**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 3

**З дисципліни:** *“Моделювання та аналіз програмного забезпечення”*

**На тему:** *“Робота з масивами та структурами List, Dictionary. Використання  
технології LINQ* *”*

**Лектор:**

доц. каф. ПЗ

Сердюк П.В.

**Виконав:**

ст. гр. ПЗ-23

Михалевич П.-І.В.

**Прийняла:**

асист. каф. ПЗ

Цимбалюк Т.М.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема роботи:** Робота з масивами та структурами List, Dictionary. Використання технології LINQ.

**Мета роботи:** Ознайомлення з масивами та структурами List, Dictionary.  
Навчитись використовувати технології LINQ.

**TЕОРЕТИЧНІ ВІДОМОСТІ**

Модульне тестування (Unit testing) — це метод [тестування програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F" \o "Тестування програмного забезпечення), який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. У [процедурному програмуванні](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) модулем вважають окрему функцію або процедуру. В [об'єктно-орієнтованому програмуванні](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) — [метод](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)).

LINQ запит є виразом, що витягує дані з джерела даних. Запити зазвичай виражаються спеціалізованою мовою запитів. Згодом для різних типів джерел даних, наприклад SQL для реляційних баз даних та XQuery для XML, було розроблено різні мови. Тому розробникам доводиться вивчати нову мову запитів кожного типу джерела даних чи формату даних, котрим вони мають забезпечити підтримку. LINQ спрощує ситуацію, реалізуючи узгоджену модель роботи з даними для різних типів джерел та форматів даних. У запиті LINQ ви завжди працюєте з об'єктами. Ви використовуєте однакові базові шаблони коду для запиту та перетворення даних на XML-документи, бази даних SQL, набори даних ADO.NET, колекції .NET та будь-які інші формати, для яких доступний постачальник LINQ.

**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.

Повинні бути реалізовані

1. Селекція частини інформації (метод Select)
2. Вибірка певної інформації (Where)
3. Операції як з списком List так і з словником Dictionary
4. Реалізувати власні методи розширювання.
5. Показати використання анонімних класів та ініціалізаторів
6. Відсортувати за якимось критерієм використовуючи шаблон IComparer.
7. Конвертувати списки в масив.
8. Відсортувати масив/список за ім’ям чи за кількістю елементів.

Індивідуальний варіант – список країн-членів ООН.

**ХІД ВИКОНАННЯ**

Код для виконання лабораторної:

namespace ConsoleApp1

{

public class WorldRegion

{

public string Name{ get; set; }

public int Id{ get; set; }

}

public static class OONCountryExtension

{

public static string GetWorldRegionName(this OONCountry oonCountry, List<WorldRegion> worldRegions)

{

return worldRegions.Where(x => x.Id == oonCountry.WorldRegionId).First().Name;

}

}

public class OONCountryComparer : IComparer<OONCountry>

{

public int Compare(OONCountry x, OONCountry y)

{

return string.Compare(x.Name, y.Name);

}

}

public class OONCountry

{

public string Name { get; set; }

public int WorldRegionId { get; set; }

public int EntryYear { get; set; }

public static void Main()

{

var WorldRegions = new List<WorldRegion>()

{

new WorldRegion(){Name="Europe", Id=1},

new WorldRegion(){ Name="Africa", Id=2},

new WorldRegion(){ Name="America", Id=3},

new WorldRegion(){Name="Asia", Id=4},

new WorldRegion(){ Name="Oceania", Id=5}

};

var OONCountries = new List<OONCountry>()

{

new OONCountry{Name="France", WorldRegionId=1, EntryYear=1945},

new OONCountry{Name="Ukraine", WorldRegionId=1, EntryYear=1945},

new OONCountry{Name="USA", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Samoa", WorldRegionId=5, EntryYear=1976},

new OONCountry{Name="Kongo", WorldRegionId=2, EntryYear=1960},

new OONCountry{Name="Oman", WorldRegionId=4, EntryYear=1971},

new OONCountry{Name="Namibia", WorldRegionId=2, EntryYear=1990},

new OONCountry{Name="Laos", WorldRegionId=4, EntryYear=1955},

new OONCountry{Name="Kirybati", WorldRegionId=5, EntryYear=1999},

new OONCountry{Name="Latvia", WorldRegionId=1, EntryYear=1991},

new OONCountry{Name="Monako", WorldRegionId=1, EntryYear=1993},

new OONCountry{Name="Kanada", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Spain", WorldRegionId=1, EntryYear=1955},

new OONCountry{Name="Italy", WorldRegionId=1, EntryYear=1955},

new OONCountry{Name="Katar", WorldRegionId=4, EntryYear=1971},

new OONCountry{Name="Mexica", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Malaysia", WorldRegionId=4, EntryYear=1957},

};

Console.WriteLine();

Console.WriteLine("Ordered by entry year : ");

Console.WriteLine();

var sortedCountries = OONCountries.OrderBy(x => x.EntryYear).ToArray();

foreach (var country in sortedCountries)

{

Console.WriteLine("{0} entry in {1} and locates in Asia.", country.Name, country.EntryYear);

}

Console.WriteLine();

Console.WriteLine("Ordered by name with IComparer : ");

Console.WriteLine();

var sortedCountries\_2 = OONCountries;

sortedCountries\_2.Sort(new OONCountryComparer());

foreach (var country in sortedCountries\_2)

{

Console.WriteLine("{0} entry in {1} and locates in Asia.", country.Name, country.EntryYear);

}

Console.WriteLine();

Console.WriteLine("Make dictionary WorldRegionName - List<OONCountry> :");

Console.WriteLine();

// Make dictionary WorldRegionName - List<OONCountry>

var OONCountriesDictionary = OONCountries.GroupBy(x => x.WorldRegionId).

ToDictionary(x => WorldRegions.Where(m => x.Key == m.Id).First().Name, y => y.OrderBy(z => z.Name).ToList());

foreach (var (key, countries) in OONCountriesDictionary)

{

foreach (var country in countries)

{

Console.WriteLine(String.Format("{0} locates in {1} and entry in OON in {2}", country.Name, key, country.EntryYear));

}

}

Console.WriteLine();

Console.WriteLine("Prints only selected names :");

Console.WriteLine();

// Prints only selected names

var countriesNames = OONCountries.Select(x => x.Name).ToList();

foreach (var countryName in countriesNames)

{

Console.WriteLine(countryName);

}

Console.WriteLine();

Console.WriteLine("Print countries which is in Asia :");

Console.WriteLine();

// Print countries which is in Asia

var countriesInAsia = OONCountries.Where(x => x.WorldRegionId == WorldRegions.Where(x => x.Name == "Asia").First().Id).ToList();

foreach (var country in countriesInAsia)

{

Console.WriteLine("{0} entry in {1} and locates in Asia.", country.Name, country.EntryYear);

}

Console.WriteLine();

Console.WriteLine("Cast to array and print :");

Console.WriteLine();

// Cast to array and print

var countriesArray = OONCountries.ToArray();

foreach (var country in countriesArray)

{

Console.WriteLine("{0} entry in {1} and locates in {2}.", country.Name, country.EntryYear,

country.GetWorldRegionName(WorldRegions));

}

Console.WriteLine();

Console.WriteLine("Order world regions by count of counries in list : ");

Console.WriteLine();

var sortedRegions = WorldRegions.OrderBy(x => OONCountries.Where(y => y.WorldRegionId == x.Id).Count()).ToList();

foreach (var region in sortedRegions)

{

Console.WriteLine("{0} with id {1}", region.Name, region.Id);

}

}

}

}

Unit тести:

namespace TestProject\_OON\_Countries

{

[TestClass]

public class UnitTest

{

public List<WorldRegion> WorldRegions

{

get

{

return new List<WorldRegion>()

{

new WorldRegion(){Name="Europe", Id=1},

new WorldRegion(){ Name="Africa", Id=2},

new WorldRegion(){ Name="America", Id=3},

new WorldRegion(){Name="Asia", Id=4},

new WorldRegion(){ Name="Oceania", Id=5}

};

}

}

// Specify the second data source.

public List<OONCountry> OONCountries

{

get

{

return new List<OONCountry>()

{

new OONCountry{Name="France", WorldRegionId=1, EntryYear=1945},

new OONCountry{Name="Ukraine", WorldRegionId=1, EntryYear=1945},

new OONCountry{Name="USA", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Samoa", WorldRegionId=5, EntryYear=1976},

new OONCountry{Name="Kongo", WorldRegionId=2, EntryYear=1960},

new OONCountry{Name="Oman", WorldRegionId=4, EntryYear=1971},

new OONCountry{Name="Namibia", WorldRegionId=2, EntryYear=1990},

new OONCountry{Name="Laos", WorldRegionId=4, EntryYear=1955},

new OONCountry{Name="Kirybati", WorldRegionId=5, EntryYear=1999},

new OONCountry{Name="Latvia", WorldRegionId=1, EntryYear=1991},

new OONCountry{Name="Monako", WorldRegionId=1, EntryYear=1993},

new OONCountry{Name="Kanada", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Spain", WorldRegionId=1, EntryYear=1955},

new OONCountry{Name="Italy", WorldRegionId=1, EntryYear=1955},

new OONCountry{Name="Katar", WorldRegionId=4, EntryYear=1971},

new OONCountry{Name="Mexica", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Malaysia", WorldRegionId=4, EntryYear=1957}

};

}

}

[TestMethod]

public void TestMethod1()

{

int expectedCountInEurope = 6;

Assert.AreEqual(expectedCountInEurope, OONCountries.Where(x => x.WorldRegionId == WorldRegions.Where(x => x.Name == "Europe").First().Id).Count());

}

[TestMethod]

public void TestMethod2()

{

int expectedCountInEurope = 4;

Assert.AreEqual(expectedCountInEurope, OONCountries.Where(x => x.WorldRegionId == WorldRegions.Where(x => x.Name == "Asia").First().Id).Count());

}

[TestMethod]

public void TestMethod3()

{

var expectedSortedOONCountries = new List<OONCountry>()

{

new OONCountry{Name="France", WorldRegionId=1, EntryYear=1945},

new OONCountry{Name="Italy", WorldRegionId=1, EntryYear=1955},

new OONCountry{Name="Kanada", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Katar", WorldRegionId=4, EntryYear=1971},

new OONCountry{Name="Kirybati", WorldRegionId=5, EntryYear=1999},

new OONCountry{Name="Kongo", WorldRegionId=2, EntryYear=1960},

new OONCountry{Name="Laos", WorldRegionId=4, EntryYear=1955},

new OONCountry{Name="Latvia", WorldRegionId=1, EntryYear=1991},

new OONCountry{Name="Malaysia", WorldRegionId=4, EntryYear=1957},

new OONCountry{Name="Mexica", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Monako", WorldRegionId=1, EntryYear=1993},

new OONCountry{Name="Namibia", WorldRegionId=2, EntryYear=1990},

new OONCountry{Name="Oman", WorldRegionId=4, EntryYear=1971},

new OONCountry{Name="Samoa", WorldRegionId=5, EntryYear=1976},

new OONCountry{Name="Spain", WorldRegionId=1, EntryYear=1955},

new OONCountry{Name="Ukraine", WorldRegionId=1, EntryYear=1945},

new OONCountry{Name="USA", WorldRegionId=3, EntryYear=1945},

};

expectedSortedOONCountries.Sort(new OONCountryComparer());

var sorted = OONCountries.OrderBy(x => x.Name).ToList();

var comparer = new OONCountryComparer();

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

{

Assert.AreEqual(expectedSortedOONCountries[i].Name, sorted[i].Name);

}

}

[TestMethod]

public void TestMethod4()

{

var expectedSortedWorldRegions = new List<WorldRegion>()

{

new WorldRegion(){ Name="Africa", Id=2},

new WorldRegion(){ Name="Oceania", Id=5},

new WorldRegion(){ Name="America", Id=3},

new WorldRegion(){Name="Asia", Id=4},

new WorldRegion(){Name="Europe", Id=1},

};

var sorted = WorldRegions.OrderBy(x => OONCountries.Where(y => y.WorldRegionId == x.Id).Count()).ToList();

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

{

Assert.AreEqual(expectedSortedWorldRegions[i].Name, sorted[i].Name);

}

}

}

}

1. Використання методу Select:

var countriesNames = OONCountries.Select(x => x.Name).ToList();

foreach (var countryName in countriesNames)

{

Console.WriteLine(countryName);

}

2 - 3. Використання методу Where, OrderBy для сортування та GroupBy з ToDictionary для перетворення з масиву у dictionary:

Console.WriteLine();

Console.WriteLine("Make dictionary WorldRegionName - List<OONCountry> :");

Console.WriteLine();

// Make dictionary WorldRegionName - List<OONCountry>

var OONCountriesDictionary = OONCountries.GroupBy(x => x.WorldRegionId).

ToDictionary(x => WorldRegions.Where(m => x.Key == m.Id).First().Name, y => y.OrderBy(z => z.Name).ToList());

foreach (var (key, countries) in OONCountriesDictionary)

{

foreach (var country in countries)

{

Console.WriteLine(String.Format("{0} locates in {1} and entry in OON in {2}", country.Name, key, country.EntryYear));

}

}

4. Реалізував власний метод розширення і конвертування в масив (7):

public static class OONCountryExtension

{

public static string GetWorldRegionName(this OONCountry oonCountry, List<WorldRegion> worldRegions)

{

return worldRegions.Where(x => x.Id == oonCountry.WorldRegionId).First().Name;

}

}

Console.WriteLine();

Console.WriteLine("Cast to array and print :");

Console.WriteLine();

// Cast to array and print

var countriesArray = OONCountries.ToArray();

foreach (var country in countriesArray)

{

Console.WriteLine("{0} entry in {1} and locates in {2}.", country.Name, country.EntryYear,

country.GetWorldRegionName(WorldRegions));

}

5. Використання анонімних ініціалізаторів класів:

var WorldRegions = new List<WorldRegion>()

{

new WorldRegion(){Name="Europe", Id=1},

new WorldRegion(){ Name="Africa", Id=2},

new WorldRegion(){ Name="America", Id=3},

new WorldRegion(){Name="Asia", Id=4},

new WorldRegion(){ Name="Oceania", Id=5}

};

var OONCountries = new List<OONCountry>()

{

new OONCountry{Name="France", WorldRegionId=1, EntryYear=1945},

new OONCountry{Name="Ukraine", WorldRegionId=1, EntryYear=1945},

new OONCountry{Name="USA", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Samoa", WorldRegionId=5, EntryYear=1976},

new OONCountry{Name="Kongo", WorldRegionId=2, EntryYear=1960},

new OONCountry{Name="Oman", WorldRegionId=4, EntryYear=1971},

new OONCountry{Name="Namibia", WorldRegionId=2, EntryYear=1990},

new OONCountry{Name="Laos", WorldRegionId=4, EntryYear=1955},

new OONCountry{Name="Kirybati", WorldRegionId=5, EntryYear=1999},

new OONCountry{Name="Latvia", WorldRegionId=1, EntryYear=1991},

new OONCountry{Name="Monako", WorldRegionId=1, EntryYear=1993},

new OONCountry{Name="Kanada", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Spain", WorldRegionId=1, EntryYear=1955},

new OONCountry{Name="Italy", WorldRegionId=1, EntryYear=1955},

new OONCountry{Name="Katar", WorldRegionId=4, EntryYear=1971},

new OONCountry{Name="Mexica", WorldRegionId=3, EntryYear=1945},

new OONCountry{Name="Malaysia", WorldRegionId=4, EntryYear=1957},

};

6. Використання IComparer для сортування:

public class OONCountryComparer : IComparer<OONCountry>

{

public int Compare(OONCountry x, OONCountry y)

{

return string.Compare(x.Name, y.Name);

}

}

Console.WriteLine();

Console.WriteLine("Ordered by name with IComparer : ");

Console.WriteLine();

var sortedCountries\_2 = OONCountries;

sortedCountries\_2.Sort(new OONCountryComparer());

foreach (var country in sortedCountries\_2)

{

Console.WriteLine("{0} entry in {1} and locates in Asia.", country.Name, country.EntryYear);

}

8. Відсортував список за роком вступу в ООН:

Console.WriteLine();

Console.WriteLine("Ordered by entry year : ");

Console.WriteLine();

var sortedCountries = OONCountries.OrderBy(x => x.EntryYear).ToArray();

foreach (var country in sortedCountries)

{

Console.WriteLine("{0} entry in {1} and locates in Asia.", country.Name, country.EntryYear);

}

Додатково використав Where із складнішими умовами:

1) Найти всі країни які знаходяться в Азії:

Console.WriteLine();

Console.WriteLine("Print countries which is in Asia :");

Console.WriteLine();

// Print countries which is in Asia

var countriesInAsia = OONCountries.Where(x => x.WorldRegionId == WorldRegions.Where(x => x.Name == "Asia").First().Id).ToList();

foreach (var country in countriesInAsia)

{

Console.WriteLine("{0} entry in {1} and locates in Asia.", country.Name, country.EntryYear);

}

2) Посортувати регіони за кількістю країн в списку, які знаходяться в даних регіонах:

Console.WriteLine();

Console.WriteLine("Order world regions by count of counries in list : ");

Console.WriteLine();

var sortedRegions = WorldRegions.OrderBy(x => OONCountries.Where(y => y.WorldRegionId == x.Id).Count()).ToList();

foreach (var region in sortedRegions)

{

Console.WriteLine("{0} with id {1}", region.Name, region.Id);

}

**РЕЗУЛЬТАТИ**

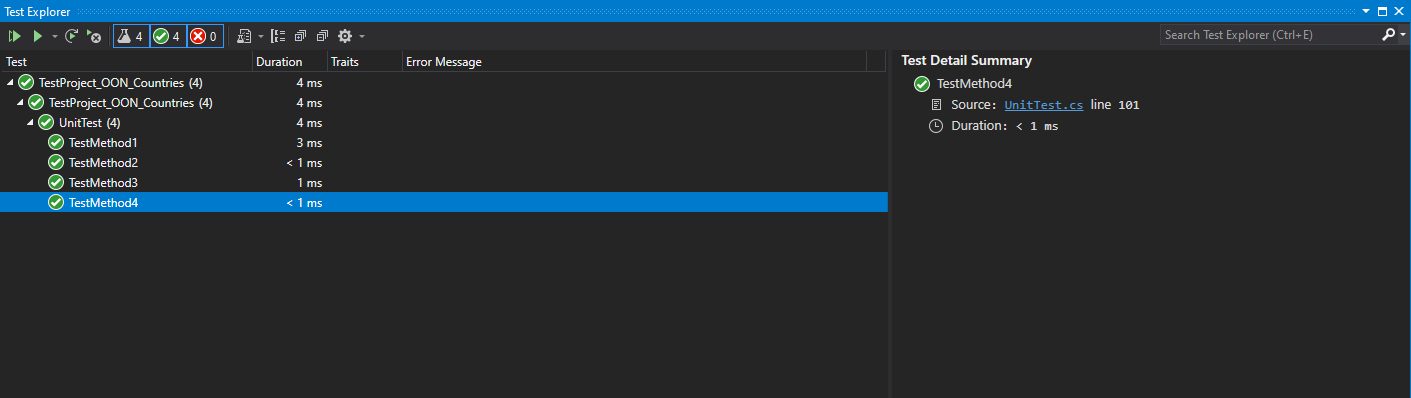


Рис.1. Результати Unit тестів



Рис. 2. Результат використання методу Select

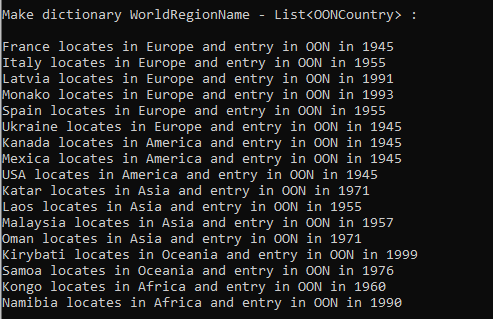


Рис.3. Результат використання методу Where та GroupBy

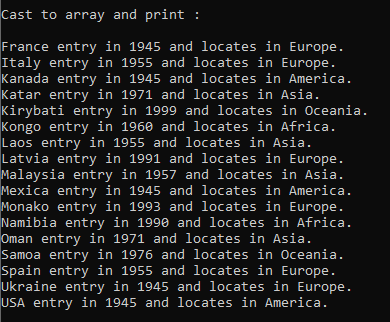


Рис.4. Результат використання методу розширення

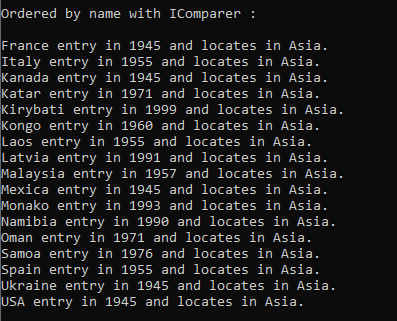


Рис.5. Результати використання IComparer для сортування

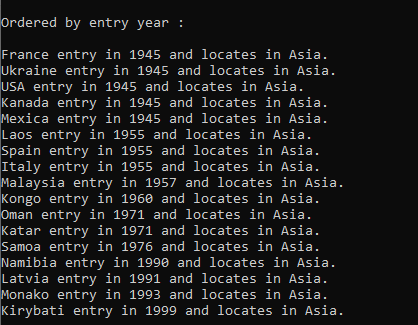


Рис.6. Результати сортування по року вступу в ООН

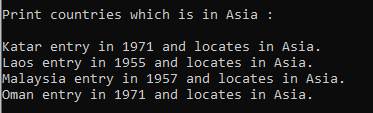


Рис.7. Результат пошуку всіх країн, що знаходяться в Азії

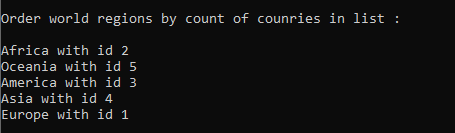


Рис.8. Результат сортування списку регіонів по кількості країн в них у відповідному списку

**ВИСНОВКИ**

На даній лабораторній роботі я ознайомився і вивчив такі структури даних як List – простий список даних та Dictionary – список даних погрупованих по унікальному ключі.

Також ознайомився з технологією LINQ та технологією модульних тестів (Unit-тести). LINQ — це API запити даних, який надає можливості запитів до . NET із синтаксисом, подібним до SQL. Запити LINQ використовують для зручного оперування з колекціями C# та швидкої взаємодії з базами даних. За допомогою LINQ-запитів можна використовувати дуже багато різноманітних функцій. На даній лабораторній роботі використав такі методи розширення: ToDictionary, ToList, ToArray (переведення до типу), Select(вибірка), Where(вибірка по критерію), Sort (сортування), OrderBy(сортування по ключу), ForEach (використання функції для кожного елементу). Ці розширення спрощують роботу програміста та зменшують кількість коду. Також створив власний метод розширення та створив власний клас, наслідуваний від IComparer, для сортування даних. Unit-тести – програма яка тестує індивідуальні частини коду для власної перевірки. Модульні тести завжди тестують маленький шматок коду, саме тому і називаються модульними, бо використовується певний модуль програми. Програміст сам обирає, що і як тестувати.