Masaüstü Uygulama Geliştirme

Hafta-6

Öğr. Gör. Zafer SERİN

Diziler(Arrays)

- Diziler içerisinde birden fazla aynı türde değer tutabilen yapılardır.
- Prosedürel programlamanın temel veri kümeleridir. Yani yazılımsal boyutta, yazılım adına RAM'de birden fazla değeri tek bir değişken altında bir veri kümesi halinde tutabilirler.
- Diziler veri kümeleri oldukları için, içlerindeki birden fazla veri üzerinde kümesel işlemler yapmamızı sağlayabilirler.
- Diziler, prosedürel programlamanın temel yapıları oldukları için daha gelişmiş yapılar olan koleksiyonlarında temelini oluştururlar ve gelişmelerine katkıda bulunurlar.

Diziler(Arrays)

- Diziler referans türlü değerlerdir. Yani nesnel yapılardır. Özlerinde classtırlar.
- Yapısal olarak RAM'de Heap'te tutulurlar.
- Dizi içerisine her türlü değer koyulabilir.(Değer türlü-referans türlü)
- Diziler içerisine koyulan değerler işlevsel olarak aynı mahiyette olmalıdır.
- Diziler içerisine eleman/değer eklendikçe bunları rastgele bir şekilde depolamaz! Düzenli bir şekilde sıralı depolayacaktır.
- Dizilerde eklenen elemanlar index(indis) ismini verdiğimiz numaralarla ardışık bir şekilde numaralandırılırlar.

Diziler(Arrays)

- İndex(indis) numarası her bir elemanı verilen ardışık sayıdır ve 0'dan başlayıp n-1'de sonlanır. n burada dizinin eleman sayısıdır.

type[] isim = new type[eleman_sayisi];

- Dizi tanımlama sürecinde eleman sayısı mecburi girilmek zorundadır. Yani dizide çalışacak değer adeti başta bildirilmelidir. Bu çok önemli bir sınırlılıktır.
- Bir dizi tanımlaması yapıldığı an bellekte o diziyi kullansakta kullanmasakta verilen eleman sayısı kadar alan tahsisinde bulunulur.

Diziler(Arrays)

- Kullanılmadığı halde diziler direkt olarak bellekten belirtilen eleman sayısı kadar alan tahsisinde bulunduğu için bu bir sınırlılıktır.
- Diziler alan tahsisi yapıldığında ilgili alanlara türüne uygun default(varsayılan) değerleri atarlar.

Diziler’de Değer Atama ve Değer Okuma

- Diziler’de değer atama ve okuma işlemleri index’ler üzerinden sağlanır.

```
int[] sayilar = new int[10];
sayilar[3] = 5;
Console.WriteLine(sayilar[3]);
```

- Diziler’de değer atama ve değer okuma işlemleri yapılırken dizinin sınırı aşılmamalıdır!
- Diziler’de değer ataması yapılırken sıralamayı göz önünde tutmak zorunda değiliz.

Diziler’de Değer Atama ve Değer Okuma

- Diziler’de eleman sayısı başlangıçta belirlendiği için duruma göre fazladan değer atayamayız. Bu bir sınırlılıktır.

For-each Iterasyonu

- C#'ta dizilerin her bir elemanına tek tek erişmek için for-each iterasyonu kullanılabilir.
- Örnek kullanım:

```
int[] elemanlar = { 11, 22, 33 };
foreach (var item in elemanlar)
{
    Console.WriteLine(item);
}
```

Metotlar

- Metotlar bir programın en küçük işlev birimi olarak düşünülebilir ve program içerisinde operasyonlar gerçekleştirmemizi sağlayan yapılardır.
- Metotlar sayesinde kod tekrarlarının önüne geçilebilir ve 1 kez yazılan metot pek çok yerde aynı işlevsellik ile kullanılabilir.
- Matematikteki fonksiyonlara oldukça benzemektedir.
- Aralarında ufak farklar olmasına karşın fonksiyon, metot ve yordam birbirine oldukça benzer yapılardır.

Genel Metot Tanımı

- Şu ana kadar gördüğümüz Main bir metottur.
- Metotlar classların veya structların içerisinde tanımlanabilir.
- Metot içinde metot tanımlanamaz(Local functionslar hariç!)
- Genel metot tanımı şu şekildedir:

[erişim belirleyicisi] [geri dönüş değeri] [metot adı](parametreler)//İmza
{
//Gövde
}

Yapılacak İşleme Göre Metotlar

- Yapılacak işleme(işlevine) göre 4 farklı türde metot oluşturulabilir. Bunlar:

1. Geriye Değer Döndürmeyen ve Parametre Almayan Metot
2. Geriye Değer Döndürmeyen ve Parametre Alan Metot
3. Geriye Değer Döndüren ve Parametre Almayan Metot
4. Geriye Değer Döndüren ve Parametre Alan Metot

Geriye Değer Döndürmeyen ve Parametre Almayan Metot

- Geriye değer döndürmeyen metotların geri dönüş değeri yoktur ve bunu belirtmek için ‘void’ anahtar sözcüğü kullanılır. Parametre almadığı için yaylı parantezlerin() arası boş bırakılır. Örneğin:

```
public void SelamYaz()
{
    Console.WriteLine("Selam");
}
```

Geriye Değer Döndürmeyen ve Parametre Alan Metot

- Geriye değer döndürmeyen metotların geri dönüş değeri yoktur ve bunu belirtmek için ‘void’ anahtar sözcüğü kullanılır. Parametre alan olduğu için yaylı parantezler arasında alınacak parametreler belirtilir. Örneğin:

```
public void Selamla(string isim, int yas)
{
    Console.WriteLine($"Merhaba {isim}, sen {yas} yaşındasın");
}
```

Geriye Değer Döndüren ve Parametre Almayan Metot

- Geriye değer döndüren metotların geri dönüş tipi metot imzasında belirtilmelidir. Metodun içerisinde de ‘return’ anahtar sözcüğü ile ilgili değer geriye döndürülmelidir. Parametre almadığı için yaylı parantezler arası boş bırakılır. Örneğin:

```
public int Topla()
{
    return 3 + 5;
}
```

Geriye Değer Döndüren ve Parametre Alan Metot

- Geriye değer döndüren metotların geri dönüş tipi metot imzasında belirtilmelidir. Metodun içerisinde de ‘return’ anahtar sözcüğü ile ilgili değer geriye döndürülmelidir. Parametre aldığı için yaylı parantezler arasına parametreler yazılmalıdır. Örneğin:

```
public int Carp(int sayi1, int sayi2)
{
    return sayi1 * sayi2;
}
```

Metotların Çağırılması

- Metotlar oluşturulduğu zaman kullanılacakları yerlerde çağırılmaları gerekmektedir. Değer döndüren metotlar tiplerine uygun değişkenlere atanarak istenildiği zaman kullanılabilir. Bazı metot çağırma örnekleri şu şekildedir:

```
SelamYaz();
Selamla("Zafer", 27);
int _topla = Topla();
int _carp = Carp(3, 4);
```