# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №2**

**«Робота з масивами та структурами List, Dictionary. Використання технології LINQ»**

**з дисципліни «Конструювання програмного забеспечення»**

Виконав:

слухач групи ПЗС-21

Гринчук Тарас

Прийняв:

ст. викл. Тушницький Р.Б.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

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

**Мета роботи**: Ознайомлення з масивами та структурами List, Dictionary в C#. Навчитись створювати проекти з використанням технології LINQ.

## Хід роботи

1. В середовищі Visual Studio створимо новий проект «Консольное приложение» (рис.1).

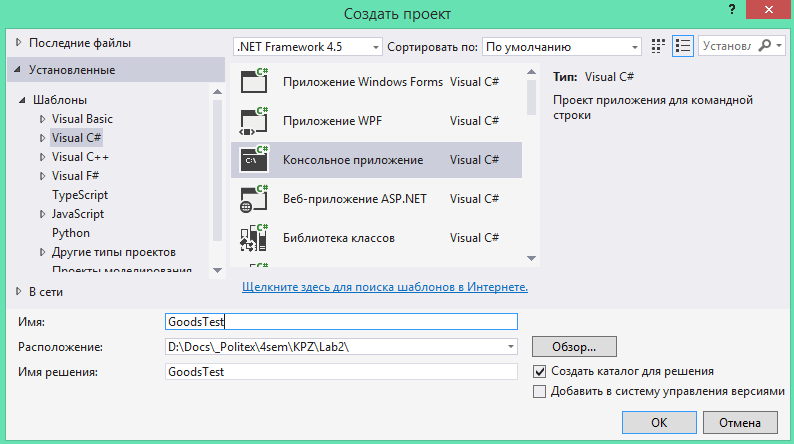


Рис. 1. Створення нового консольного проекту

Реалізуємо два зв’язаних класи: Goods та TypeOfGoods. У класі Goods міститься посилання на відповідний тип товару TypeOfGoods:

class Goods

{

public String Name { get; set; }

public int Typeid { get; set; }

public Goods(string name, int typeid)

{

Name = name;

Typeid = typeid;

}

}

class TypeOfGoods

{

public string Name { get; set; }

public int ID { get; set; }

}

2) Розробимо ”фіктивний об’єкт”, що симулює табличні дані:

class Program

{

public List<TypeOfGoods> types

{

get

{

return new List<TypeOfGoods>()

{

new TypeOfGoods() {Name = "Product", ID = 1},

new TypeOfGoods() {Name = "Goods", ID = 2},

new TypeOfGoods() {Name = "Service", ID = 3},

};

}

}

public List<Goods> goods

{

get

{

return new List<Goods>()

{

new Goods{Name = "Cake", Typeid = 1},

new Goods{Name = "Candy", Typeid = 1},

new Goods{Name = "Cookies", Typeid = 1},

new Goods{Name = "Milk", Typeid = 2},

new Goods{Name = "Sugar", Typeid = 2},

new Goods{Name = "Flour", Typeid = 2},

new Goods{Name = "Cooking", Typeid = 3},

new Goods{Name = "Delivery", Typeid = 3},

};

}

}

static void Main(string[] args)

{

//Точка входу в програму

}

}

3)Перевизначити метод ToString(). Використовуючи команду String.Format

public override string ToString()

{

return String.Format("{0} (Typeid = {1})", Name, Typeid);

}

4) Вивести на консоль усі дані використовуючи цикл foreach. Результат на рис. 2.

static void Main(string[] args)

{

Program P = new Program();

foreach (var g in P.goods)

{

Console.WriteLine(g);

}

Console.ReadKey();

}

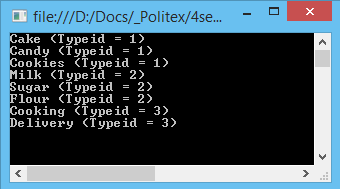


Рис. 2. Результат циклу foreach

5) Розробити і заповнити структуру, у якій за ключем можна витягнути окремий клас, або їх список (Результат на рис. 3):

//Хеш-таблиця типів товарів

Dictionary<int, TypeOfGoods> dict1 = new Dictionary<int, TypeOfGoods>();

foreach (var g in P.types)

{

dict1.Add(g.ID, g); //g.ID - ключ

}

//Виведемо екземпляр з ключом = 2

Console.WriteLine("\nID = 2 Name = " + dict1[2].Name);

//За допомогою linq виведемо всі елементи з ключами < 3

var res =

from n in P.types

where n.ID < 3

select n;

Console.WriteLine("LINQ: ID < 3:");

foreach (var x in res)

{

Console.WriteLine(x);

}

//Хеш-таблиця списків товарів по їх типах

Dictionary<int, List<Goods>> dict2 = new Dictionary<int, List<Goods>>();

foreach (var t in P.types)

{

//список товарів з типом = t

List<Goods> L = new List<Goods>();

foreach (var g in P.goods)

if (g.Typeid == t.ID) L.Add(g);

dict2.Add(t.

ID, L);

}

Console.WriteLine("\nTYPE = 3. Goods :");

//Виведемо всі товари з Typeid == 3

List<Goods> list = dict2[3];

foreach (var g in list)

Console.WriteLine(g);

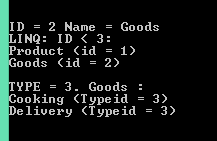


Рис. 3. Результат проведених вибірок

6) Відсортувати за якимось критерієм використовуючи шаблон IComparer та IComparable.

Реалізуємо інтерфейс IComparable, щоб відсортувати список товарів по іменам у прямому порядку:

class Goods : IComparable<Goods>

{

public int CompareTo(Goods other)

{

//Сортування за алфавітом назв

return Name.CompareTo(other.Name);

}

}

Реалізуємо інтерфейс IComparer для нового класу GoodsByIDDescOrder, щоб відсортувати список товарів по типам у зворотньому порядку (якщо тип співдадає – сортуємо по імені):

class GoodsByIDDescOrder : IComparer<Goods>

{

public int Compare(Goods x, Goods y)

{

int res = y.Typeid.CompareTo(x.Typeid);

if (res == 0) //тоді за назвою

res = x.CompareTo(y);

return res;

}

}

Код сортування: (результат виконання на рис. 4):

//відсортуємо список за алфавітом

Console.WriteLine("\nSort by name:\n");

List<Goods> sorted = new List<Goods>(P.goods);

sorted.Sort();

foreach (var s in sorted) Console.WriteLine(s);

//відсортуємо список в зворотньому порядку по ID

Console.WriteLine("\nSort by ID (desc. order):\n");

sorted.Sort(new GoodsByIDDescOrder());

foreach (var s in sorted) Console.WriteLine(s);

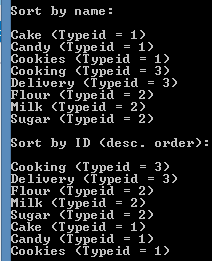


Рис. 4. Сортування списку товарів

7) Конвертуємо списки в масиви:

//переведемо список товарів в масив

Goods[] arrGoods = P.goods.ToArray();

//переведемо список типів товарів в масив

TypeOfGoods[] arrTypes = types.ToArray();

Відсортуємо масив товарів за іменами товарів. Оскільки, ми вже раніше реалізували інтерфейс IComparable для класу Goods сортування масиву arrGoods матиме вигляд:

//сортування товарів за іменем

Console.WriteLine("\nArray of goods before sorting:\n");

foreach (var s in arrGoods) Console.WriteLine(s);

Array.Sort(arrGoods);

Console.WriteLine("\nArray of goods after sorting:\n");

foreach (var s in arrGoods) Console.WriteLine(s);

Результати сортування товарів зображені на рис. 5:

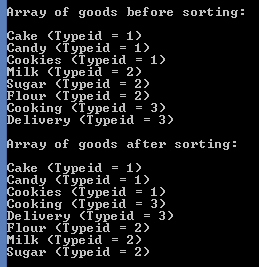


Рис. 5. Сортування масиву товарів

Відсортуємо масив типів товарів за кількістю товарів, що до них належать. Для цього створимо клас, який реалізує інтерфейс IComparer для класу TypeOfGoods:

//сортування типів товарів за кількістю товарів, що до них входять

class TypesByCountOfGoods : IComparer<TypeOfGoods>

{

public int Compare(TypeOfGoods x, TypeOfGoods y)

{

return x.CountOfGoods.CompareTo(y.CountOfGoods);

}

}

Саме сортування матиме вигляд:

//сортування за к-стю товарів

Array.Sort(arrTypes, new TypesByCountOfGoods());

Console.WriteLine("\nArray of types after sorting:\n");

foreach (var s in arrTypes) Console.WriteLine(s + " Count = " + s.CountOfGoods);

Результати сортування типів товарів зображені на рис. 6:

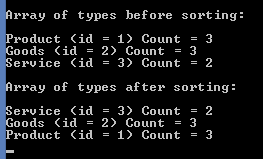


Рис. 6. Результат сортування типів товарів

## Текст програми

using System;

using System.Collections.Generic;

using System.Collections.Specialized;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace GoodsTest

{

class Program

{

public List<TypeOfGoods> types

{

get

{

return new List<TypeOfGoods>()

{

new TypeOfGoods() {Name = "Product", ID = 1},

new TypeOfGoods() {Name = "Goods", ID = 2},

new TypeOfGoods() {Name = "Service", ID = 3},

};

}

}

public List<Goods> goods

{

get

{

return new List<Goods>()

{

new Goods{Name = "Cake", Typeid = 1},

new Goods{Name = "Candy", Typeid = 1},

new Goods{Name = "Cookies", Typeid = 1},

new Goods{Name = "Milk", Typeid = 2},

new Goods{Name = "Sugar", Typeid = 2},

new Goods{Name = "Flour", Typeid = 2},

new Goods{Name = "Cooking", Typeid = 3},

new Goods{Name = "Delivery", Typeid = 3},

};

}

}

static void Main(string[] args)

{

Program P = new Program();

foreach (var g in P.goods)

{

Console.WriteLine(g);

}

Console.WriteLine("\n");

//Хеш-таблиця типів товарів

Dictionary<int, TypeOfGoods> dict1 = new Dictionary<int, TypeOfGoods>();

foreach (var g in P.types)

{

dict1.Add(g.ID, g); //g.ID - ключ

}

//Виведемо екземпляр з ключом = 2

Console.WriteLine("\nID = 2 Name = " + dict1[2].Name);

//За допомогою linq виведемо всі елементи з ключами < 3

var res =

from n in P.types

where n.ID < 3

select n;

Console.WriteLine("LINQ: ID < 3:");

foreach (var x in res)

{

Console.WriteLine(x);

}

//Хеш-таблиця списків товарів по їх типах

Dictionary<int, List<Goods>> dict2 = new Dictionary<int, List<Goods>>();

var types = P.types;

foreach (var t in types)

{

//список товарів з типом = t

List<Goods> L = new List<Goods>();

foreach (var g in P.goods)

if (g.Typeid == t.ID) L.Add(g);

dict2.Add(t.ID, L);

//Запишемо к-ть товарів для кожного типу в статичну змінну

t.CountOfGoods = L.Count;

}

Console.WriteLine("\nTYPE = 3. Goods :");

//Виведемо всі товари з Typeid == 3

List<Goods> list = dict2[3];

foreach (var g in list)

Console.WriteLine(g);

//відсортуємо список за алфавітом

Console.WriteLine("\nSort by name:\n");

List<Goods> sorted = new List<Goods>(P.goods);

sorted.Sort();

foreach (var s in sorted) Console.WriteLine(s);

//відсортуємо список в зворотньому порядку по ID

Console.WriteLine("\nSort by ID (desc. order):\n");

sorted.Sort(new GoodsByIDDescOrder());

foreach (var s in sorted) Console.WriteLine(s);

//переведемо список товарів в масив

Goods[] arrGoods = P.goods.ToArray();

//сортування товарів за іменем

Console.WriteLine("\nArray of goods before sorting:\n");

foreach (var s in arrGoods) Console.WriteLine(s);

Array.Sort(arrGoods);

Console.WriteLine("\nArray of goods after sorting:\n");

foreach (var s in arrGoods) Console.WriteLine(s);

//переведемо список типів товарів в масив

TypeOfGoods[] arrTypes = types.ToArray();

Console.WriteLine("\nArray of types before sorting:\n");

foreach (var s in arrTypes) Console.WriteLine(s + " Count = " + s.CountOfGoods);

//сортування за к-стю товарів

Array.Sort(arrTypes, new TypesByCountOfGoods());

Console.WriteLine("\nArray of types after sorting:\n");

foreach (var s in arrTypes) Console.WriteLine(s + " Count = " + s.CountOfGoods);

Console.ReadKey();

}

}

class Goods : IComparable<Goods>

{

public String Name { get; set; }

public int Typeid { get; set; }

public int CompareTo(Goods other)

{

//Сортування за алфавітом назв

return Name.CompareTo(other.Name);

}

public override string ToString()

{

return String.Format("{0} (Typeid = {1})", Name, Typeid);

}

public Goods(string name, int typeid)

{

Name = name;

Typeid = typeid;

}

public Goods()

{

}

}

class TypeOfGoods

{

public int CountOfGoods = 0; //ксть ттоарів, що входять до цього типу

public TypeOfGoods()

{

}

public string Name { get; set; }

public int ID { get; set; }

public override string ToString()

{

return String.Format("{0} (id = {1})", Name, ID);

}

}

//сортування за ID у зворотньому порядку

class GoodsByIDDescOrder : IComparer<Goods>

{

public int Compare(Goods x, Goods y)

{

int res = y.Typeid.CompareTo(x.Typeid);

if (res == 0) //тоді за назвою

res = x.CompareTo(y);

return res;

}

}

//сортування типів товарів за кількістю товарів, що до них входять

class TypesByCountOfGoods : IComparer<TypeOfGoods>

{

public int Compare(TypeOfGoods x, TypeOfGoods y)

{

return x.CountOfGoods.CompareTo(y.CountOfGoods);

}

}

}

## ВИСНОВКИ

На даній лабораторній роботі я ознайомився з масивами та структурами List, Dictionary в C#. Навчився створювати проекти з використанням технології LINQ.