PROGRAMLAMA DİLLERİ I

Öğr. Gör. M. Mutlu YAPICI

Ankara Üniversitesi
Elmadağ Meslek Yüksekokulu

Öğr. Gör. M. Mutlu YAPICI

Ders İzlencesi

Hafta	Modüller/İçerik/Konular
1. Hafta	
2. Hafta	
3. Hafta	
4. Hafta	
5. Hafta	
6. Hafta	
7. Hafta	
8. Hafta	
9. Hafta	
10. Hafta	
11. Hafta	
12. Hafta	
13. Hafta	
14. Hafta	

Döngüler

Döngüler bir program içerisinde belirli işlerin defalarca yapılmasını sağlayan komut bloklarıdır. Sonsuz döngüler yapılabildiği gibi belirli kriterler sağlanana kadar devam eden döngüler de yapılabilir.

4 tip döngü vardır. Bunlar:

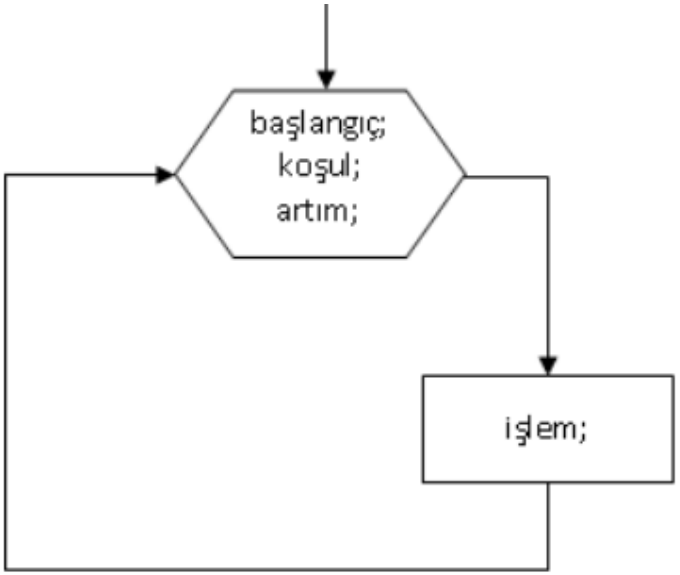
- for döngüleri
- while döngüleri
- do while döngüleri
- foreach döngüleri' dir.

For Döngüsü

Belirlenen başlangıç değerinden itibaren belirtilen koşul sağlanana kadar içine yazıldığı kod parçasını ardı ardına çalıştıran bir döngü çeşididir.

Yapısı:

```
for(başlangıç; koşul; artım)
{
    yapılacak işler;
}
```



Başlangıç, döngü kontrol değişkeni olarak da ifade edilebilir. Döngü içerisinde bir sayaç görevini görür.

Koşul, döngünün ne kadar çalışacağını denetleyen mekanizmadır. Koşul sağlanıyorken döngü çalışmaya devam eder. Koşul sağlanmadığında ise döngü durur. Koşulda genellikle başlangıç değerinin durumu denetlenir.

Artım, başlangıç değerinin döngünün her adımda artma ya da azaltma miktarını belirler. Eğer başlangıç değeri hiç değişmez ise sonsuz döngü oluşur.

For Döngüsü Örnekleri

Örnek 2-1: 1'den 10'a kadar olan sayıları ekrana yazdırınız.

```
byte i;
for(i=1;i<=10;i++)
{
    Console.WriteLine(i);
}
```

Yukarıdaki kodu incelediğimizde;

- **Döngü kontrol değişkenimiz olan i'ye 1 değerini atayarak başlangıç değerimizi,**
- **Döngümüzün ne zamana kadar döneceğini belirlediğimiz koşulumuzu i<=10 ifadesini,**
- **i++ ile de i değerimizi döngümüzün her dönüşünde 1 arttıracağımızı**

belirliyoruz.

For Döngüsü Örnekleri

Örnek :10'dan 0'a geriye doğru sayan ve sayıları ekrana yazdıran programı yazdırınız.

For Döngüsü Örnekleri

Örnek :Klavyeden girilen 10 sayıdan en büyüğünü bulan C# programını yazınız.

Cevap: En büyük sayıyı bulmak için girilen sayıları bir referans değer ile karşılaştırmamız gerekir. Başta sıfır olarak aldığımız referans değer kendinden daha büyük bir değer ile karşılaştığında yeni referans değerimiz bu sayı olacaktır. İşlem sonunda referans değer yazdırılacak

```
int a, b, rf = 0;
   for (a = 1; a <= 10; a++)
   {
        Console.Write(a + ".Sayı:");
        b = Convert.ToInt32(Console.ReadLine());
        if (rf < b)
        {
             rf = b;
        }
   }
   Console.WriteLine("Girilen sayılar içerisinde
en büyük sayı:{0} dır", rf);
```

For Döngüsü Örnekleri

Örnek :0'dan klavyeden girilen sayıya kadar olan sayıların toplamını ekrana yazdıran programı yazınız.

For Döngüsü Örnekleri

Örnek :Aşağıdaki çıktıyı veren programı for döngüsü ile yazınız.

For Döngüsü Örnekleri

Örnek :Aşağıdaki çıktıyı veren programı for döngüsü ile yazınız.

```
*****
*****
*****
*****
*****
```

For Döngüsü Örnekleri

Örnek :Aşağıdaki çıktıyı veren programı for döngüsü ile yazınız.

For Döngüsü Örnekleri

Örnek :1 ile 50 arasındaki sayılardan asal olanlarını bulup ekrana yazan C# programını yazınız.

Cevap: Asal sayılar sadece bire ve kendisine bölünebilen sayılardır. İlk döngümüz 1 ile 50 arasındaki sayıları elde etsin.İkinci döngü ise bunların içerisinden asal olan sayıları bulsun.

```
int a, b;
            bool d;
            for (a = 2; a <= 50; a++)
            {
                d = false;
                for (b = 2; b <= a - 1; b++)
                {
                    if (a % b == 0)
                    {
                        d = true;
                        break;
                    }
                    if (d == false)
                    {
                        Console.WriteLine(a);
                    }
```

While Döngüsü

While döngüsü bir koşul sağlanıyorken dönmeye devam eder. Koşul yanlış (false) sonucunu verdiği zaman ise sonlandırılır.

Yapısı:

```
while(koşul)
{
    yapılacak işler;
}
.
```

Koşul, döngünün ne kadar çalışacağını denetleyen mekanizmadır. Koşul sağlanıyorken döngü çalışmaya devam eder. Koşul sağlanmadığında ise döngü durur. Koşulda genellikle başlangıç değerinin durumu denetlenir.

While Döngüsü Örnekleri

Örnek : Klavyeden Sıfır Girilene Kadar Girilen Sayıların Toplamını aldıran program.

```
İnt say,toplam=0;

While(say!=0)
{
        Console.Write("Sayı Gir");
        say=Convert.ToInt16(Console.ReadLine());
        toplam+=say;
}

Console.Write("Sayıların Toplamı={0}",toplam);
Console.Readkey();
```

While Döngüsü Örnekleri

Örnek : Klavyeden Sıfır Girilene Kadar Girilen Sayıların Toplamını aldıran program. (break ile yapınız)

```
İnt say,toplam=0;
While(true)
{
        if(say!=0)
                break;

        Console.Write(“Sayı Gir”);
        say=Convert.ToInt16(Console.ReadLine());
        toplam+=say;
}

Console.Write(“Sayıların Toplamı={0}”,toplam);
Console.Readkey();
```

While Döngüsü Örnekleri

Örnek : Bilgisayara rastgele ürettirdiğimiz bir sayıyı 5 hakta tahmin etmeye çalışan bir bilgisayar programı yazınız?

```
            int hak = 5;
            Random rnd = new Random();
            int tutulan = rnd.Next(1, 50);
            int sayi=0;
while (hak>0)
{
            Console.Write("Bir sayı giriniz: ");
            sayi = Convert.ToInt32(Console.ReadLine());
            hak = hak - 1;
            if (sayi == tutulan)
            {
                  Console.WriteLine("Tebrikler sayıyı doğru tahmin ettiniz");
                        break;
            }else {
                        if (sayi > tutulan)
                                    Console.WriteLine("Aşağı");
                        else
                                    Console.WriteLine("Yukarı");
                  }
```

Do - While Döngüsü

For ve while döngülerinde döngü bloklarının koşul sağlanmadığı takdirde hiç çalıştırılmama ihtimali vardır. Ancak döngünün en az bir kere çalıştırılması istenilen durumlarda do-while döngüleri kullanılırlar.
Do-While döngülerinde koşul döngü içerisindeki işlemler bir kez gerçekleştirildikten sonra kontrol edilir. Koşul doğru olduğu müddetçe de döngü içerisindeki işlemler tekrarlanmayı sürdürür.
Genel yazım şekli şöyledir.

```
do
{
        yapılacak işler;
}
while(koşul);
```

Do - While Döngüsü Örnekleri

Örnek : Klavyeden negatif sayı girilene kadar sayı girilmesini sağlayan, bu girilen sayılarda 4' e kalansız bölünebilen veya 7' ye kalansız bölünemeyen sayıların adetini bulduran programı yazınız?

Çözüm

```
int girilensayi= 1, adet = 0;
do
{
        Console.Write("Sayı Gir=");
        girilensayi = Convert.ToInt16(Console.ReadLine());
        if(girilensayi%4 ==0 || girilensayi %7 !=0)
          adet++;
}While(girilensayi >0)
Console.WriteLine("Adet="+adet);
Console.ReadKey();
```

Do - While Döngüsü Örnekleri

Örnek : Klavyeden girilen bir sayıyı 2 li sayı sistemine çeviren C# programını yazınız?

Dallanma - Atlama (Jump) Komutları

Programın akışı esnasında başka satırlara atlama işlemi gerçekleştiren bir takım anahtar sözcükler vardır.

Bunlar;

- **break,**
- **continue,**
- **goto,**
- **return anahtar sözcükleridir.**

Break

Break anahtar sözcüğü döngülerden çıkmak için kullanılır. Döngülerde, break anahtar sözcüğüne rastlandığı anda döngüden çıkılır ve program döngü bloğundan sonraki ilk deyimle akışına devam eder.

Break anahtar sözcüğü döngü bloklarının ya da switch bloklarının dışında kullanılamazlar.

Break Örneği

Örnek : ‘A’ harfinden başlayıp ‘Z’ye kadar devam eden bir döngü de ‘K’ harfine gelindiğinde döngüden çıkan programın kodunu yazınız.

Çözüm:

```
        for (char i = 'A'; i <= 'Z'; i++)
        {
                if (i == 'K')
                break;
                Console.WriteLine(i);
                Console.WriteLine("Döngüye devam ediliyor...");
        }
Console.WriteLine("Döngüden çıkıldı...");
```

Continue

Continue ifadesi, break ifadesine benzerdir ve bir for, foreach, while ya da do...while döngüsü içinde de kullanılabilir. Ancak, döngünün dışına çıkmak yerine mevcut döngüden çıkarak bir sonraki döngüye geçişi sağlayacaktır.

Örnek :

```
int i = 1;
int k = 1;
while (i < 10)
{
        Console.WriteLine("i:{0} iken k:{1}",i,k);
        i++;
        continue;
        k++;
}
```

Break Örneği

Örnek : ‘A’ harfinden başlayıp ‘Z’ye kadar devam eden bir döngü de ‘K’ harfine gelindiğinde döngüden çıkan programın kodunu yazınız.

Çözüm:

```
        for (char i = 'A'; i <= 'Z'; i++)
        {
                if (i == 'K')
                break;
                Console.WriteLine(i);
                Console.WriteLine("Döngüye devam ediliyor...");
        }
Console.WriteLine("Döngüden çıkıldı...");
```

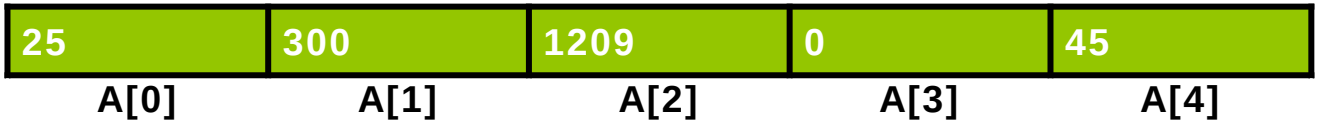
Diziler (Arrays)

Değişkenleri öğrenirken gördük ki her değişkene sadece bir değer atayabiliriz. Bazı durumlarda aynı tipteki değişkenleri bir arada tutma ihtiyacı duyabiliriz. C# bize aynı tipteki değişkenleri tek bir adla saklayabileceğimiz **dizileri (Array)** sunmaktadır.

Dizi (array), ortak isimle anılan aynı tipteki veriler topluluğudur. Diziler bir programlama dilindeki en önemli veri yapılarından biridir. Bellekte ardışık olarak yer kaplayan veri kümesine array(dizi) adı verilir.

Diziler yapılarına göre ikiye ayrılır.

- Statik Diziler
- Dinamik Diziler



Neden Dizilere İhtiyaç Var?

Diziler sayesinden aynı türdeki işlemler için birden fazla değişken tanımlamaktan kurtuluruz. Tek bir değişkenle 10 larca 100lerce veriyi ayrı ayrı saklama imkanı sunar diziler.

Örneğin: Sınıfımızda 180 kişinin isimlerini ve programlama dersi notlarını programa girip ilk üç kişiyi, notlarının ortalamasını ve kaç kişinin kalıp kaç kişinin geçtiğini bulmak istiyoruz.

Bunun için aşağıdaki gibi her bir öğrencinin adı soyadı ve notu için bir değişken tanımlamanın saçma ve kullanışsız olduğu aşikardır.

- **string** isim1,isim2,isim3,isim4……isim180;
- **int** not1,not2,not3,not4,not5……..not180;

Bunun yerine sadece her bir tür için iki adet 180 elemanlı dizi değişkeni tanımlamak daha kullanışlı ve kolay olacaktır.

```
string[] isimler=new string[180];
int[] notlar=new int[180];
```

Statik Diziler

Statik diziler temel dizi yapısını oluştururlar. Dinamik dizilere göre belirli kısıtlamalara sahiptirler. Klasik dizilerin programlama tekniklerine getirdikleri kolaylıkların dışında birtakım kısıtlamaları da vardır. Bu kısıtlamaların en başında da dizilerin boyutları gelmektedir. Bir dizinin boyutu, dizi tanımlanırken belirlenir ve programın akışı esnasında genişletilip-daraltılama.

Bir diğer kısıtlama da; örneğin, programın başlangıcında 250 elemanlı bir dizi tanımladık ve bunun yalnızca 120'sini kullandık, geriye kalan 130 elemanlık bellek alanı ise boşu boşuna bellekte yer kaplamış olur.

Statik Dizi Tanımlama

Dizi tanımlamada iki farklı yol izlenebilir.

- **Birincisi: Sadece dizinin türünü ve boyutunu belirtmek yoluyla tanımlanabilir. Bu tür tanımlamalarda dizilerin içerisi boş olur değer atanmamış halidir.**

Örnek:

```
string[] isimler = new string[180];
int[] numara = new int[180];
```

- **İkincisi: Dizinin tanımlaması yapılırken boyutunu belirtmek yerine dizi değerlerini yükleme yoluyla gerçekleştirilir.**

Örneğin:

```
string[] meyveler = {"elma", "kiraz", "çilek", "armut", "erik"};
```

Bu tür tanımlamalarda dizinin içerisine direk olarak dizi değerlerini de yüklemiş oluruz

Statik Dizi

Tanımlarken Dikkat Edilecek Kurallar

1. Arrayler hafızaya toplu şekilde yerleştikleri için tanımlanan bir arrayin eleman sayısı programın hafızada kapladığı alanı doğrudan etkileyecektir. Array boyutunu belirlerken kullanılacak eleman sayısından daha fazlasının tanımlanmamasına dikkat edilmelidir.
2. Elemanları gösteren sayıların(indis) sıfırdan başlayıp eleman sayısının bir eksiğine kadar devam ettiği unutulmamalıdır.

Statik Dizilere Değer Girişi

Dizilere değer girişi yapılırken dizinin hangi elemanına değer girilecekse o eleman belirtilmelidir.

Örnek: Aşağıdaki 5 elemanlı dizinin 3. elemanına 24 değerini giriniz.

```
int [ ] sayilar=new int[5];
sayilar[2]=24 ; ///burada dizilerin eleman sayılarının 0 dan başladığını unutmamalıyız. Bu sebeple 3. eleman 2 nolu indise sahiptir.
```

Statik Dizilerden Değer Okuma

Dizilerde değer okurken dizinin hangi elemanından değer okunacaksao eleman belirtilmelidir.

Örnek: Aşağıdaki 5 elemanlı dizinin 5. elemanındaki değeri yazdırın.

```
int [ ] sayilar=new int[5];
Console.WriteLine(sayilar[4]); ///burada dizilerin eleman sayılarının 0 dan başladığını unutmamalıyız. Bu sebeple 3. eleman 2 nolu indise sahiptir.
```

Statik Dizi Örnekleri

Örnek: 5 kişilik bir sınıfta öğrenci adları ve öğrenci notları verilen öğrencilerin ad ve notlarını sırasıyla alt alta listeleyen programı yazınız.

Statik Dizi Örnekleri

Örnek: 0-25 arasındaki çift sayıları 20 elemanlı diziye yazdınız.

Statik Dizi Örnekleri

Örnek: Klavyeden 5 adet şehri diziye kaydederek daha sonra bu dizinin elemanlarını listeleyen programı yazınız..

Foreach

Örnek: Klavyeden 5 adet şehri diziye kaydederek daha sonra bu dizinin elemanlarını listeleyen programı yazınız.(Foreach döngüsü ile).

Statik Dizi Örnekleri

Örnek: Klavyeden 10 adet sayı girilmesini isteyip bu sayıları 10 elemanlı bir diziye kaydeden ve dizi içindeki değerleri küçükten büyüğe sıralayan programı yazınız.

```
int[] sayilar =new int[10];
 for (int m = 1; i <=10; m++)
        {
            Console.WriteLine("{0} . Sayıyı giriniz",m);
            sayilar[m-1]=Convert.ToInt32(Console.ReadLine());
        }
        for (int i = 0; i < sayilar.Length-1; i++)
        {
           for (int j = i; j >= 1; j--)
           {
              if (sayilar[j] < sayilar[j - 1])
              {
                 int tasiyiciSayi = sayilar[j];
                 sayilar[j] = sayilar[j - 1];
                 sayilar[j - 1] = tasiyiciSayi;
              }

        }
```

Bazı Dizi Özellikleri

Diziler, .NET Framework içinde tanımlı Array sınıfı temsil eder. Tüm diziler Array sınıfında tanımlı özellikleri ve metotları kullanırlar. Bu metotlardan ve özelliklerden en sık kullanılanları şunlardır;

- **Length,**
- **Clear,**
- **Reverse**
- **Sort**
- **indexOf**
- **Copy**
- **CopyTo**

Length

Dizinin saklayabileceği toplam eleman sayısını veren ve int türünde bir değer döndüren özelliktir. Yani dizinin boyutunu dödürür.

Örnek:

```
int[] ciftSayilar = new int[10];
elemanSayisi = ciftSayilar.Length;
Console.WriteLine("ciftSayilar dizi içerisinde toplam {0} eleman bulunmaktadır.",elemanSayisi);
```

Clear

Parametre olarak verilen dizinin, belirtilen indeks aralığındaki tüm değerlerini temizler. Temizleme işleminde atanan değer, dizi elemanlarının tiplerine göre değişir.

Örneğin int tipinde tanımlı bir dizinin elemanları temizlenirse 0 değerini alacaktır. Buna karşın String tipindeki elemanlar "" (boş yazı) değerini alır.

Kullanım:

```
Array.Clear(diziAdi, baslangicIndeksi, Adet);
```

Örnek: Yeni tanımladığımız ve içeriğini oluşturduğumuz bir dizinin elemanlarının nasıl silindiğini inceleyelim

```
int[] sayilar = new int[] { 43, 16, 66, 26 };
        Console.WriteLine("Temizlenmeden önce dizinin elemanları");
        Console.WriteLine("-----------------------------------");
        for (int i = 0; i < sayilar.Length; i++)
                Console.WriteLine("sayilar[{0}]: {1}", i,sayilar[i]);
        Array.Clear(sayilar, 1, 2);
        Console.WriteLine("\nTemizlendikten sonra dizinin elemanları");
        Console.WriteLine("-----------------------------------");
        for (int i = 0; i < sayilar.Length; i++)
                Console.WriteLine("sayilar[{0}]: {1}", i, sayilar[i]);
```

Reverse

Parametre olarak verilen dizinin eleman sırasını tersine çevirir. Dizinin tüm elemanlarının veya belirli indeks aralığındaki elemanlarının sırası tersine çevrilebilir.

Kullanımı: Array.Reverse(diziAdi);

Örnek : Alfabe adlı dizi içerisine girilen A-Z'ye harfleri tersten ekrana yazdıran programın kodunu yazınız.

```
char[] alfabe = new char[26];  char harf;    int i=0;
            Console.WriteLine("A'dan Z'ye İngiliz Alfabesi");
            Console.WriteLine("-------------------------");
            for (harf = 'A'; harf <= 'Z'; harf++)
            {
                        alfabe[i] = harf;
                        Console.Write(alfabe[i] + " ");
                        i++;
            }
            Console.WriteLine("\n");
            Array.Reverse(alfabe);
            Console.WriteLine("Z'den Z'ya İngiliz Alfabesi");
            Console.WriteLine("-------------------------");
            for (i = 0; i < 26; i++)
                        Console.Write(alfabe[i] + " ");
```

Sort

Parametre olarak verilen dizinin elemanlarını küçükten büyüğe sıralar. Eğer dizi numerik ise rakamların büyüklüğüne göre, yazı tiplerinde ise baş harflerine göre sıralanır.Reverse'ün tersidir.

Kullanımı: Array.Sort(diziAdi);

Örnek : Klavyeden girilen 5 sayıyı küçükten büyüğe sıralayan programı yazınız.

```
int[] sayilar = new int[5];
int i=0;
for (i = 0; i < 5; i++)
{
        Console.Write(i + 1 + ". Sayıyı Giriniz :");
        sayilar[i] = Convert.ToInt32(Console.ReadLine());
}
Array.Sort(sayilar);
```

IndexOf

İlk parametrede verilen dizide, ikinci parametrede verilen değeri arar. Aranan değer dizide bulunursa bulunan elemanın indeks değeri, bulunamazsa -1 döndürür.

Kullanımı: Array.IndexOf(diziAdi, arananDeger);

Örnek : Daha önceden değerleri girilmiş olan bir dizi içerisinde istenilen değeri bulan programın kodunu yazınız.

```
string[] iller = new string[] { "Ankara", "İstanbul","Kütahya", "İzmir", };
string aranan = "Kütahya";
int i=0;
        foreach (string il in iller)
        {
                Console.WriteLine("iller[{0}]: {1}", i, il);
                i++;
        }
        int indeks = Array.IndexOf(iller, aranan);
Console.WriteLine("İlleri dizisi içerisinde Kütahya'nın indeksi: "+indeks);
```

Copy

System.Array sınıfının Copy adlı durağan yöntemi kullanılarak değerler kopyalanabilir.

```
int[] rakam = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
int[] kopya = new int[rakam.Length];
Array.Copy(rakam, kopya, kopya.Length);
```

CopyTo

Dizi kopyalama işlemlerinden bir tanesi de bir dizideki elemanları başka bir diziye kopyalamak için for ifadesi yerine System.Array sınıfının CopyTo adlı bir dizinin elemanlarını başka bir diziye kopyalayan oluşum yöntemidir.

Kullanımı : diziAdi.CopyTo(kopyaDiziAdi, Kopyalamaya Başlanacak İndis)

```
int[] rakam = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
int[] kopya = new int[rakam.Length];
rakam.CopyTo(kopya, 0);
```

Yukarıda kullanılan CopyTo yöntemi içindeki 0. değeri dizinin başlangıç değeridir. Bu kopyalama işlemini herhangi bir dizi değerinden başlatabilirsiniz.

Clone

Üçüncü bir kopyalama yolu ise klonlama yöntemidir. Bu yöntem de System.Array sınıfı bir diziyi tümüyle oluşturur ve onu klonlamak için Clone adlı oluşum yöntemini kullanır.

Kullanım Şekli:

```
int[] rakam = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
int[] kopya = (int[])rakam.Clone();
```