

15-ma'ruza. Kolleksiylar

- ❖ ArrayList
- ❖ Ro'yxat List<T>
- ❖ List ni qanday yaratish mumkin?
- ❖ Dictionary

Kalit so'zlar: *dictionary, iterator, kolleksiya, link, lo'g'at, metod, navbat, ob'yekt, ro'yxat, sinf, stek, yield*

C# da o'zida bir xil tipdagi ob'yektlar to'plamini saqlovchi massivlar mavjud bo'lsada, lekin ular bilan ishlash har doim ham qulay emas. Masalan, massivlar cheklangan sondagi ob'yektlarni saqlaydi, lekin ba'zida qancha miqdordagi ob'yekt bilan ishlashga ehtiyojimiz borligini oldindan bilmaymiz. Bunday hollarda kolleksiyalardan foydalanish qulayroq. Kolleksiylarning yana bir afzallik tomoni, ularning ko'pchiligi ma'lumotlarning standart strukturalarini qo'llaydi, masalan, turli maxsus masalalarni yechishda foydali bo'lgan stek, navbat, lo'g'at kabilar.

Kolleksiya sinflarining katta qismi System.Collections (sodda to'ldirilmagan kolleksiyalar sinfi), System.Collections.Generic (kolleksiyalarning to'ldirilgan yoki tipiklashgan sinfi) va System.Collections.Specialized (kolleksiyalarning maxsus sinfi) nomlar fazosidan tarkib topadi. Shuningdek, vazifalarning parallel bajarilishini va ko'p tarmoqli kirishni ta'minlash uchun System.Collections.Concurrent nomlar fazosidan kolleksiyalar sinfi qo'llaniladi.

ArrayList

ArrayList sinfi ob'yektlar kolleksiyasini ifodalaydi. Agar turli tipdagi ob'yektlar – satr, son va hokazolarni birga saqlash zarurati bo'lsa, u holda aynan shu sinf mos keladi.

Sinfning asosiy metodlari:

int Add (object value): ro'yxatga value ob'yektini qo'shadi

void AddRange (ICollection col): kolleksiyada ishlatiladigan interfeys ICollection ni o'zida ifodalovchi col kolleksiyalar ob'yektlarini ro'yxatga qo'shadi.

void Clear (): ro'yxatdan barcha elementlarni o'chiradi

bool Contains (object value): ro'yxatda value ob'yekti mavjudligini tekshiradi.

Agar mavjud bo'lsa, true, aks holda false qaytaradi.

void CopyTo(Array array): joriy ro'yxatni array massiviga nusxalaydi.

ArrayList GetRange(int index, int count): index qiymatli indeksdan boshlab joriy ro'yxatning count elementlarini o'z ichiga olgan yangi ArrayList ro'yxatini qaytaradi.

int IndexOf(object value): value elementi indeksini qaytaradi.

void Insert(int index, object value): ro'yxatga index indeksi bo'yicha value ob'yektini joylashtiradi.

void InsertRange(int index, ICollection col): ro'yxatga index indeksdan boshlab ICollection kolleksiyasini joylashtiradi.

int LastIndexOf(object value): qiymat ob'ekti ro'yxatidagi oxirgi hodisa indeksini qaytaradi.

void Remove(object value): ro'yxatdan value ob'yektini o'chiradi.

void RemoveAt(int index): ro'yxatdan index indeksi bo'yicha elementni o'chiradi.

void RemoveRange(int index, int count): ro'yxatdan index indeksidan boshlab count elementlarini o'chiradi.

void Reverse(): ro'yxatni teskarisiga aylantiradi.

void SetRange(int index, ICollection col): index indeksidan boshlab col kolleksiyasi elementlarini ro'yxatga nusxalaydi.

void Sort(): kolleksiyani saralaydi.

Bundan tashqari Count xossasi yordamida ro'yxatdagi elementlar sonini olish ham mumkin.

Sinfning qo'llanilishini misolda qarab chiqamiz:

```
using System;
```

```
using System.Collections;
```

```
namespace Collections
```

```
{
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            ArrayList list = new ArrayList();
```

```
            list.Add(2.3); // ro'yxatga double tipli ob'yekt qo'shish
```

```
            list.Add(55); // ro'yxatga int tipli ob'yekt qo'shish
```

```
            list.AddRange(new string[] { "Hello", "world" });
```

```
            // ro'yxatga satrli massiv qo'shish
```

```
            foreach (object o in list)
```

```
            {
```

```
                Console.WriteLine(o);
```

```
            }
```

```
            list.RemoveAt(0); // birinchi elementni o'chirish
```

```
            list.Reverse(); // ro'yxatni teskarisiga aylantirish
```

```
            // indeks bo'yicha elementni olish
```

```
            Console.WriteLine(list[0]);
```

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

```
            {
```

```
                Console.WriteLine(list[i]);
```

```
            }
```

```
            Console.ReadLine();
```

```
        }
```

```
    }
```

}

ArrayList sinfi System.Collections nomlar sohasida joylashganligi sababli loyihaga uni qo'shishimiz talab etiladi (using System.Collections;).

Dastlab ixtiyoriy sinf ob'yektini yaratish konstruktori yordamida kolleksiya ob'yektini yaratamiz: ArrayList list = new ArrayList();. Shuningdek, ehtiyojga qarab, massivlardagi singari kolleksiyalarda ham boshlang'ich initsializatsiyani amalga oshirishimiz mumkin, masalan,

```
ArrayList list = new ArrayList(){1, 2, 5, "string", 7.7};
```

So'ngra navbati bilan turli qiymatlarni qo'shamiz. Mazkur sinf ham, ko'pchilik boshqa sinflar singari ikkita qo'shish usuliga ega: bitta ob'yektni qo'shish uchun Add metodidan, ob'yektlar to'plamini, masalan, massiv yoki boshqa kolleksiyalarni qo'shish uchun AddRange metodidan foydalanadi.

foreach sikli orqali ro'yxatning barcha ob'yektlaridan o'tib chiqishimiz mumkin. Mazkur kolleksiya faqatgina son yoki satr emas, balki turli turli ob'yektlarni saqlaganligi sababli, olinayotgan ob'yekt tipi sifatida object: foreach (object o in list) tipi olingan.

Ko'pchilik kolleksiyalar, shu jumladan, Arraylist ham, Remove/RemoveAt metodlari yordamida o'chirishni amalga oshiadi. Mazkur holatda biz o'chirilayotgan elemen indeksini RemoveAt metodiga uzatib birinchi elementni o'chirdik.

Yakunda biz kolleksiya elementlarini yana ekranga chiqardik, faqat bunda for sikli orqali.

Indeksator orqali biz kolleksiya elementlarini xuddi massivlardagi singari indeks bo'yicha olishimiz mumkin: object firstObj = list[0];

Ro'yxat List<T>

System.Collections.Generic nomlar sohasidagi List<T> sinfi bir tipdagi ob'yektlarning sodda ro'yxatini aks ettiradi.

Uning metodlari orasidan quyidagilarni ajratish mumkin:

void Add(T item): yangi elementni ro'yxatga qo'shish.

void AddRange(ICollection collection): ro'yxatga kolleksiya yoki massiv qo'shish.

int BinarySearch(T item): ro'yxatda elementni binar qidirish. Agar element topilsa, u holda metod shu elementning kolleksiyadagi indeksini qaytaradi. Bunda ro'yxat saralangan bo'lishi lozim.

int IndexOf(T item): ro'yxatdan birinchi topilgan element indeksini qaytaradi.

void Insert(int index, T item): ro'yxatga item elementini index pozitsiyasiga joylashtiradi.

bool Remove(T item): item elementini ro'yxatdan o'chiradi, agar o'chirish muvaffaqiyatli bo'lsa, u holda true qaytaradi.

void RemoveAt(int index): ko'rsatilgan index indeksli elementni o'chirish.

void Sort(): ro'yxatni saralash.

Ro'yxat qo'llanilishini misolda qarab chiqaylik:

```
using System;
```

```

using System.Collections.Generic;

namespace Collections
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> numbers = new List<int>() { 1, 2, 3, 45 };
            numbers.Add(6); // element qo'shish

            numbers.AddRange(new int[] { 7, 8, 9 });

            numbers.Insert(0, 666); // ro'yxatda 1-o'ringa 666 sonini joylashtiramiz

            numbers.RemoveAt(1); // ikkinchi elementni o'chiramiz

            foreach (int i in numbers)
            {
                Console.WriteLine(i);
            }

            List<Person> people = new List<Person>(3);
            people.Add(new Person() { Name = "Том" });
            people.Add(new Person() { Name = "Билл" });

            foreach (Person p in people)
            {
                Console.WriteLine(p.Name);
            }

            Console.ReadLine();
        }
    }

    class Person
    {
        public string Name { get; set; }
    }
}

```

Bu yerda ikkita ro'yxat yaratiladi: biri int tipli ob'yektlar uchun, boshqasi – Person ob'yektlari uchun. В первом случае мы выполняем начальную инициализацию списка: `List<int> numbers = new List<int>() { 1, 2, 3, 45 };`

List - elementlarni va obyektlarni saqlashga moslashgan to'plam. Bu to'plamga **List** deb nom berilishiga sabab, ro'yxatlar bilan qulay ishlash mumkin (tarjimasidan bilib olsa bo'ladi). Ro'yxatda nima qilishimiz mumkin, elementlarni ixtiyoriy joyga

qo'shish yoki ixtiyoriy joyidan o'chirib tashlash va tartiblash (sort) funksiyasi borligi bilan boshqa to'plamlardan ajralib turadi.

MUHIM QOIDALAR!

- List elementlari null qiymat qabul qilishi mumkin
- Elementlari qiymati bir xil bo'lishi mumkin
- Elementlari [0] indeksdan boshlanadi

List ni qanday yaratish mumkin?

1-Bosqich. Kod yuqorisida using System.Collections.Generic; ni kiritib o'tamiz

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

2-Bosqich. List shabloni yordamida List yaratamiz

3-Bosqich. Element qo'shamiz

Element qo'shishning ikki yo'li mavjud:

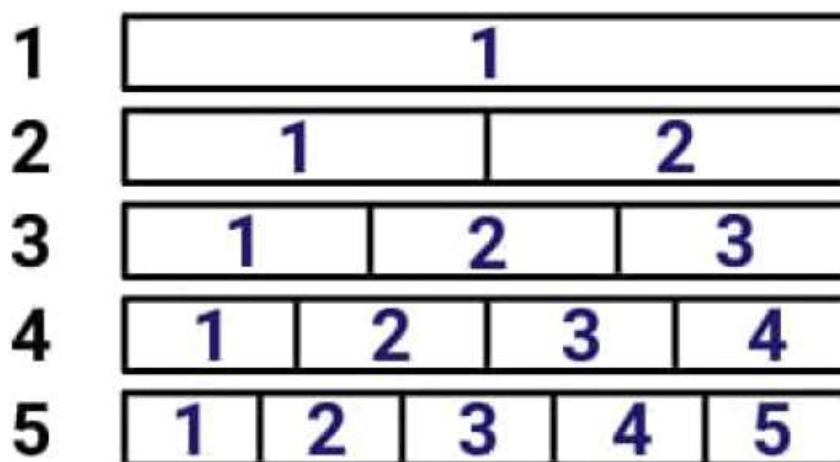
1) Xuddi massivdek, Listni yaratgan zahotingiz elementlarini kiritib qo'yishingiz mumkin

```
List<int> my_list = new List<int>() { 1, 2, 3, 4, 5 };
```

2) Add() va AddRange() funksiyasi yordamida ham element qo'shish mumkin
Add()

```
1 List<int> my_list = new List<int>();  
2 my_list.Add(1);  
3 my_list.Add(2);  
4 my_list.Add(3);  
5 my_list.Add(4);  
6 my_list.Add(5);
```

Listga element qo'shish



Elementlar shu tarzda birin ketin joylashib boradi

AddRange()

```
1 List<int> my_list = new List<int>();
2 my_list.Add(1);
3 my_list.Add(2);
4 my_list.Add(3);
5
6 int[] array = new int[2];
7 array[0] = 4;
8 array[1] = 5;
9
10 my_list.AddRange(array);
11
12 foreach(int value in my_list)
13     Console.WriteLine(value);
```

Microsoft Visual Studio Debug Console

```
1
2
3
4
5
D:\ConsoleApp11\ConsoleApp11\bin\Debug\net5.0\ConsoleApp11.exe (process 4600) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close when debugging stops.
Press any key to close this window . . .
```

Listda **foreach** ni quyidagi usulda ham ishlatish mumkin.

```
my_list.ForEach(a => Console.WriteLine(a));
```

Yoki oddiy usulda:

```

1  for (int a = 0; a < my_list.Count; a++)
2  {
3      Console.WriteLine(my_list[a]);
4  }

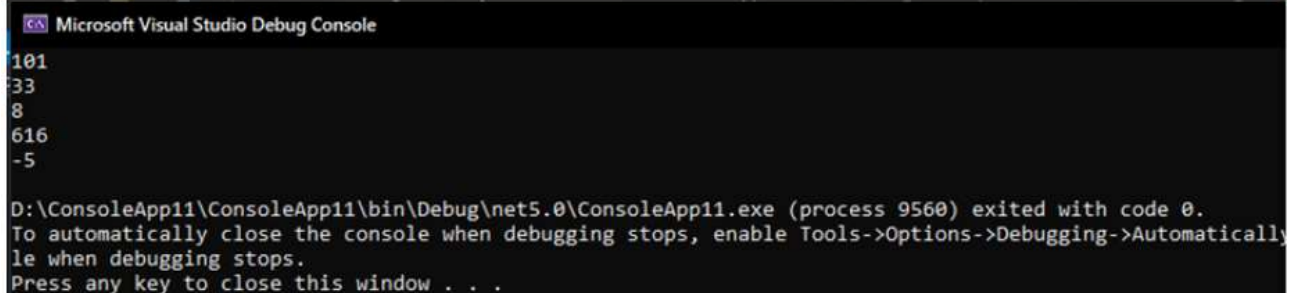
```

Listga misollar:

```

1  using System;
2  using System.Collections.Generic;
3  class Program
4  {
5      static void Main(string[] args)
6      {
7          List<int> my_list = new List<int>();
8          my_list.Add(101);
9          my_list.Add(33);
10         my_list.Add(8);
11         my_list.Add(616);
12         my_list.Add(-5);
13
14
15         my_list.ForEach(a => Console.WriteLine(a));
16     }
17 }

```



```

Microsoft Visual Studio Debug Console
101
33
8
616
-5

D:\ConsoleApp11\ConsoleApp11\bin\Debug\net5.0\ConsoleApp11.exe (process 9560) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically
close when debugging stops.
Press any key to close this window . . .

```

Dictionary

Dictionary "TKey, TValue" shablonidagi klass bo'lib, **System.Collection.Generics** namespace sida joylashgan. **Dictionary** kalitlar va qiymatlar to'plami bo'lib, boshqa to'plamlar kabi kalit/qiymat ko'rinishida element qabul qiladi. **Hashtable** dan farqi, **Dictionary** da qidirish tizimi tezroq

```

using System.Collections.Generic;


```

MUHIM QOIDALAR!

- Kalitlar bir xil bo'lmashligi kerak
- Kalit null qiymat qabul qilmaydi

Misol:

```
1 Dictionary<string, string> Mdic = new Dictionary<string, string>();
2
3     Mdic.Add("1", ".NET"); //1 juftliklar soni
4     Mdic.Add("2", "C#"); //2
5     Mdic.Add("3", "Asp.NET");//3
6     Mdic.Add("4", "LINQ"); //4
7
8     Console.WriteLine($"juftliklar soni : {Mdic.Count}");
9
10    foreach (var item in Mdic)
11    {
12        Console.WriteLine($"Kalit = {item.Key} Qiymat = {item.Value}");
13    }
```

 Microsoft Visual Studio Debug Console

```
juftliklar soni : 4
Kalit = 1 Qiymat = .NET
Kalit = 2 Qiymat = C#
Kalit = 3 Qiymat = Asp.NET
Kalit = 4 Qiymat = LINQ
```

```
1 Dictionary<string, string> Mdic = new Dictionary<string, string>();
2
3     Mdic.Add("1", ".NET"); //1 juftliklar soni
4     Mdic.Add("2", "C#"); //2
5     Mdic.Add("3", "Asp.NET");//3
6     Mdic.Add("4", "LINQ"); //4
7
8     Console.WriteLine($"juftliklar soni : {Mdic.Count}");
9
10    Dictionary<string, string>.KeyCollection KColl = Mdic.Keys;
11
12    foreach (var item in KColl)
13    {
14        Console.WriteLine($"kalit - {item}");
15    }
```



```
Microsoft Visual Studio Debug Console
juftliklar soni : 4
kalit - 1
kalit - 2
kalit - 3
kalit - 4
```

```
1 Dictionary<string, string> Mdic = new Dictionary<string, string>();
2
3     Mdic.Add("1", ".NET"); //1 juftliklar soni
4     Mdic.Add("2", "C#"); //2
5     Mdic.Add("3", "Asp.NET");//3
6     Mdic.Add("4", "LINQ"); //4
7
8     Console.WriteLine($"juftliklar soni : {Mdic.Count}");
9
10    Dictionary<string, string>.ValueCollection VColl = Mdic.Values
11
12    foreach (var item in VColl)
13    {
14        Console.WriteLine($"qiymat - {item}");
15    }
```

```
Microsoft Visual Studio Debug Console
juftliklar soni : 4
qiymat - .NET
qiymat - C#
qiymat - Asp.NET
qiymat - LINQ
```

```
1 static void Main(string[] args)
2 {
3     Dictionary<string, string> Mdic = new Dictionary<string, string>();
4
5     Mdic.Add("1", ".NET"); //1 juftliklar soni
6     Mdic.Add("2", "C#"); //2
7     Mdic.Add("3", "Asp.NET");//3
8     Mdic.Add("4", "LINQ"); //4
9
10    Console.WriteLine($"juftliklar soni : {Mdic.Count}");
11
12    Console.WriteLine("Clear.....");
13    Mdic.Clear(); // hamma elementlarni o`chirib yuboradi
14
15    Console.WriteLine($"juftliklar soni : {Mdic.Count}");
16 }
```

Microsoft Visual Studio Debug Console

```
juftliklar soni : 4  
Clear.....  
juftliklar soni : 0
```

D:\C#\Dictionary\Dictionary\bin\Debug\net5.0\Dictionary.exe (process 15636)
Press any key to close this window . . .

```
1  static void Main(string[] args)  
2      {  
3          Dictionary<string, string> Mdic = new Dictionary<string, string>();  
4  
5          Mdic.Add("1", ".NET"); //1 juftliklar soni  
6          Mdic.Add("2", "C#"); //2  
7          Mdic.Add("3", "Asp.NET");//3  
8          Mdic.Add("4", "LINQ"); //4  
9  
10         Console.WriteLine($"juftliklar soni : {Mdic.Count}");  
11  
12         foreach (var item in Mdic)  
13         {  
14             Console.WriteLine($"kalit - {item.Key} qiymat - {item.Value}");  
15         }  
16  
17         Console.WriteLine("*****");  
18  
19         Mdic.Remove("1"); //kaliti 1 ga teng bo'lgan juftlikni o'chirib  
20         Mdic.Remove("2"); //kaliti 2 ga teng bo'lgan juftlikni o'chirib  
21  
22         Console.WriteLine($"juftliklar soni : {Mdic.Count}");  
23  
24         foreach (var item in Mdic)  
25         {  
26             Console.WriteLine($"kalit - {item.Key} qiymat - {item.Value}");  
27         }  
28     }
```

Microsoft Visual Studio Debug Console

```
juftliklar soni : 4  
kalit - 1 qiymat - .NET  
kalit - 2 qiymat - C#  
kalit - 3 qiymat - Asp.NET  
kalit - 4 qiymat - LINQ  
*****  
juftliklar soni : 2  
kalit - 3 qiymat - Asp.NET  
kalit - 4 qiymat - LINQ
```

Foydalanish uchun tavsiya etiladigan adabiyotlar

1. Троелсен Эндрю, Джепикс Филипп. Язык программирования C# 7 и платформы .NET и .NET Core. Вильямс. 2018
2. Албахари Бен, Албахари Джозеф. C# 7.0. Справочник. Полное описание языка. Пер. с англ.-СПб: “Альфа-книга”, 2018, -1024 с.
3. Ю.С. Магда C#. Язык программирования Си Шарп. – Изд. ДМК Пресс, 2013, 190 с.
4. Лабор В.В. C#: Создание приложение для Windows. – Мн.: Харвест, 2003, 384 с.
5. <https://metanit.com/sharp/tutorial/4.3.php>

Mundarija

1-ma'ruza. Vorislik. Vorislikda konstruktorlarni ishlatish.	3
2-ma'ruza. Ichma-ich joylashgan sinflar.....	14
3-ma'ruza. Polimorfizm	21
4-ma'ruza. Abstrakt sinflar.....	29
5-ma'ruza. Interfeyslar	40
6-ma'ruza. Umumlashgan turlar.....	47
7-ma'ruza. System.Object asosiy tayanch sinfidan hosilaviy sinflarni yaratish.....	54
8-ma'ruza. Istisnolarni qayta ishlash.....	59
9-ma'ruza. Windows Forms ilovalarni yaratish.....	69
10-ma'ruza. Windows Forms ilovalari xossalari, sozlamalari.....	78
11-ma'ruza. Grafika xizmatlari	88
12-ma'ruza. Animatsiyalar.....	99
13-ma'ruza. Delegatlar. Lyambda ifodalar	116
14-ma'ruza. Hodisalar	127
15-ma'ruza. Kolleksiyalar	141