueItemAtEnd(value);

}

// Assert

Assert.Equal(values.Count, myDeq.Count);

}

[Fact]

public void DequeueItemFromStart\_WhenDeqIsEmpty\_ShouldThrow()

{

var myDeq = new MyDeq<int>();

var act = () =>

{

myDeq.DequeueItemFromStart();

};

Assert.Throws<InvalidOperationException>(act);

}

[Theory]

[ClassData(typeof (MyDeqTestData))]

public void DequeueItemFromStart\_AllItem\_ShouldBeEmpty<T>(MyDeq<T> values)

{

var myDeq = new MyDeq<T>();

foreach (var item in values)

{

myDeq.EnqueueItemAtEnd(item);

}

for (int i = 0; i < values.Count; i++)

{

\_ = myDeq.DequeueItemFromStart();

}

Assert.Empty(myDeq);

}

[Fact]

public void DequeueItemFromEnd\_WhenDeqIsEmpty\_ShouldThrow()

{

var myDeq = new MyDeq<int>();

var act = () =>

{

myDeq.DequeueItemFromEnd();

};

Assert.Throws<InvalidOperationException>(act);

}

[Theory]

[ClassData(typeof(MyDeqTestData))]

public void DequeueItemFromEnd\_AllItem\_ShouldBeEmpty<T>(MyDeq<T> values)

{

var myDeq = new MyDeq<T>();

foreach (var item in values)

{

myDeq.EnqueueItemAtEnd(item);

}

for (int i = 0; i < values.Count; i++)

{

\_ = myDeq.DequeueItemFromEnd();

}

Assert.Empty(myDeq);

}

[Fact]

public void PeekItemFromEnd\_WhenDeqIsEmpty\_ShouldThrow()

{

var myDeq = new MyDeq<int>();

var act = () =>

{

myDeq.PeekItemFromEnd();

};

Assert.Throws<InvalidOperationException>(act);

}

[Fact]

public void PeekItemFromEnd\_ShouldReturnItem()

{

var myDeq = new MyDeq<int>();

myDeq.EnqueueItemAtEnd(69);

var act = myDeq.PeekItemFromEnd();

Assert.Equal(69, act);

}

[Fact]

public void PeekItemFromStart\_WhenDeqIsEmpty\_ShouldThrow()

{

var myDeq = new MyDeq<int>();

var act = () =>

{

myDeq.PeekItemFromStart();

};

Assert.Throws<InvalidOperationException>(act);

}

[Fact]

public void PeekItemFromStart\_ShouldReturnItem()

{

var myDeq = new MyDeq<int>();

myDeq.EnqueueItemAtEnd(69);

var act = myDeq.PeekItemFromStart();

Assert.Equal(69, act);

}

}

public class MyDeqCollectionTests

{

[Fact]

public void Clear\_WhenDeqIsEmpty\_ShouldNotThrow()

{

var myDeq = new MyDeq<int>();

myDeq.Clear();

Assert.Empty(myDeq);

}

[Theory]

[ClassData(typeof(MyDeqTestData))]

public void Clear\_WhenDeqIsNotEmpty\_ShouldBeEmpty<T>(MyDeq<T> values)

{

values.Clear();

Assert.Empty(values);

}

[Theory]

[ClassData(typeof(MyDeqTestData))]

public void Contains\_WhenValidItem\_ShuouldBeTrue<T>(MyDeq<T> values)

{

var myDeq = new MyDeq<T>(15);

myDeq.EnqueueItemAtEnd(values.PeekItemFromEnd());

var act = myDeq.Contains(values.PeekItemFromEnd());

Assert.True(act);

}

[Fact]

public void Contains\_WhenValidItem2\_ShuouldBeTrue()

{

var myDeq = new MyDeq<int>();

for (int i = 0; i < 8; i++)

{

myDeq.EnqueueItemAtEnd(i + 1);

}

\_ =myDeq.DequeueItemFromStart();

\_ =myDeq.DequeueItemFromStart();

\_ =myDeq.DequeueItemFromStart();

for (int i = 0; i < 5; i++)

{

myDeq.EnqueueItemAtEnd(i + 100);

}

var act = myDeq.Contains(8);

Assert.True(act);

}

[Fact]

public void Contains\_WhenItemIsNull\_ShuouldThrow()

{

var myDeq = new MyDeq<string>();

string item = null;

var act = () =>

{

myDeq.Contains(item);

};

Assert.Throws<ArgumentNullException>(act);

}

[Fact]

public void Contains\_WhenNotContainsItem\_ShuouldBeFalse()

{

var myDeq = new MyDeq<int>(4);

myDeq.EnqueueItemAtEnd(22);

var act = myDeq.Contains(0);

Assert.False(act);

}

[Fact]

public void Contains\_WhenDeqIsEmpty\_ShuouldBeFalse()

{

var myDeq = new MyDeq<int>();

var act = myDeq.Contains(0);

Assert.False(act);

}

[Theory]

[ClassData(typeof(MyDeqTestData))]

public void GetEnumerator\_ShouldBeEqual<T>(MyDeq<T> values)

{

var arr = new T[values.Count];

values.CopyTo(arr, 0);

var enumerator = values.GetEnumerator();

int index = 0;

while (enumerator.MoveNext())

{

Assert.Equal(arr[index], enumerator.Current);

index++;

}

enumerator.Reset();

}

[Fact]

public void Constructor\_WhenInvalidCapacity\_ShouldThrow()

{

var act = () => {

var myDeq = new MyDeq<int>(-1);

};

Assert.Throws<ArgumentOutOfRangeException>(act);

}

Результати



