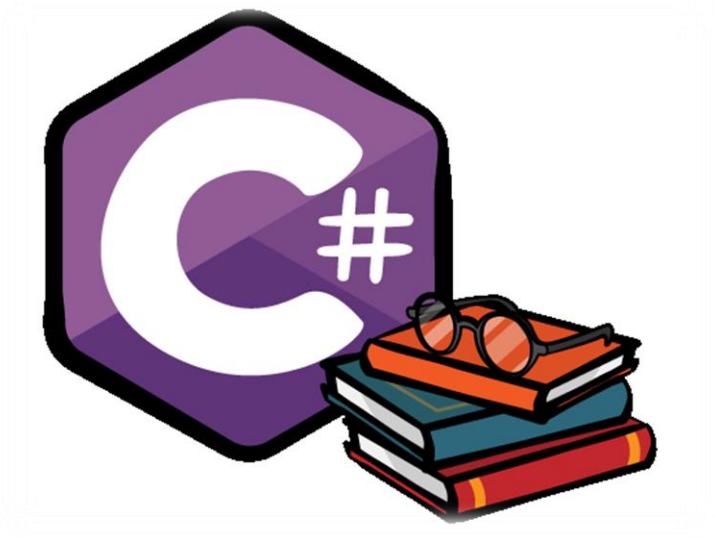

CONSOLE EKRANI-7

CONSOLE EKRANI

- ✓ *ArrayList Kullanımı*

KAZANIMLAR

- ✓ *ArrayList değişkenlerinin kullanımın öğrenilmesi*

C# ArrayList Kullanımı ve ArrayList Örnekleri

C # içinde ArrayList nedir?

ArrayList koleksiyonu C# içindeki dizilere benzer. ArrayList'in en büyük farkı dinamik bir yapıya sahip olmasıdır.

Diziler için dizinin dizi bildirimi sırasında tutabileceği öğelerin sayısını tanımlamanız gerekir. Ancak ArrayList koleksiyonunda, bunun önceden yapılması gerekmez. Öğeler, herhangi bir zamanda ArrayList koleksiyonuna eklenebilir veya koleysiyondan kaldırılabilir. ArrayList koleksiyonu için mevcut işlemlere daha ayrıntılı olarak bakalım

ArrayList Tanımlaması

Bir ArrayList beyanı aşağıda verilmiştir. ArrayList veritipinin yardımıyla bir ArrayList oluşturulur. "new" anahtar kelime ArrayList'in bir nesnesini oluşturmak için kullanılır. Nesne daha sonra a1 değişkenine atanır. Yani şimdi a1 değişkeni ArrayList'in farklı öğelerine erişmek için kullanılacaktır.

```
ArrayList a1 = new ArrayList();
```

ArrayList'e eleman ekleme

add methodu: ArrayList öğesine bir eleman eklemek için kullanılır. Ekleme yöntemi dizi listesine herhangi bir tür veri tipi elemanı eklemek için kullanılabilir. Böylece dizi listesine bir Tamsayı, bir String veya bir Boolean değeri ekleyebilirsiniz. Ekleme yönteminin genel sözdizimi aşağıda verilmiştir.

Dinamik Boyut

Standart diziler sabit boyutludur; programlama aşamasında dizinin boyutu belirtilir ve programın çalışması sırasında değiştirilemez. ArrayList ise değişken boyutludur. Eleman ekleme ve çıkarma durumuna göre boyutu dinamik olarak değişmektedir.

```
static void Main(string[] args)
    {
        ArrayList DinaimikDizi = new ArrayList();
        for (int i = 0; i < 9; i++)
            DinaimikDizi.Add(i);


    }
```

ArrayList'in başlangıç kapasitesi 4'tür. Mevcut kapasite yeterli gelmediğinde, arkaplanda kapasite 2 katına çıkarılmaktadır. Yukarıdaki örneği inceleyecek olursak; for döngüsü ile 0'dan 8'e kadar olan sayılar koleksiyonumuza eklenmektedir. Başlangıç kapasitesi 4 olan koleksiyona, 5. sayı eklenmek istenildiğinde, koleksiyon kapasitesini 2 katına çıkararak 8 yapmaktadır. Aynı şekilde 9. sayı ekleneceği zaman mevcut kapasite tekrar 2 katına çıkarılıp 16 yapılmaktadır.

Metotlar ve Özellikler

Nesne Oluşturma

ArrayList sınıfından bir nesne oluşturuyoruz. Bu nesne, koleksiyondaki nesnelerin referanslarını tutacaktır.

```
ArrayList DinamikDizi = new ArrayList();
```

Add(object x) Metodu

Add() metodu ile veri tipi fark etmeksizin koleksiyon içerisine her türlü öğeyi ekleyebiliriz.

```
DinamikDizi.Add("Serdar");
```

Remove(object x) Metodu

Remove() metoduna parametre olarak girilen ifade koleksiyon içerisinden silinir.

```
DinamikDizi.Remove("Serdar");
```

Clear() Metodu

Clear() metodu koleksiyon içerisindeki tüm öğeleri silmektedir.

```
DinamikDizi.Clear();
```

Count Özelliği

Koleksiyon içerisinde yer alan elemanların sayısını döndürmektedir.

```
int ElemanSayisi=DinamikDizi.Count;
```

Capacity Özelliği

Koleksiyonun kapasitesinin döndürmektedir.

```
int Kapasite = DinamikDizi.Capacity;
```

Contains(object x) Metodu

Koleksiyon içerisinde parametre olarak girilen öğeyi arar. Öğe bulunursa TRUE, bulanamazsa FALSE döndürür.

```
bool varmi = DinamikDizi.Contains(456);
```

Sort() Metodu

Koleksiyon içerisindeki öğeleri artan sırada sıralamaktadır. Sıralama işleminin gerçekleşebilmesi için öğelerin karşılaştırılabilir olması gerekmektedir.

```
DinamikDizi.Sort();
```

BinarySearch(object x) Metodu

Koleksiyon içerisinde parametre olarak girilen değeri arar. Değer bulunursa değerin indeks numarasını döndürür. Eğer değer bulunamaz ise negatif bir değer döndürür. Bu metodu kullanabilmek için önce Sort() metodu ile koleksiyonu sıralamamız gerekmektedir.

```
int indeks=DinamikDizi.BinarySearch(18);
```

CopyTo(Array ar, int i) Metodu

Koleksiyon içerisindeki tüm öğeleri, ar dizisine i.indeksinden başlayarak kopyalayacaktır. ar dizisi koleksiyondaki öğelerin tipiyle uyumlu olmalıdır.

```
string[] Dizi = new string[20];

DinamikDizi.CopyTo(Dizi, 2);
```

ToArray() Metodu

Koleksiyonda yer alan elemanların kopyasını içeren bir dizi döndürür. Belirli işlemler de daha hızlı işlem süreleri elde etmek için bu yöntem kullanılabilir.

```
int[] Dizi = new int[20];

Dizi =(int[]) DinamikDizi.ToArray(typeof(int));
```

TrimToSize() Metodu

Koleksiyonun kapasitesini, eleman sayısına eşitler.

```
DinamikDizi.TrimToSize();
```

Reverse() Metodu

Koleksiyonun içeriğini ters çevirir.

```
DinamikDizi.Reverse()
```

RemoveRange(int i, int c) Metodu

Koleksiyon içerisinde, i. indeksten başlayarak c adet elemanı çıkarır/siler.

```
DinamikDizi.RemoveRange(1, 3);
```

AddRange(Koleksiyon c)

c koleksiyonunu, çağrıda bulunulan koleksiyonun sonuna ekler.

```
ArrayList Ek= new ArrayList();

DinamikDizi1.AddRange(Ek);
```

InsertRange(int i, Koleksiyon c)

c koleksiyonunun tüm elemanlarını, çağrıda bulunulan koleksiyonun i. indeksinden itibaren eklemeye başlar. Çağrıda bulunulan koleksiyonun elemanları silinmez, sadece c koleksiyonunun tüm elemanlarına yer açacak şekilde ötelenir.

```
ArrayList Yeni=new ArrayList();

DinamikDizi1.InsertRange(1, Yeni);
```

LastIndexOf(object x) Metodu

Parametre olarak girilen öğenin rastlanıldığı son konumun indeksinin döndürür. Değer mevcut değilse -1 döndürür.

```
int indeks=DinamikDizi.LastIndexOf("serdar");
```

GetRange(int i, int c) Metodu

Çağrıda bulunulan koleksiyonun i.indeksinden başlayarak c adet elemanını içeren bir nesne döndürür. Döndürülen nesne, çağrıda bulunan nesne ile aynı elemanlara referansta bulunur.

```
ArrayList Parca = DinamikDizi1.GetRange(2, 3);
```

SetRange(int i, Koleksiyon c)

c koleksiyonunun tüm elemanlarını, çağrıda bulunulan koleksiyonun i. indeksinden başlayarak üzerine yazar.

```
DinamikDizi.SetRange(2, YeniDizi);
```

Örnek: *Koleksiyondan Öğeleri Çekelim*

```
ArrayList DinamikDizi = new ArrayList();
            // Koleksiyona 128,18,56 değerlerini ekledik.
            DinamikDizi.Add(128);
            DinamikDizi.Add(18);
            DinamikDizi.Add(56);
            // foreach ile koleksiyonun tüm elemanlarını ekrana yazdırdık.
            foreach (var i in DinamikDizi)
                Console.WriteLine(i);
            // 18 değerini koleksiyondan çıkardık.
            DinamikDizi.Remove(18);
            // Bu sefer for döngüsü ile koleksiyonun tüm elemanlarını ekrana yazdırdık.
            for (int i = 0; i < DinamikDizi.Count; i++)


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

Örnek: *Koleksiyonda Arama Yapalım*

```
  ArrayList DinamikDizi = new ArrayList();
            // Koleksiyona 128,18,56 değerlerini ekledik.
            DinamikDizi.Add(128);
            DinamikDizi.Add(18);
            DinamikDizi.Add(56);
            /* 56 değeri koleksiyonda mevcut olduğu için Contains() metodu TRUE *
döndürecek ve ekrana "Mevcut." yazılacaktır. */
            if (DinamikDizi.Contains(56))
                Console.WriteLine("Mevcut.");
            else


                        Console.WriteLine("Mevcut Değil.");
```

Örnek: *Herhangi Bir Öğenin İndeks Numarasını Bulalım*

```
    ArrayList DinamikDizi = new ArrayList();
            // Koleksiyona 128,18,56 değerlerini ekledik.
            DinamikDizi.Add(128);
            DinamikDizi.Add(18);
            DinamikDizi.Add(56);
```

```
// Koleksiyon içerisindeki öğeleri artan sırada sıraladık. (18-56-128)
// NOT: BinarySearch() metodunu kullanabilmek için sıralama yapmamız gerek.
DinamikDizi.Sort();
// 18 değerinin indeks numarasını ekrana yazdırdık. (İndeksi:0)
Console.WriteLine(DinamikDizi.BinarySearch(18));
```

Örnek : *10 adet sayıyı tek ve çift olarak dolduran ArrayList Örneği*

```
static void Main(string[] args)
            {
                ArrayList tekSayilar = new ArrayList();
                ArrayList ciftSayilar = new ArrayList();
                int ortalama = 0, toplam = 0;
                Random rnd = new Random();
                for (int i = 0; i < 10; i++)
                {
                    int sayi = rnd.Next(1, 100);
                    Console.Write(sayi + "   ");
                    toplam += sayi;
                    if (sayi % 2 == 0)
                    {
                        ciftSayilar.Add(sayi);
                    }
                    else
                    {
                        tekSayilar.Add(sayi);
                    }

                }
                Console.WriteLine();

Console.WriteLine("============================================================");
                ortalama = toplam / 10;

                Console.WriteLine("Sayıların Ortalaması : {0}", ortalama);
                Console.WriteLine("Tek Sayıların Adeti : {0}", tekSayilar.Count);
                Console.WriteLine("Çift Sayıların Adeti : {0}", ciftSayilar.Count);
                Console.ReadKey();
            }
```

Dikkat : *Bu kod ile **karmaList** isimli ArrayList nesnemizin içinde **string, int, bool, float** ve **char** tiplerinden oluşan verileri aynı anda saklarız.*

*Hatırlarsanız, ArrayList sınıfının bize sunduğu özelliklerden biri olan **Reverse()** metodudundan bahsetmiştik. ArrayList dizimizin içerisindeki elemanları ters çevirme işlemi için **bu** metodu kullanıyorduk. **Reverse()** metodunu ve ArrayList'lerde nasıl birden farklı türdeki elemanları kullanacağımızı bir örnekle inceleyelim:*

```
using System;
using System.Collections;

class name
{
    static void Main(string[] args)
    {
        // karmaList isimli ArrayList nesnesi oluşturalım.
        ArrayList karmaList = new ArrayList();

        // karmaList'e değişik tipte elemanlar ekliyoruz.
        karmaList.Add("Ali");
        karmaList.Add(23);
        karmaList.Add(false);
        karmaList.Add(52.8d);
        karmaList.Add('r');

        // karmaList'in elemanlarını ekrana yazdırıyoruz:
        Console.WriteLine("\t karmaList'in elemanları:");
            foreach (object eleman in karmaList)
            Console.WriteLine(eleman);

        karmaList.Reverse();         // karmaList'imizi ters çeviriyoruz.

        Console.WriteLine("\n ---> karmaList'in elemanları ve türleri <---");
            foreach (object eleman in karmaList)
            Console.WriteLine("Türü: {0,15} değeri: {1,7}", eleman.GetType(), eleman);
             Console.ReadLine();
    }
}
```