Лабораторна робота №№ 4, 5
Розширені можливості мови C#

План
1. Класи для роботи з колекціями.
2. Основи LINQ.
3. Завдання.

1. На практиці часто виникає задача обробки груп однотипних даних (пошук, сортування, перетворення тощо). Для зберігання заздалегідь невизначеної кількості різних об'єктів використовують спеціальні **контейнерні класи** (**колекції**), що реалізують функціонал стандартних структур даних, таких, наприклад, як стеки, черги, списки, словники і т. п. Усі класи-контейнери C# можна розділити на дві групи: **неузагальнені** та **узагальнені**.

Неузагальнені колекції (оголошені у просторі імен System.Collections) працюють лише з об'єктами наперед заданих певних типів даних. Оскільки всі класи C# є похідними від System.Object, то зазвичай неузагальнені колекції оперують наборами об'єктів цього типу (або його нащадками).

Наприклад, динамічний масив змінної довжини може бути оголошений та ініціалізований наступним чином:

```
// Динамічний масив елементів типу object
ArrayList lst = new ArrayList() {3.14159 , 2.71828, "Hello world!", 'c', 3.0f};
```

На сьогодні використання неузагальнених колекцій не рекомендується, тому що вони мають відносно низьку продуктивність.

Узагальнені колекції оголошені в просторі імен System.Collections.Generic і є аналогами відповідних шаблонних (template) класів мови C++. Вони дозволяють працювати з об'єктами різних типів даних, які фактично задаються як їх параметри. Наприклад:

```
// Список дійсних чисел
List<double> num s = new List<double>() {1. 1 , 2. 2 , 3. 14 , 4. 78 , 5. 99};
```

Перелік узагальнених колекцій, що найчастіше використовуються, наведено в табл. 1.

Розглянемо приклад роботи з колекцією дійсних чисел, сформованою за допомогою генератора випадкових чисел. Побудуємо таку вибірку з *n* чисел і виведемо її на екран.

Це можна зробити, наприклад, таким чином:

Таблиця 1. Основні узагальнені колекції

Узагальнений клас	Призначення
Dictionary<TKey, TValue>	Узагальнена колекція пар <ключ, значення>
LinkedList<T>	Двохзв’язний список
List<T>	Динамічний список
Queue<T>	Узагальнена черга
SortedDictionary<TKey, TValue>	Узагальнений словник відсортованої множини пар <ключ, значення>
SortedSet<T>	Колекція унікальних відсортованих об'єктів
Stack<T>	Узагальнений стек

```
using System;
using System.Collections.Generic;

public class Programm
{
   public static void Main()
   {
      var rand = new Random(); // Створення генератора випадкових чисел
      var nums = new List<double>();
      var num = 0;

      Console.WriteLine("Enter n");
      num = Convert.ToInt32(Console.ReadLine());
       // Заповнення списку випадковими величинами дійсного типу
      for (int i = 0; i < num; i++)
         nums.Add(rand.NextDouble());
      // Виведення макс., мін. та середнього значення вибірки
      Console.WriteLine("Max value: {0}", nums.Max());
      Console.WriteLine("Min value: {0}", nums.Min());
      Console.WriteLine("Avg value: {0}", nums.Average());
      // Виведення вибірки
      Console.WriteLine("Random values");
      foreach (double it in nums)
         Console.WriteLine(it);

      Console.ReadKey();
   }
}
```

Тут слід звернути увагу на те, що в наведеному прикладі використовується тип даних **var**, що автоматично виводиться. Компілятор за значенням, яке ініціалізує змінну, сам визначає, до якого типу даних віднести відповідну змінну.

Результат роботи програми наведено на рис. 1.

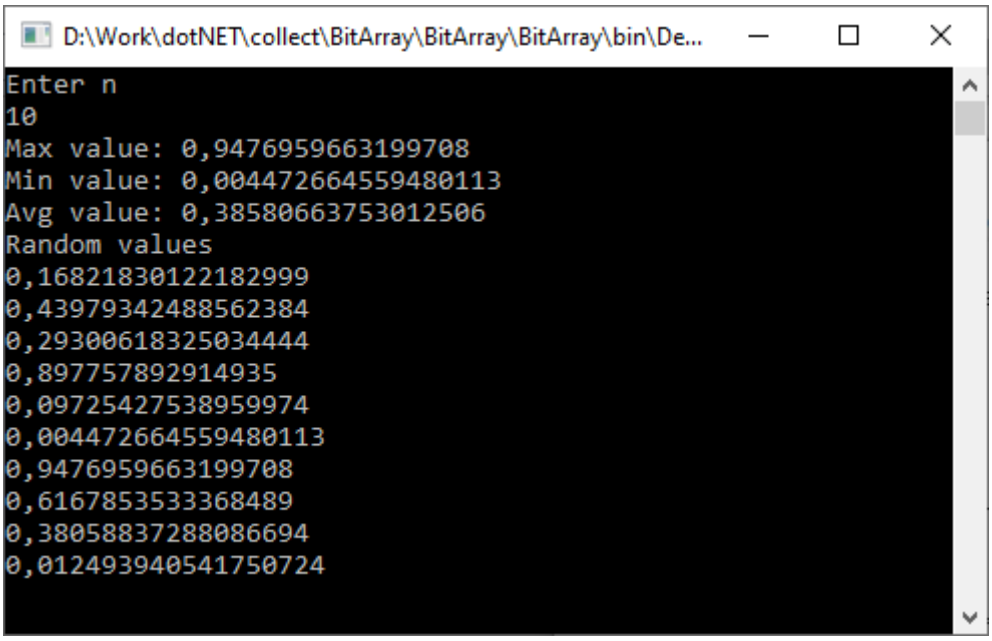

Рис. 1 – Приклад роботи із узагальненою колекцією List

Як видно з наведеного прикладу датчик випадкових чисел генерує значення в діапазоні від 0 до 1.

Розглянемо ще один приклад роботи з узагальненою колекцією SortedDictionary<TKey, TValue>. Напишемо програму, в якій відбувається зчитування текстового файлу і розбиття вмісту на окремі слова (словом будемо вважати довільну послідовність символів, укладену між пробілами). При цьому кожне зчитане слово зберігається у словнику, де ключем є саме слово, а відповідним йому значенням – кількість його поява у файлі.

Таку програму можна реалізувати, наприклад, так:

```
using System;
using System.IO;
using System.Collections.Generic;

public class Programm
{
   public static void Main()
   {
      var path = "d:/tmp/text.txt";
      var dic = new SortedDictionary<string, int>();
      string[] buffer;

      try
```

```
        {
            StreamReader sr = new StreamReader(new FileStream(path,
                                                    FileMode.Open));

            while (!sr.EndOfStream)
            {
                buffer = sr.ReadLine().Split(' ');
                foreach (var it in buffer)
                    if (dic.ContainsKey(it))
                        dic[it]++;
                    else
                        dic.Add(it, 1);
            }
            sr.Close();
            Console.WriteLine("Содержимое файла");
            foreach (var it in dic)
                Console.WriteLine("{0}: {1}", it.Key, it.Value);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
        Console.ReadKey();
    }
}
```

В наведеному прикладі створюється потоковий об'єкт для читання файлу (при цьому використовуються класи StreamReader і FileStream з простору імен System.IO), а потім здійснюється строкове зчитування даних із файлового потоку. Кожен черговий рядок за допомогою Split() розбивається на масив рядків (окремих слів), кожен елемент якого у вихідному рядку відокремлюється від іншого елемента заданим роздільником – пропуском (інакше кажучи, функція Split() поділяє вихідний рядок на безліч окремих елементів за критерієм заданого роздільника). Потім кожне отримане слово додається до колекції – сортований словник SortedDictionary.

Результат роботи програми наведено на рис. 2. Як видно з даного рисунка, розділові знаки окремо не обробляються в даній програмі, що є її недоліком, тому що одне і теж слово з комою і точкою в кінці в даному прикладі будуть вважатися різними.

Рис. 2 – Приклад роботи з колекцією SortedDictionary

2. Для різних маніпуляцій із колекціями зручно використовувати технологію LINQ.

LINQ (Language-Integrated Query) – це проста мова запитів до різних джерел даних, що підтримують інтерфейс IEnumerable. До них, наприклад, належать масиви, стандартні колекції, набори даних DataSet, документи XML та багато іншого.

Розглянемо приклад застосування запиту LINQ до колекції SortedDictionary для знаходження у ній заданого ключового значення.

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;

public class Programm
{
   public static void Main()
   {
      var path = "d:/tmp/text.txt";
      var dic = new SortedDictionary<string, int>();
      string[] buffer;

      try
      {
         StreamReader sr = new StreamReader(new FileStream(path,
                                                FileMode.Open));
```

```
            while (!sr.EndOfStream)
            {
               buffer = sr.ReadLine().Split(' ');
               foreach (var it in buffer)
                  if (dic.ContainsKey(it))
                     dic[it]++;
                  else
                     dic.Add(it, 1);
            }
            sr.Close();

            // LINQ-запрос
            var sel = dic.Where(val=>val.Key == "King" || val.Key == "Edward")
                         .OrderBy(val=>val.Value);

            foreach (var it in sel)
               Console.WriteLine("{0}: {1}", it.Key, it.Value);

         }
         catch (Exception e)
         {
            Console.WriteLine(e.Message);
         }
         Console.ReadKey();
      }
   }
```

У цьому прикладі для вибірки з об'єкта відсортованого словника dic за допомогою вбудованого в клас SortedDictionary механізму LINQ-запитів вибирається нова колекція значень, що задовольняють критерію відбору, заданому за допомогою лямбда-виразу val=>val.Key == "King" || val.Key == "Edward"». Після чого відібрана колекція елементів сортується не за ключом, а частотою входження.

Результат роботи цієї програми наведено на рис. 3.

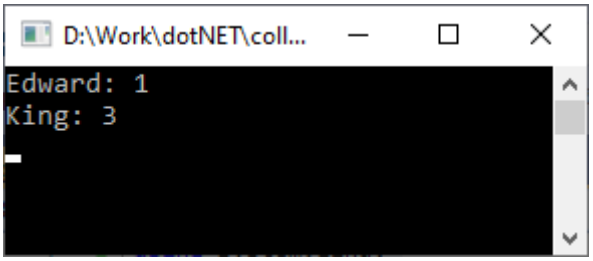

Рис. 3 – Результат LINQ-запиту

Більш детально із запитами LINQ можна ознайомитись за таким посиланням: https://docs.microsoft.com/ru-ru/dotnet/standard/linq/ .

3 .

1) Переробити наведену програму обробки файлів таким чином, щоб розділові знаки відокремлювалися від слів і додавалися в відсортований словник окремо.

2) За допомогою LINQ вивести інформацію про частоту використання у заданому тексті:

1 варіант – точок;

2 варіант – точок з комою;

3 варіант – знаки запитання;

4 варіант – знаків оклику;

5 варіант – тире.