BİLGİSAYAR PROGRAMLAMA-II- 3.ders

DİZİLER

Diziler bir çok bilgiyi tek bir değişken içerisinde tutmamızı sağlayan ifadelerdir. Dizide tutulan bilgiler Ram da tutulur. Elektrikler kesildiğinde dizideki bilgilerde kaybolacaktır. C# da dizi tanımlama iki şekilde olmaktadır. Bunlar boyutlu dizi tanımlama ve boyutsuz dizi tanımlamadır.

Bilgiler diziye her eklendiğinde dizinin sıfırlanmaması için tanımlamaları Globalde (alt yordamların en üstünde) yapmak gerekir. Dizideki ilk eleman her zaman [0] sıfır indisi ile tutulur. Dolayısı ile diziye bilgi eklerken yada okuturken ilk eleman sıfırıncı eleman olmalıdır.

Dizinin içinde kaç eleman olduğunu bilmiyorsak, dizideki eleman sayısınca döngünün dönmesini istiyorsak **foreach()** döngüsü kullanmak gerekir. Bu döngünün yapısı şu şekildedir.

Foreach Döngüsü

Bu döngü diziler için kullanımı kolay bir döngüdür. Eğer bir dizideki tüm elemanlar üzerinde işlem yapmak istiyorsak ve dizinin eleman sayısını bilmiyorsak kullanabiliriz. Döngü her döndüğünde diziden sırayla okunan eleman bir değişkene atılır ve döngü içinde de bu değişken kullanılır.

```
foreach (string Eleman in Dizi)
{
      listBox1.Items.Add(Eleman);
}
```

Boyutlu Dizi Tanımlama

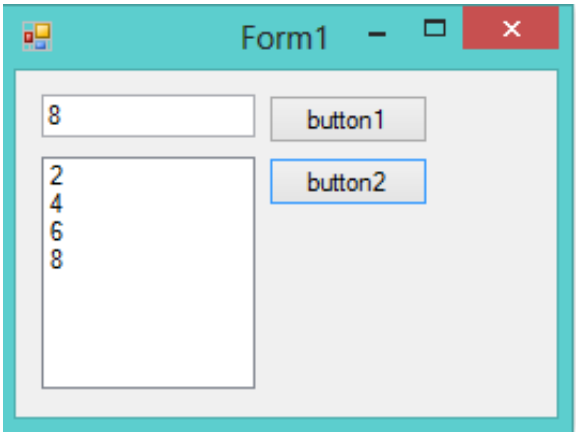
Bu tanımlamada dizinin eleman sayısı ve tipi belirlenmelidir. Örnek tanımlama şekli aşağıdaki gibidir.

```
int [] Dizi = new int[100];
```

Dizideki elemanlar okunurken dizinin boyutu bilindiği için for() döngüsü kullanılabilir. Tabiki diziler için en kullanışlı döngü foreach() döngüsüdür. Bu döngüyü kullanmak daha çok tercih edilmelildir.

Örnek

Şekildeki gibi bir Form üzerine 2 buton, 1 Texbox, 1 Listbox ekleyin. Textbox'a girilen sayıları birinci butona tıklayınca diziye eklesin. Daha sonra ikinci butona tıklandığında tüm bilgiler Diziden okunup ListBox a eklensin.

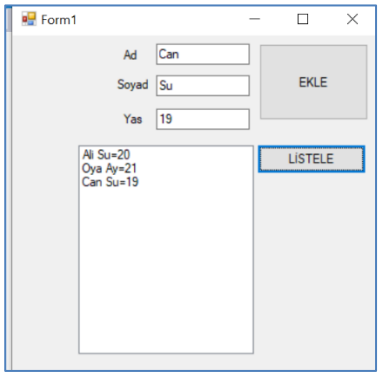



```
using System.Collections;
int [] Dizi1 = new int[100];
int i=0;

private void button1_Click(object sender, EventArgs e)
```

```
{
        i++;
        Dizi1[i] =Convert.ToInt32(textBox1.Text);
}

private void button2_Click(object sender, EventArgs e)
{
        foreach (string eleman in Dizi1)
        {
                try
                {
                listBox1.Items.Add(eleman);
                }
                catch { }
        }
}
```

Çok Boyutlu Dizi Kullanımı

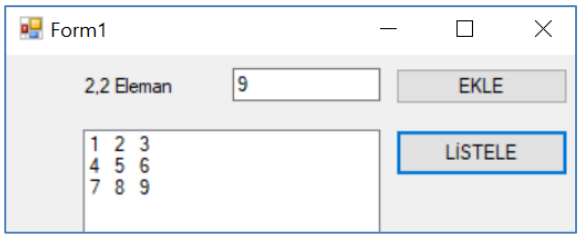
Örnek 1:




```
string[,] Dizi = new string[100, 3];
int i = 0;

//EKLEME**************
private void button1_Click(object sender, EventArgs e)
{
    string Ad = txtAd.Text;
    string Soyad = txtSoyad.Text;
    string Yas = txtYas.Text;

    Dizi[i, 0] = Ad;
    Dizi[i, 1] = Soyad;
    Dizi[i, 2] = Yas;

    i++;
}

//LİSTELEME*************
private void button2_Click(object sender, EventArgs e)
{
    int KisiSayisi = i;
```

```
    for(int k=0; k<KisiSayisi; k++)
    {
        listBox1.Items.Add(Dizi[k,0] + " " +  Dizi[k,1] + "=" + Dizi[k,2]);
    }
}
```

Örnek 2;




```
int[,] Dizi = new int[10, 3]; //10 tane 3 sütunluk bilgi eklenebilir.
int i = 0;
int j = 0;

//EKLEME**************
private void button1_Click(object sender, EventArgs e)
{
  lblMatrisElemani.Text = i + "," + j + " Eleman";

  int Sayi = Convert.ToInt32(txtSayi.Text);

  Dizi[i, j] = Sayi;

  j++;

  if(j==3)
  {
    i++;
    j = 0;
  }
}

//LİSTELEME*************
private void button2_Click(object sender, EventArgs e)
{
  int SatirSayisi = i;

  for(int k=0; k<SatirSayisi; k++)
  {
    listBox1.Items.Add(Dizi[k,0] + "    " +  Dizi[k,1] + "    " + Dizi[k,2]);
  }
}
```

Boyutsuz Dizi Tanımlama (ArrayList Dizisi)

Bu dizide dizinin boyutu ve tipini belirlemeye gerek yoktur. Normalde dizilerde tüm elemanlar tanımlanan tipde olmak zorundadır. Fakat bu dizi tanımlamasında farklı tipleri (string, int vs) aynı dizi içerisinde tutmak mümkündür. Her bir dizi hücresinin içerisinde farklı boyutta elemanlar bulunabilir. Bu elemanlar okunup değişkenlere atılırken tiplerin kontrolü burada önemli olacaktır.

ArrayList komutu ile dizi tanımlayabilmek için aşağıdaki kütüphanenin sayfaya eklenmiş olması gerekir.

```
using System.Collections;
```

Dizinin tanımlaması aşağıdaki gibi yapılır.

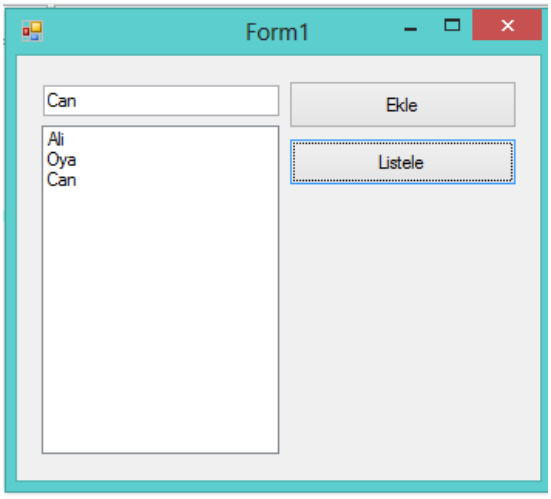
```
ArrayList Dizi = new ArrayList();
```

ArrayList ile ilgili olarak kullanabileceğimiz bazı komutlar şunlardır.

```
Dizi.Add(TextBox1.Text);//Diziye eleman ekler
Dizi[i].ToString();     //Diziden okumayı sağlar
Dizi.Clear();           //dizi içerisindeki tüm elemanları siler.
Dizi.Count;             //dizinin eleman sayısını verir.
Dizi.RemoveAt(3);       //indis numarası 3 olan elemanı Diziden siler.

Dizi[3];          //indis numarası 3 olan elemanı getirir.
```

Örnek

Yukarıdaki aynı örneği boyutsuz dizi (ArrayList) kullanarak yapın.

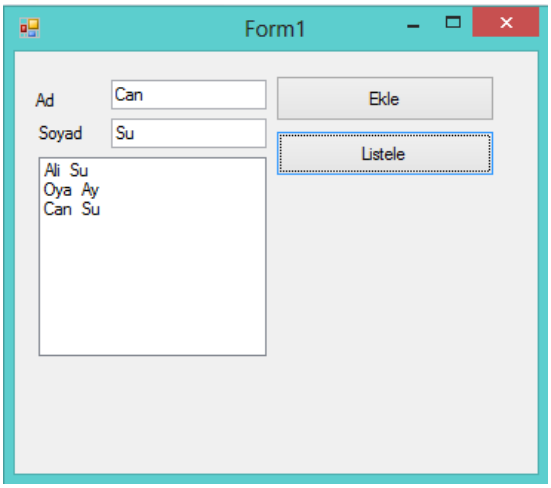



```
ArrayList Dizi = new ArrayList();

private void button1_Click(object sender, EventArgs e)
{
      Dizi.Add(textBox1.Text);
}

private void button2_Click(object sender, EventArgs e)
{
      foreach (string Eleman in Dizi)
      {
            listBox1.Items.Add(Eleman);
      }
}
```

Örnek

Şekildeki gibi bir 2 Textbox dan Kişilerin Ad ve Soyad bilgilerini alın. Ekle butonuna tıklandığında her seferinde bu bilgileri tek boyutlu bir diziye eklesin. Ardından Listele butonuna tıklandığında kişilerin Ad ve Soyadlarını ListBox'da görüntülesin.

1. Yöntem




```
ArrayList DiziAd = new ArrayList();
ArrayList DiziSoyad = new ArrayList();

private void button1_Click(object sender, EventArgs e)
{
      DiziAd.Add(textBox1.Text);
      DiziSoyad.Add(textBox2.Text);

}

private void button2_Click(object sender, EventArgs e)
{
      for (int i = 0; i < DiziAd.Count; i++)
      {
            listBox1.Items.Add(DiziAd[i] + "   " + DiziSoyad[i]);
      }
}
```

LİSTELER , List<Tip>

Dizilerde bulunan bir çok dezavantajı (Boyut ve tip belirleme zorluklarını) gidermek için List nesneleri kullanılabilir. List ile ArrayList arasındaki en büyük fark Tip Tanımlama zorunluluğudur. Arraylist içerisine hem int hemde string yada başka değişkenler alabilir. Fakat List ler ise tek bir tane tip alabilir. Yine burda da her eleman eklendikçe hafızada yeni yer açılır. Böylelikle boyut tanımlama zorunluluğu olmamış olur. Tip tanımlama olduğu için Hafıza kullanımı diğerlerine göre yine avantajlı olmuş oluyor. Uygulamalarınızda bunu kullanmanız tavsiye edilir.

Bu komutun çalışabilmesi için

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

kütüphanesinin programa eklenmiş olması gerekir. Varsayılan olarak zaten eklenmiş durumda çıkar. İnternet programcılığında eklemek gerekir.

Bir listeyi tanımlarken

```
List<int> sayilar = new List<int>();
```

Yada string tanımlayacaksak;

```
List<string> isimler = new List<string>();
```

Listeye değer eklerken kullanımı şu şekildedir.

```
sayilar.Add(11);
sayilar.Add(32);
sayilar.Add(213);
sayilar.Add(819);
sayilar[3]; //dizideki 3. Nolu gözdeki elemanı getirir.
```

Listede kaç adet eleman olduğunu almak için;

```
sayilar.Count
```

Listenin içindeki bilgileri okumak için şu döngü kullanılabilir.

```
foreach (int sayi in sayilar)
{
listBox1.Items.Add(sayi.ToString());
}
```

Listeden bir elemanı değeri ile çıkarmak için şu ifade kullanılır. Burada değeri 213 olan sayı listeden çıkarılmaktadır.

```
sayilar.Remove(213);  //değeri 213 olan elemanı siler.
sayilar.RemoveAt(2);  //sıra numarası 2 olan elemanı siler. Komut "RemoveAt"
```

Listede elemanın bulunup bulmadığını bulma.

```
if (sayilar.Contains(819))
{
MessageBox.Show( "819 sayisi listede vardır");
}
```

Listede kaçıncı sırada olduğunu bulmak için aşağıdaki uygulama kullanılabilir.

```
int SiraNo = sayilar.BinarySearch(213);
MessageBox.Show(SiraNo.ToString()); //SiraNo=2 olarak verir.
```

Diziyi Listeye çevirmek için aşağıdaki uygulama kullanılabilir.

```
string[] dizi = new string[3];
dizi[0] = "Ali";
dizi[1] = "Oya";
dizi[2] = "Can";

List<string> isimler = new List<string>(dizi);

foreach (string isim in isimler)
{
listBox1.Items.Add(isim);
}
```