МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 2
З дисципліни: *"Моделювання та аналіз програмного забезпечення"*
На тему: *"Linq, List and Dictionary"*

Лектор:
доц. каф. ПЗ
Сердюк П.В.

Виконав:
ст. гр. ПЗ-22
Чаус Олег

Прийняв:
викл. каф. ПЗ
Микуляк А. В.

« ____ » _____ 2023 р.

∑= ____ _____.

Львів – 2023

Тема роботи: Linq, List and Dictionary.

Мета роботи: Навчитися працювати з масивами та структурами List, Dictionary. Засвоїти технологію LINQ.

ТЕОРЕТИЧНІ ВІДОМОСТІ

LINQ (англ. Language Integrated Query - запити, інтегровані в мову) - компонент Microsoft .NET Framework, який додає нативні можливості виконання запитів даних до мов, що входять у .NET. Хоча порти існують для PHP (PHPLinq), JavaScript (linq.js), TypeScript (linq.ts), і ActionScript (ActionLinq), - жоден з них не є абсолютно еквівалентним LINQ в C# (де LINQ - не просто додаткова бібліотека, а частина мови).

LINQ розширює можливості мови, додаючи до неї вирази запитів, що є схожими на твердження SQL та можуть бути використані для зручного отримання та обробки даних масивів, XML документів, реляційних баз даних та сторонніх джерел. LINQ також визначає набір імен методів (що називаються стандартними операторами запитів, або стандартними операторами послідовностей), а також правила перекладу, що має використовувати компілятор для перекладу текучих виразів у звичайні, використовуючи їх назву, лямбда-вирази та анонімні типи.

Набір операторів запитів, що визначені в LINQ, надається користувачу вигляді API.

ЗАВДАННЯ

Для демонстрації роботи розробити Unit-тести для моделі ПЗ відповідно до варіанту проекту. Unit-тести повинні створювати списки, словники або інші структури об’єктів, та робити із ними різні операції використовуючи LINQ запити.

Як альтернатива Unit-тестам можна використати консольне застосування (для максимальної кількості балів таких запитів повинно бути більше).

Повинні бути реалізовані такі запити:

1. Селекція частини інформації (метод Select)
2. Вибірка певної інформації (Where)
3. Операції як з списком List, так і з словником Dictionary
4. Власні методи розширювання
5. Використання анонімних класів та ініціалізаторів
6. Сортування за певним критерієм використовуючи шаблон IComparer
7. Конвертування списків в масив
8. Сортування масиву/списку за ім’ям чи за кількістю елементів

Тести повинні працювати виключно з моделлю об’єктів. Модель об'єктів повинна включати в себе різні типи даних (чисельні, стрічкові, логічні, множинні і т.д.), для кожного типу даних повинен бути мінімум 1 тест.

ХІД ВИКОНАННЯ

1. Реалізував зв’язні класи Candy та CandyCategory.

```
public class Candy
{
    12 usages  styanr *
    public Candy(string name, int caloriesCount, int categoryID, double price)
    {
        Name = name;
        CaloriesCount = caloriesCount;
        CategoryID = categoryID;
        Price = price;
    }
    6 usages
    public string Name { get; }
    4 usages
    public int CaloriesCount { get; }

    2 usages
    public double Price { get;  }
    8 usages
    public int CategoryID { get; }

}
```

Рис. 1. Клас Candy.

```
public class CandyCategory
{
    10 usages
    public string Name { get; set; }
    9 usages
    public  int ID { get; set; }
}
```

Рис. 2. Клас CandyCategory.

2. Розробив фіктивний об’єкт зі списками категорій та цукерок, що симулює табличні дані.

```
public static class ModelTests
{
    public static readonly List<CandyCategory> Categories = new()
    {
        new CandyCategory() { Name = "Sour", ID = 001 },
        new CandyCategory() { Name = "Fruity", ID = 002 },
        new CandyCategory() { Name = "Chewy", ID = 003 },
        new CandyCategory() { Name = "Hard", ID = 004 },
        new CandyCategory() { Name = "Chocolate", ID = 005 },
        new CandyCategory() { Name = "Boiled", ID = 006 },
        new CandyCategory() { Name = "Caramel", ID = 007 },
    };

    public static readonly List<Candy> Candies = new()
    {
        new Candy(name:"Gummy Bears", caloriesCount:130, categoryID:003, price:2.99),
        new Candy(name:"Jelly Beans", caloriesCount:100, categoryID:003, price:1.99),
        new Candy(name:"Lollipop", caloriesCount:75, categoryID:006, price:0.99),
        new Candy(name:"Sour Patch Kids", caloriesCount:120, categoryID:001, price:2.49),
        new Candy(name:"Kit Kat", caloriesCount:210, categoryID:005, price:1.79),
        new Candy(name:"Hershey's Kisses", caloriesCount:25, categoryID:005, price:0.99),
        new Candy(name:"Toffee", caloriesCount:160, categoryID:007, price:3.49),
        new Candy(name:"Werther's Original", caloriesCount:120, categoryID:007, price:2.99),
        new Candy(name:"Skittles", caloriesCount:140, categoryID:002, price:1.99),
        new Candy(name:"Starburst", caloriesCount:120, categoryID:002, price:1.99),
        new Candy(name:"Jawbreakers", caloriesCount:80, categoryID:004, price:1.49),
        new Candy(name:"Caramel Candy", caloriesCount:70, categoryID:007, price:2.49),
    };
```

Рис. 3. Клас ModelTests.

3. Вивів дані, використовуючи String.Format та метод LINQ ForEach.

```
public static void PrintData()
{
    Candies.ForEach(candy =>
    {
        Console.WriteLine(
            String.Format("|{0, -20}|{1, -4}|{2, -5}|{3, -3}|"
                , candy.Name, candy.CaloriesCount.ToString(), candy.Price.ToString(), candy.CategoryID.ToString()));
    });
}
```

Рис. 3. Метод PrintData, що друкує дані списку Candies.

4. Розробив структуру Dictionary, та заповнив її списками з цукерок за ключем категорії.

```
new *
public void DictionaryTest()
{
    var groupedByCategories :Dictionary<string,List<...>> = Categories
        .ToDictionary(category => category.Name, category => Candies
            .Where(candy => candy.CategoryID == category.ID)
            .ToList());
    Assert.AreEqual( expected: groupedByCategories.Keys.Count,  actual: Categories.Count);
}
```

Рис 4. Структура Dictionary.

5. Реалізував операції з іншими списковими структурами: SortedList, Queue, Stack, LinkedList.

```
public void SortedListTest()
{
    Dictionary<string, List<Candy>> groupedByCategories = Categories
        .ToDictionary(category => category.Name, category => Candies
            .Where(candy => candy.CategoryID == category.ID)
            .ToList());
    var sortedList = new SortedList<string, List<Candy>>(groupedByCategories);
    Assert.AreEqual( expected: sortedList.Keys[0],  actual: "Boiled");
}

[Test]
new *
public void QueueTest()
{
    var queue = new Queue<Candy>( collection: Candies);
    queue.Dequeue();
    Assert.AreEqual( expected: queue.Peek().Name,  actual: "Jelly Beans");
}
```

```
[Test]
styanr *
public void StackTest()
{
    var stack = new Stack<CandyCategory>(collection: Categories);
    stack.Pop();
    Assert.AreEqual(expected: stack.Peek().Name, actual: "Boiled");
}


[Test]
styanr *
public void LinkedListTest()
{
    var linkedList = new LinkedList<Candy>(collection: Candies);
    Assert.AreEqual(expected: linkedList.Count, actual: Candies.Count);
}
```

Рис. 5-6. Операції зі списковими структурами.

6. Дослідив групування.

```
[Test]
styanr *
public void GroupSortTest()
{
    var groupedByCategories :List<{ID,Names}> = Candies.GroupBy(candy => candy.CategoryID, candy => candy.Name,
        (id :int, nameList :IEnumerable<string>) => new
    {
        ID = id,
        Names = nameList
    }).OrderBy((a :{ID,Names}) => a.ID).ToList();

    Assert.AreEqual(expected: groupedByCategories[6].Names.Count(), actual: 3);
}
```

Рис. 7. Групування та сортування.

7. Створив власний метод розширення для списку.

```
public static class ExtensionMethods
{
    1 usage
    public static List<T> Shuffle<T>(this List<T> list, int? seed = null)
    {
        Random rng = (seed == null) ? new Random() : new Random(seed.Value);
        var shuffledList = new List<T>(list);
        int n = shuffledList.Count();
        while (n > 1) {
            n--;
            int k = rng.Next( maxValue: n + 1);
            (shuffledList[k], shuffledList[n]) = (shuffledList[n], shuffledList[k]);
        }
        return shuffledList;
    }

}
```

Рис. 8. Метод розширення Shuffle.

```
[Test]
new *
public void ShuffleTest()
{
    var nameList = Candies.Select(candy => candy.Name).ToList();
    var nameListShuffled = new List<string>(nameList).Shuffle(100);

    Assert.AreEqual( expected: nameList.Count(),  actual: nameListShuffled.Count());
    CollectionAssert.AreNotEqual( expected: nameList,  actual: nameListShuffled);
    CollectionAssert.AreEqual( expected: nameList.Order(),  actual: nameListShuffled.Order());
}
```

Рис. 9. Тестування розширеного методу.

8. Використав складніші операції з групуванням та сортуванням.

```
[Test]
new *
public void ComplexSortTest()
{
    var sortedDict :Dictionary<double,List<...>> = Candies // List<Candy>
        .GroupBy(candy => candy.Price) // IEnumerable<IGrouping<...>>
        .OrderByDescending(grouping => grouping.Count()) // IOrderedEnumerable<IGrouping<...>>
        .ToDictionary(grouping => grouping.Key,
            grouping => grouping.OrderBy(candy => candy, new CandyComparer()).ToList());


    Assert.AreEqual( expected: sortedDict.Count(), actual: 7);
    Assert.AreEqual( expected: sortedDict.Keys.First(), actual: 1.99, delta: 0.01);
    Assert.AreEqual( expected: sortedDict[0.99][0].Name, actual: "Hershey's Kisses");

    var mergedList :IEnumerable<Candy> = sortedDict.SelectMany(pair => pair.Value);
    Assert.AreEqual( expected: mergedList.Count(), actual: Candies.Count());
}
```

Рис. 10. Складніші операції з сортуванням та групуванням.

Були реалізовані такі запити:

1. Селекція частини інформації (метод Select)

```
var nameList = Candies.Select(candy => candy.Name).ToList();
```

Рис. 11. Селекція частини інформації.

2. Вибірка певної інформації (Where)

```
var groupedByCategories :Dictionary<string,List<...>> = Categories
    .ToDictionary(category => category.Name, category => Candies
        .Where(candy => candy.CategoryID == category.ID)
        .ToList());
```

Рис. 12. Вибірка певної інформації

3. Операції як з списком List, так і з словником Dictionary
4. Власні методи розширювання
5. Використання анонімних класів та ініціалізаторів

```
var groupedByCategories :List<{ID,Names}> = Candies.GroupBy(candy => candy.CategoryID, candy => candy.Name,
    (id :int , nameList :IEnumerable<string> ) => new
{
    ID = id,
    Names = nameList
}).OrderBy((a :{ID,Names} ) => a.ID).ToList();
```

Рис. 13. Використання анонімних класів та ініціалізаторів.

6. Сортування за певним критерієм використовуючи шаблон IComparer
7. Конвертування списків в масив

```
[Test]
new *
public void ToArrayTest()
{
    var caloriesArray int[] = Candies.Select(candy => candy.CaloriesCount).OrderDescending().ToArray();
    Assert.AreEqual( expected: caloriesArray.Length,  actual: Candies.Count());
    Assert.AreEqual( expected: caloriesArray[0],  actual: Candies.Select(candy => candy.CaloriesCount).Max());
}
```

Рис. 14. Конвертування списку в масив.

8. Сортування масиву/списку за ім’ям чи за кількістю елементів

ВИСНОВКИ

Під час виконання даної лабораторної роботи я навчився працювати з масивами, списками та словниками в мові програмування C#. Я дослідив різні методи та операції, які можна виконувати з цими структурами. Також я познайомився з технологією LINQ, яка є потужним інструментом для фільтрації, сортування та обробки даних. Я дослідив різні методи та оператори LINQ, такі як Where, Select, OrderBy, GroupBy тощо.