

**T.C.
MİLLÎ EĞİTİM BAKANLIĞI**

BİLİŞİM TEKNOLOJİLERİ

KONTROL DEYİMLERİ 482BK0124

Ankara, 2011

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR	ii
GİRİŞ	1
ÖĞRENME FAALİYETİ-1	3
1. KARAR KONTROL DEYİMLERİ	3
1.1. If-Else Deyimi	3
1.2. İç-İçe If İfadesi	7
1.3. Switch-Case Deyimi	12
UYGULAMA FAALİYETİ	19
ÖLÇME VE DEĞERLENDİRME	20
ÖĞRENME FAALİYETİ-2	21
2. DÖNGÜ DEYİMLERİ	21
2.1. Döngü Çeşitleri	21
2.1.1. For Döngüsü	21
2.1.2. While Döngüsü	26
2.1.3. Do...While Döngüsü	28
2.1.4. Foreach Döngüsü	29
2.2. Jump (Dallanma – Atlama) Komutları	30
2.2.1. Break Anahtar Sözcüğü	30
2.2.2. Continue Anahtar Sözcüğü	32
2.2.3. Goto Anahtar Sözcüğü	33
2.2.4. Return Anahtar Sözcüğü	33
UYGULAMA FAALİYETİ	34
ÖLÇME VE DEĞERLENDİRME	35
ÖĞRENME FAALİYETİ-3	36
3. DİZİLER	36
3.1. Dizi Oluşturma	36
3.2. Diziye Değer Girme	38
3.3. Diziyi Yazdırma	43
3.4. Bazı Dizi Özellikleri ve Metotları	44
3.4.1. Length	44
3.4.2. Clear(dizi,baslangic,adet)	45
3.4.3. Reverse(Dizi)	47
3.4.4. Sort(Dizi)	48
3.4.5. IndexOf(Dizi,arananDeğer)	49
3.5. Dinamik Diziler	51
3.5.1. Capacity Özelliği:	52
3.5.2. Count Özelliği:	52
3.5.3. Add Metodu:	52
3.5.4. Insert Metodu:	53
3.5.5. Remove Metodu:	55
3.5.6. RemoveAt Metodu:	56
3.5.7. Sort Metodu:	57
UYGULAMA FAALİYETİ	58
ÖLÇME VE DEĞERLENDİRME	59
MODÜL DEĞERLENDİRME	60
CEVAP ANAHTARLARI	61
KAYNAKÇA	62

AÇIKLAMALAR

KOD	482BK0124
ALAN	Bilişim Teknolojileri
DAL/MESLEK	Alan Ortak
MODÜLÜN ADI	Kontrol Deyimleri
MODÜLÜN TANIMI	Bu modül programlama altyapısını oluşturan kontrol deyimleri kavramlarının öğrenildiği bir öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Bu modülün ön koşulu yoktur.
YETERLİK	Kontrol deyimlerini kullanmak.
MODÜLÜN AMACI	Genel Amaç Bu modül ile gerekli ortam sağlandığında; temel programlama işlemlerinde kontrol deyimlerini kullanabilecektir. Amaçlar <ol style="list-style-type: none">1. Karar kontrol deyimlerini kullanabileceksiniz.2. Döngü kontrollerini kullanabileceksiniz.3. Dizilerle çalışabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Ortam: Bilgisayar laboratuvarı Donanım: Bilgisayar, Programlama Yazılımı
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Programlama Temelleri dersinin bu modülünde sizler, programlamanın temel yapı taşlarından “Akış Kontrol Deyimleri” ile öğreneceksiniz.

Programcılığa ilk adımlarınızı bu kontrol deyimleriyle gerçekleştireceksiniz.

Kontrol deyimleri programlarımızın işleyişinde çeşitli kontrol ve akış işlemlerini gerçekleştirmenizi sağlar.

Bu modül ile if, switch, for, while, do-while, foreach ve dizi yapılarını öğrenip, her konu sonunda bolca örneklerle konuları pekiştirmeniz sağlanacaktır.

ÖĞRENME FAALİYETİ-1

AMAÇ

Bu modül ile karar kontrol deyimlerini kullanabilecek ve programlarınızda uygulayabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız

- Günlük hayatta bir olayın gerçekleşmesi için önceden istenilen/beklenen koşullara verilebilecek örnekleri düşününüz.
- Evden dışarı çıkmadan önce havanın durumuna göre üzerimize alacağımız kıyafetlere nasıl karar veririz? Araştırınız.
- Bir kişinin askere gidebilmesi için gerekli şartlar nelerdir? Araştırınız.

1. KARAR KONTROL DEYİMLERİ

Program yazarken bazı noktalarda belirli koşullar altında gerçekleşmesini istenilen durumlar olabilir. Bu bölümde anlatılan if-else ve switch deyimleri ile bu tür kapsamlı programlar geliştirilebilir.

Genel anlamda programlama dilinde kullanılan koşul yapıları iki çeşittir. Bunlar;

- if-else deyimi
 - switch deyimi
- dir.

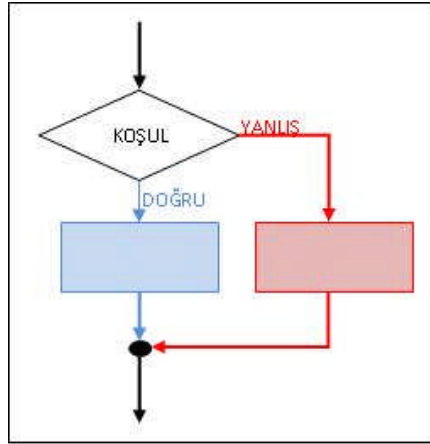
1.1. If-Else Deyimi

If deyimi bir programın akışını kontrol etmek için kullanılır. Belirli bir şarta göre yapılması istenilen işlemler, If-Else deyimi kullanılarak gerçekleştirilir.

If-Else deyiminin kullanımı ve akış diyagramları ile gösterimi ise şu şekildedir.

➤ Kullanımı:

```
if(koşul)
{
    Koşul doğruysa yapılacak işlemler;
}
else
{
    Koşul yanlışsa yapılacak işlemler;
}
```



Şekil 0.1: If-Else Deyimi Akış Diyagramı

Yukarıdaki diyagramdan da görüleceği üzere, programın akışı If deyiminin olduğu satıra geldiğinde parantezler içerisindeki KOŞUL ifadesi çalıştırılır. Bu koşul ifadesi **true (Doğru)** yada **false (Yanlış)** olmak üzere bir değer üretmektedir.

Şayet koşulumuz doğruysa (true) programımızın akışı mavi renkle gösterilen **doğruysa** kısmından devam edecek ve kırmızıyla gösterilen **yanlışsa** kısmına uğramayacaktır. Eğer koşulumuz yanlışsa (false) bu sefer programımız **yanlışsa** kısmından kırmızıyla belirtilen yoldan devam edecektir.

Not 1: Eğer programımızın akışında sadece koşulun doğru olmasına bağlı işlem yapılması isteniyor, koşulun yanlış olduğu durumlarda işlem yapılması istenmiyorsa Else bloğu program içerisinde hiç kullanılmaz.

➤ **Kullanımı:**

```
if(koşul)
{
    Koşul doğruysa yapılacak işlemler;
}
```

Not 2: Eğer If veya Else'den sonra sadece bir komut yazılacak ise küme parantezleri ({}) kullanılmayabilir.

➤ **Kullanımı:**

```
if(koşul)
    Koşul doğruysa yapılacak işlemler;
else
    Koşul yanlışsa yapılacak işlemler;
```


Örnek 1-1: Klavyeden yaşı girilen kişinin ehliyet alıp alamayacağını belirten programı yazınız.

Bu örneğimizde sayıların karşılaştırılmasını inceleyelim.

```
Console.Write("Yaşınızı giriniz: ");
int yas = Convert.ToInt32(Console.ReadLine());
if (yas < 18)
    Console.WriteLine("Yaşınız 18'den küçük olduğu için ehliyet alamazsınız");
else
    Console.WriteLine("Ehliyet alabilecek yaştasınız.");
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler	Ekran Çıktısı
16	
25	
18	
17	

Eşitlik bakımından değişkenleri karşılaştırmak için == operatörünü kullandığına, özellikle dikkat edin. Bu amaç için = operatörünü kullanmayınız. Tek bir = operatörü, değişkenleri atamak için kullanılır.

Örnek 1-2: “Ünlü şairimiz Mehmet Akif’in soyadı nedir?” sorusunu kullanıcıya soran cevabını isteyen programı yazınız.

Bu örneğimizde metinsel ifadelerin karşılaştırılmasını inceleyelim.

```
Console.Write("Ünlü şairimiz Mehmet Akif'in soyadı nedir?\nCevabınız: ");
string cevap = Console.ReadLine();
if (cevap == "Ersoy")
{
    Console.Write("Tebrikler bu sorumuza doğru cevap verdiniz...");
}
else
{
    Console.WriteLine("Malesef yanlış cevap");
}
Console.ReadLine();
```

Not 3: Bazı programlama dilleri büyük/küçük harf duyarlı bir dil olduğu için “Ersoy”, “ersoy” veya “ERSOY” cevaplarından yalnızca “Ersoy” cevabını kabul edecektir.

If koşul deyimlerde zaman zaman birden fazla koşula bağlı bir takım işlemler yapmamız gerekebilir.

➤ **Kullanımı:**

VE (&&) bağlacı ile

```
if((koşul1) && (koşul2))
{
    koşul1 ve koşul2 doğruysa yapılacak işlemler;
}
else
{
    koşullardan en az birisi veya her ikisi de yanlış ise
    yapılacak işlemler;
}
```

VEYA (||) bağlacı ile

```
if((koşul1) || (koşul2))
{
    koşul1 veya koşul2'den en az birisi veya her ikisi de doğruysa
    yapılacak işlemler;
}
else
{
    koşullardan her ikisi de yanlış ise yapılacak işlemler;
}
```

Örnek 1-3: Klavyeden girilen sayının hem 3'e hem de 5'e kalansız bölünüp bölünemediğini ekrana yazan programı yazınız.

```
Console.Write("Bir sayı giriniz:");
int sayi = Convert.ToInt32(Console.ReadLine());
if((sayi % 3 == 0) && (sayi % 5 ==0))
    Console.WriteLine("{0} sayısı hem 3'e hem de 5'e kalansız
    bölünebilir",sayi);
else
    Console.WriteLine("{0} sayısı hem 3'e hem de 5'e kalansız
    bölünemez", sayi);
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler	Ekran Çıktısı
150	
38	
64	
90	

Örnek 1-4: Klavyeden girilen cinsiyet ve yaş bilgilerine göre, kişinin askere gidip gidemeyeceğini yazan programı yazınız.

```
char cinsiyet;  
int yas;  
Console.Write("Lütfen cinsiyetinizi giriniz (E/K):");  
cinsiyet=Convert.ToChar(Console.ReadLine());  
Console.Write("Lütfen yaşıınızı giriniz:");  
yas =Console.Read();  
if (((cinsiyet == 'E') || (cinsiyet=='e')) && (yas >= 20))  
{  
    Console.WriteLine("Askere Gidebilir");  
}  
else  
{  
    Console.WriteLine("Askere Gidemez");  
}  
}
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler		Ekran Çıktısı
Cinsiyet	Yaş	
E	18	
K	21	
E	23	
K	19	

1.2. İç-İçe If İfadesi

Birden fazla koşula ihtiyaç duyulan durumlarda iç-içe If ifadeleri kullanılırlar. Bir if koşuluna kaç tane else if ekleyebileceğiniz konusunda hiçbir sınır yoktur.

İç-içe If ifadelerinin kullanımı ise şu şekildedir.

➤ Kullanımı:

```
if(koşul1)  
{  
    koşul1 doğruysa yapılacak işlemler;  
}  
else if(koşul2)  
{  
    koşul1 yanlışsa ve koşul2 doğruysa yapılacak işlemler;  
}  
else  
{  
    her iki koşul da yanlışsa yapılacak işlemler;  
}
```

Dilerseniz iç-içe If ifadelerini birkaç örnekle açıklamaya çalışalım.

Örnek 1-5: Klavyeden girilen iki sayıyı karşılaştıran programı yazınız.

```
int sayi1, sayi2;
Console.Write("1. sayıyı giriniz: ");
sayi1 = Convert.ToInt32(Console.ReadLine());
Console.Write("2. sayıyı giriniz: ");
sayi2 = Convert.ToInt32(Console.ReadLine());
if(sayi1>sayi2)
    Console.WriteLine("{0} sayısı {1} sayısından
büyüktür.", sayi1, sayi2);
else if(sayi1<sayi2)
    Console.WriteLine("{0} sayısı {1} sayısından
büyüktür.", sayi2,
sayi1);
else
    Console.WriteLine("{0} sayısı ile {1} sayısı
birbirine eşittir.", sayi1, sayi2);
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler		Ekran Çıktısı
sayi1	sayi2	
12	17	
43	43	
98	21	
-106	106	
-66	-16	

Örnek 1-6: Klavyeden girilen puanın 5'lik sistemdeki not karşılığını yazan programı yazınız.

```
Console.Write("Puanınızı giriniz (0-100):");
int puan = Convert.ToInt32(Console.ReadLine());
if (puan >= 0 && puan < 25)
    Console.WriteLine("Puanınızın 5'lik sistemdeki karşılığı
0'dır");
else if (puan >= 25 && puan < 45)
    Console.WriteLine("Puanınızın 5'lik sistemdeki karşılığı
1'dir");
else if (puan >= 45 && puan < 55)
    Console.WriteLine("Puanınızın 5'lik sistemdeki karşılığı
2'dir");
else if (puan >= 55 && puan < 70)
    Console.WriteLine("Puanınızın 5'lik sistemdeki karşılığı
3'tür");
else if (puan >= 70 && puan < 85)
```

```

        Console.WriteLine("Puanınızın 5'lik sistemdeki karşılığı 4'tür");
    else if (puan >= 85 && puan <= 100)
        Console.WriteLine("Puanınızın 5'lik sistemdeki karşılığı 5'tir");
    else
        Console.WriteLine("Hatalı puan girdiniz. Puanınız 0 ile 100 arasında olmalıdır.");
}

```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler	Ekran Çıktısı
86	
69	
43	
77	
14	
52	

İç-içe If ifadelerinin bir başka kullanımı da şu şekildedir.

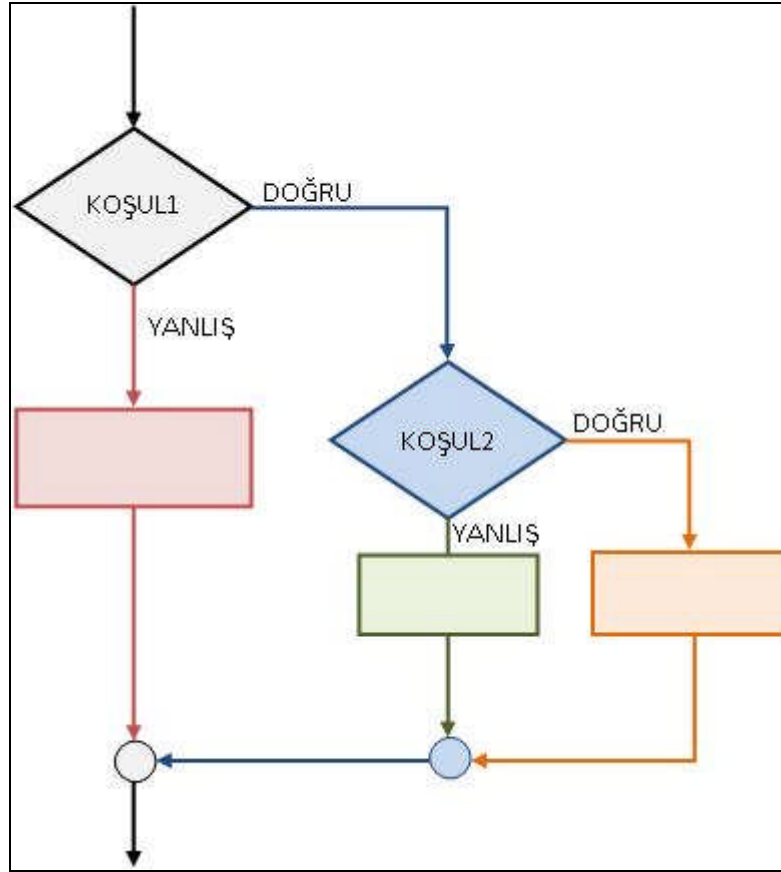
Örneğin bir koşulun sağlanması durumunda başka koşullara göre işlem yapılması istenilen durumlarda yine iç-içe If ifadeleri kullanılırlar. Bu durumdaki iç-içe If ifadelerinin kullanımları ve akış diyagramlarıyla gösterimi şu şekildedir;

➤ Kullanımı:

```

if(koşul1)
{
    if(koşul2)
    {
        koşul2 doğruysa yapılacak işlemler;
    }
    else
    {
        koşul2 yanlışsa yapılacak işlemler;
    }
}
else
{
    koşul1 yanlışsa yapılacak işlemler;
}

```



Şekil 0-2. İç-içe If İfadesi

Örnek 1-7: Daha önceden belirlenen kullanıcı adı ve şifreyi kontrol eden programı yazınız.

```

string kullanıcıAdi, sifre;
Console.Write("Lütfen kullanıcı adınızı giriniz:");
kullanıcıAdi = Console.ReadLine();
if (kullanıcıAdi == "Admin" || kullanıcıAdi=="ADMİN" ||
kullanıcıAdi=="admin")
{
    Console.Write("Lütfen şifrenizi giriniz:");
    sifre = Console.ReadLine();
    if (sifre == "123rty")
        Console.WriteLine("Tebrikler Kullanıcı ve Şifreniz
Doğru");
    else
        Console.WriteLine("Şifrenizi Hatalı Girdiniz");
}
else
    Console.WriteLine("Kullanıcı Adınızı Hatalı Girdiniz");
  
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler		Ekran Çıktısı
Kullanıcı Adı	Şifre	
Admin	123RTY	
Yönetici	123rty	
Admin	123rty	
admin	123rty	
ADMİN	123rty	

!!! Uyarı: Yukarıdaki örnekte şifre “123rty” şeklinde verilmiştir.

Örnek 1-8: Basit bir hesap makinesi yapımı.

```
byte secim;
double sayi1,sayi2,sonuc;
Console.WriteLine("1.TOPLAMA");
Console.WriteLine("2.ÇIKARMA");
Console.WriteLine("3.ÇARPMA");
Console.WriteLine("4.BÖLME");
Console.WriteLine("-----");
Console.Write("İşlem tipinizi seçiniz (1-4):");
secim = Convert.ToByte(Console.ReadLine());
if (secim == 1)
{
    Console.Clear();
    Console.WriteLine("*****");
    Console.WriteLine("* Seçilen işlem TOPLAMA işlemi *");
    Console.WriteLine("*****");
    Console.Write("1.Sayıyı giriniz:");
    sayi1 = Convert.ToDouble(Console.ReadLine());
    Console.Write("2.Sayıyı giriniz:");
    sayi2 = Convert.ToDouble(Console.ReadLine());
    sonuc = sayi1 + sayi2;
    Console.WriteLine("Sonuç={0}", sonuc);
}
else if (secim == 2)
{
    Console.Clear();
    Console.WriteLine("*****");
    Console.WriteLine("* Seçilen işlem ÇIKARMA işlemi *");
    Console.WriteLine("*****");
    Console.Write("1.Sayıyı giriniz:");
    sayi1 = Convert.ToDouble(Console.ReadLine());
    Console.Write("2.Sayıyı giriniz:");
    sayi2 = Convert.ToDouble(Console.ReadLine());
    sonuc = sayi1 - sayi2;
    Console.WriteLine("Sonuç={0}", sonuc);
}
```

```

}
else if (secim == 3)
{
    Console.Clear();
    Console.WriteLine("*****");
    Console.WriteLine("* Seçilen işlem ÇARPMA işlemi *");
    Console.WriteLine("*****");
    Console.Write("1.Sayıyı giriniz:");
    sayi1 = Convert.ToDouble(Console.ReadLine());
    Console.Write("2.Sayıyı giriniz:");
    sayi2 = Convert.ToDouble(Console.ReadLine());
    sonuc = sayi1 * sayi2;
    Console.WriteLine("Sonuç={0}", sonuc);
}
else if (secim == 4)
{
    Console.Clear();
    Console.WriteLine("*****");
    Console.WriteLine("* Seçilen işlem ÇIKARMA işlemi *");
    Console.WriteLine("*****");
    Console.Write("1.Sayıyı giriniz:");
    sayi1 = Convert.ToDouble(Console.ReadLine());
    Console.Write("2.Sayıyı giriniz:");
    sayi2 = Convert.ToDouble(Console.ReadLine());
    if (sayi2 != 0)
    {
        sonuc = sayi1 / sayi2;
        Console.WriteLine("Sonuç={0}", sonuc);
    }
    else
        Console.WriteLine("!!! SIFIRA BÖLME HATASI !!!");
}
}

```

1.3. Switch-Case Deyimi

Switch-Case deyimi de If-Else deyimleri gibi karar kontrol mekanizmalarında kullanılmaktadır. Switch-Case deyimi genellikle karmaşık if-else bloklarının yerine, daha okunabilir oldukları için tercih edilmektedir. Switch-Case ile yapabileceğimiz karşılaştırmaları if-else ile de yapabiliriz.

Switch-Case yapısı şu şekilde çalışır; bir deyimin değeri, sabitlerden oluşan bir listede peş peşe test edilir. Deyimin değeri sabitlerden birisiyle eşleşince, bu eşleşmeyle ilgili işlemler gerçekleştirilir.

Switch-Case ifadesinin genel formu şu şekildedir;

➤ **Kullanımı:**

```
switch(ifade)
{
    case sabit1:
        Yapılacak işlemler;
        break;
    case sabit2:
        Yapılacak işlemler;
        break;
    case sabit3:
        Yapılacak işlemler;
        break;
    .
    .
    .
    default:
        Yapılacak işlemler;
        break;
}
```

Switch-Case yapısının çalışmasına bir göz atalım;

- Önce switch parantezleri içerisindeki ifade hesaplanır.
- Programın akışı, hesaplanan ifade ile aynı case sabitinin bulunduğu satıra gelir.
- Eğer hesaplanan ifade, mevcut case sabitlerinden herhangi birisi ile eşleşmiyorsa **default** anahtar sözcüğünün bulunduğu yere gelir ve program buradan devam eder.

Her case satırı içerisindeki işlemlerimiz tamamlandıktan sonra, ilgili case satırının sonuna geldiğimizi belirtmek için **break** komutu kullanılır.

Eğer aşağıdaki örnekteki gibi break komutu kullanılmazsa, “*Control cannot fall through from one case label ('case 1:') to another*” yani “*Bir case etiketinden ('case1:') başka bir case etiketine geçilemez*” hatasını alırız.

```
switch(ifade)
{
    case 1:
        Yapılacak işlemler;
    case 2:
        Yapılacak işlemler;
        break;
    case 3:
        Yapılacak işlemler;
        break;
    default:
        Yapılacak işlemler;
        break;
}
```

Switch-case yapısında case durumların sırasının sorun olmamaktadır. default durumunu bile ilk sıraya koyabilirsiniz. Sonuç olarak, iki durum aynı olamayacağı için ilgili case yapısına gelindiğinde o satırın çalışması sağlanacaktır.

```
switch(ifade)
{
    default:
        Yapılacak işlemler;
        break;
    case 3:
        Yapılacak işlemler;
        break;
    case 1:
        Yapılacak işlemler;
        break;
    case 2:
        Yapılacak işlemler;
        break;
}
```

Switch-Case Yapısı İle İlgili Önemli Kurallar:

- Case anahtar sözcüğünün yanındaki ifadeler sabit olmak zorundadırlar. Bu ifadeler içerisinde değişken bulunamaz.
- Case ifadeleri herhangi bir tam sayı sabiti, karakter veya string sabiti olabilir.
- Default durumunu istediğimiz yere yazabiliriz. Aynı şekilde case ifadelerini de istediğimiz sırada yazabiliriz.
- Bir switch bloğunda iki veya daha fazla sayıda aynı değere sahip case ifadesi bulunamaz.
- Bir switch bloğunda default case olmak zorunda değildir.
- Akış herhangi bir case ifadesine geldiğinde, akış farklı bir case ifadesine yönlendirilmek istenirse goto anahtar sözcüğü kullanılır.

Örnek 1-9: Klavyeden girilen 1-12 arasındaki sayı değerine göre o sıradaki ayın ismini veren programı yazınız

```
byte ay;
Console.WriteLine("1-12 arasında bir sayı giriniz:");
ay = Convert.ToByte(Console.ReadLine());
switch (ay)
{
    case 1:
        Console.WriteLine("{0}.ay OCAK ayıdır.", ay);
        break;
    case 2:
        Console.WriteLine("{0}.ay ŞUBAT ayıdır.", ay);
        break;
    case 3:
        Console.WriteLine("{0}.ay MART ayıdır.", ay);
        break;
    case 4:
```

```

        Console.WriteLine("{0}.ay NİSAN ayıdır.", ay);
        break;
    case 5:
        Console.WriteLine("{0}.ay MAYIS ayıdır.", ay);
        break;
    case 6:
        Console.WriteLine("{0}.ay HAZİRAN ayıdır.", ay);
        break;
    case 7:
        Console.WriteLine("{0}.ay TEMMUZ ayıdır.", ay);
        break;
    case 8:
        Console.WriteLine("{0}.ay AĞUSTOS ayıdır.", ay);
        break;
    case 9:
        Console.WriteLine("{0}.ay EYLÜL ayıdır.", ay);
        break;
    case 10:
        Console.WriteLine("{0}.ay EKİM ayıdır.", ay);
        break;
    case 11:
        Console.WriteLine("{0}.ay KASIM ayıdır.", ay);
        break;
    case 12:
        Console.WriteLine("{0}.ay ARALIK ayıdır.", ay);
        break;
    default:
        Console.WriteLine("Girmiş olduğunuz değer 1-12 arasında
değildir.");
        break;
}

```

Örnek 1-10: Klavyeden girilen değer ile seçimi yapılan şeklin alanını veya çevresini bulan programı yazınız

```

string sekil,secim;
int kenar1, kenar2;
Console.WriteLine("1.KARE----->(kare)");
Console.WriteLine("2.DİKDÖRTGEN--->(dikdörtgen)");
Console.WriteLine("-----");
Console.Write("Lütfen şeklin ismini yazınız:");
sekil = Console.ReadLine();
switch (sekil)
{
    case "kare":
        Console.WriteLine(" # ALAN----->(alan)");
        Console.WriteLine(" # ÇEVRE----->(çevre)");
        Console.WriteLine("-----");
        Console.Write("Lütfen seçiminizi yazınız:");
        secim = Console.ReadLine();
        switch (secim)
        {
            case "alan":

```

```

        Console.WriteLine("Karenin bir kenar uzunluğunu
giriniz:");
        kenar1 = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Karenin alanı={0}", kenar1*kenar1);
        break;
    case "çevre":
        Console.WriteLine("Karenin bir kenar uzunluğunu
giriniz:");
        kenar1 = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Karenin çevresi={0}", kenar1 * 4);
        break;
    default:
        Console.WriteLine("Geçerli bir seçim yapmadınız...");
        break;
    }
    break;
case "dikdörtgen":
    Console.WriteLine(" # ALAN----->(alan)");
    Console.WriteLine(" # ÇEVRE----->(çevre)");
    Console.WriteLine("-----");
    Console.WriteLine("Lütfen seçiminizi yazınız:");
    secim = Console.ReadLine();
    switch (secim)
    {
        case "alan":
            Console.WriteLine("Dikdörtgenin bir kenar uzunluğunu
giriniz:");
            kenar1 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Dikdörtgenin diğer kenar uzunluğunu
giriniz:");
            kenar2 = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Dikdörtgenin alanı={0}", kenar1 *
kenar2);
            break;
        case "çevre":
            Console.WriteLine("Dikdörtgenin bir kenar uzunluğunu
giriniz:");
            kenar1 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Dikdörtgenin diğer kenar uzunluğunu
giriniz:");
            kenar2 = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Karenin çevresi={0}",
(kenar1+kenar2) * 2);
            break;
        default:
            Console.WriteLine("Geçerli bir seçim yapmadınız...");
            break;
    }
    break;
default:
    Console.WriteLine("Geçerli bir seçim yapmadınız...");
    break;
}
}

```

Örnek 1-11: Bir önceki konuda Örnek-1-8’de IF-ELSE ile yaptığımız basit hesap makinesi programını Switch-Case ile biraz değiştirerek tekrar yapalım.

```
char secim;
double sonuc;
int sayi1, sayi2;
Console.WriteLine("1.TOPLAMA--->T");
Console.WriteLine("2.ÇIKARMA--->C");
Console.WriteLine("3.ÇARPMA --->R");
Console.WriteLine("4.BÖLME --->B ");
Console.WriteLine("-----");
Console.Write("İşlem tipinizi seçiniz (T-C-R-B):");
secim = Convert.ToChar(Console.ReadLine());
switch(secim){
    case 'T':
        Console.Clear();
        Console.WriteLine("*****");
        Console.WriteLine("* Seçilen işlem TOPLAMA işlemi *");
        Console.WriteLine("*****");
        Console.Write("1.Sayıyı giriniz:");
        sayi1 = Convert.ToInt32(Console.ReadLine());
        Console.Write("2.Sayıyı giriniz:");
        sayi2 = Convert.ToInt32(Console.ReadLine());
        sonuc = sayi1 + sayi2;
        Console.Write("Sonuç={0}", sonuc);
        break;
    case 'C':
        Console.Clear();
        Console.WriteLine("*****");
        Console.WriteLine("* Seçilen işlem ÇIKARMA işlemi *");
        Console.WriteLine("*****");
        Console.Write("1.Sayıyı giriniz:");
        sayi1 = Convert.ToInt32(Console.ReadLine());
        Console.Write("2.Sayıyı giriniz:");
        sayi2 = Convert.ToInt32(Console.ReadLine());
        sonuc = sayi1 - sayi2;
        Console.Write("Sonuç={0}", sonuc);
        break;
    case 'R':
        Console.Clear();
        Console.WriteLine("*****");
        Console.WriteLine("* Seçilen işlem ÇARPMA işlemi *");
        Console.WriteLine("*****");
        Console.Write("1.Sayıyı giriniz:");
        sayi1 = Convert.ToInt32(Console.ReadLine());
        Console.Write("2.Sayıyı giriniz:");
        sayi2 = Convert.ToInt32(Console.ReadLine());
        sonuc = sayi1 * sayi2;
        Console.Write("Sonuç={0}", sonuc);
        break;
    case 'B':
```

```

Console.Clear();
Console.WriteLine("*****");
Console.WriteLine("* Seçilen işlem ÇIKARMA işlemi *");
Console.WriteLine("*****");
Console.Write("1.Sayıyı giriniz:");
sayi1 = Convert.ToInt32(Console.ReadLine());
Console.Write("2.Sayıyı giriniz:");
sayi2 = Convert.ToInt32(Console.ReadLine());
switch (sayi2)
{
    default:
        sonuc = sayi1 / sayi2;
        Console.Write("Sonuç={0}", sonuc);
        break;
    case 0:
        Console.WriteLine("!!! SIFIRA BÖLME HATASI
!!!");
        break;
}
break;
default:
    Console.WriteLine("T-C-R-B 'den farklı bir değer
girdiniz...");
    break;
}

```

Not: Yukarıdaki bazı satırlar alta kaymıştır, programı yazarken kayan satırların tek satırda olmasına dikkat ediniz.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
➤ Klavyeden girilen sayıların tek mi çift mi olduğunu bulan programı yazınız.	➤ Bir sayının 2'ye bölümünden kalan sıfır(0) ise sayı çifttir. ➤ Mod alma (%) işlemi kullanınız.
➤ Klavyeden girilen iki metinden uzun olanını ekrana yazdıran programı yazınız.	➤ Metinsel ifadelerin uzunlukları <code>.length</code> özelliğiyle bulunur. ➤ Girilen metinlerin tek tek uzunluklarını bulup karşılaştırınız.
➤ Havanın “Güneşli, Yağmurlu ve Kar Yağışlı” olması durumlarına göre, kişinin Gömlek, Hırka veya Kazak giymesi hususunda uyarı mesajını Switch-Case kullanarak ekrana yazdırınız.	➤ Örnek: <i>Hava yağmurluysa; “Bugün hava yağmurlu, hırka giymelisin.”</i> ➤ Havanın her durumu için bir case yapısı oluşturup, kontrolü sağlayınız.
➤ Klavyeden girilen 3 sayının karşılaştırmasını yapıp, en büyük ve en küçük sayıları ekrana yazdıran programı yazınız.	➤ Örnek 1-5'ten faydalanabilirsiniz.
➤ Klavyeden girilen rakamı yazıyla ekrana yazdıran programı Switch-Case kullanarak yazınız.	➤ Her bir rakam için tek tek case yapısı oluşturup, kontrolü sağlayınız.
➤ Bir işyerinde çalışan işçilerin maaşlarına uygulanan kesinti miktarları çıkarıldıktan sonra ellerine geçecek olan net maaşlarını hesaplayan programı yazınız. Kesinti miktarları şu şekilde olacaktır. Kişinin maaşı; ➤ 1.000-2.500 TL arasındaysa 384 TL kesinti ➤ 2.501-4.000 TL arasındaysa 567 TL kesinti ➤ 4.000 TL ve üzeri ise 863 TL kesinti yapılacaktır.	➤ Örnek 1-6'da ki gibi birden fazla koşullu durumlara dikkat ediniz.

ÖLÇME VE DEĞERLENDİRME

Bu faaliyet kapsamında kazandığınız bilgileri, aşağıdaki soruları cevaplayarak belirleyiniz.

1. Aşağıdakilerden hangisi büyüktür ya da eşittir manasına gelen karşılaştırma operatörleridir?
A) \leq
B) \neq
C) $= >$
D) \geq
2. Aşağıda verilen bilgilerden hangisi **yanlıştır**?
A) break komutu içerisinde bulunduğu case satırını sonlandırmak için kullanılır.
B) default bloğu bir Switch-Case yapısında bulunmasa da olur.
C) Bir Switch -Case yapısında birden fazla aynı değere sahip Case ifadesi olabilir.
D) Akış farklı bir case ifadesine yönlendirilmek istenirse goto anahtar sözcüğü kullanılır.
3. $\text{if}(\text{sayi1} > 0) \parallel (\text{sayi1} < 5)$ ifadesindeki koşul aşağıdakilerden hangisidir?
A) sayi1 büyüktür sıfırdan ve sayi1 büyüktür 5'ten.
B) sayi1 büyüktür sıfırdan ve sayi1 küçüktür 5'ten.
C) sayi1 büyüktür sıfırdan veya sayi1 büyüktür 5'ten.
D) sayi1 büyüktür sıfırdan veya sayi1 küçüktür 5'ten.
4. Case bloğunu sonlandırmak için kullanılan anahtar kelime aşağıdakilerden hangisidir?
A) break
B) default
C) goto
D) return
5. Aşağıda verilen for döngüsü tanımlamalarından hangisinde döngü sonsuz bir döngüye girer?
A) `for(int i=0;i<100;i++)`
B) `for(int i=0;i<100;i--)`
C) `for(int i=100;i>0;i--)`
D) `for(int i=0;i<=100;i=i+5)`

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

Bu modül ile döngü kontrollerini kullanabilecek ve programlarınızda uygulayabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız

- Günümüzde kullandığımız sayaç çeşitlerini ve çalışma prensiplerini araştırınız.
- Bir bilgisayar programı vasıtasıyla aynı işlemi defalarca tekrar etmek yerine tek bir seferde nasıl gerçekleştiririz? Araştırınız.

2. DÖNGÜ DEYİMLERİ

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.

2.1. Döngü Çeşitleri

2.1.1. For Döngüsü

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

For döngüsünün kullanımı şu şekildedir;

➤ **Kullanımı:**

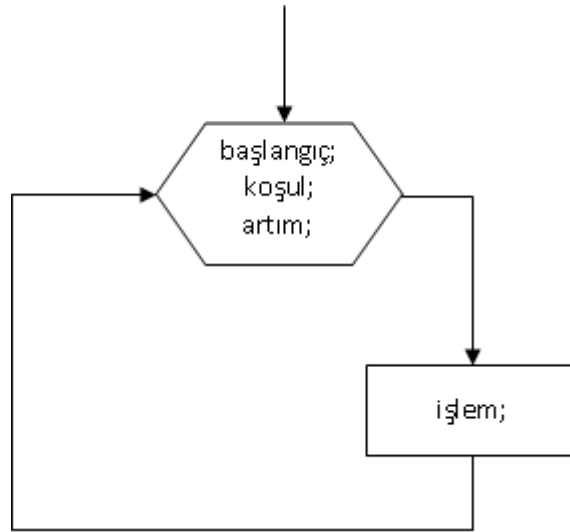
```
for(başlangıç;koşul;artım)
{
    yapılacak işler;
}
```

Başlangıç, döngü kontrol değişkeni olarak da ifade edilebilir. Döngü içerisinde bir sayacı 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.

Akış diyagramlarıyla for döngüsünün gösterimi de şu şekildedir.



Resim 0-1. For Döngüsü Akış Diyagramı

Şimdi basit bir örnekle for döngüsünün çalışmasını inceleyelim.

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

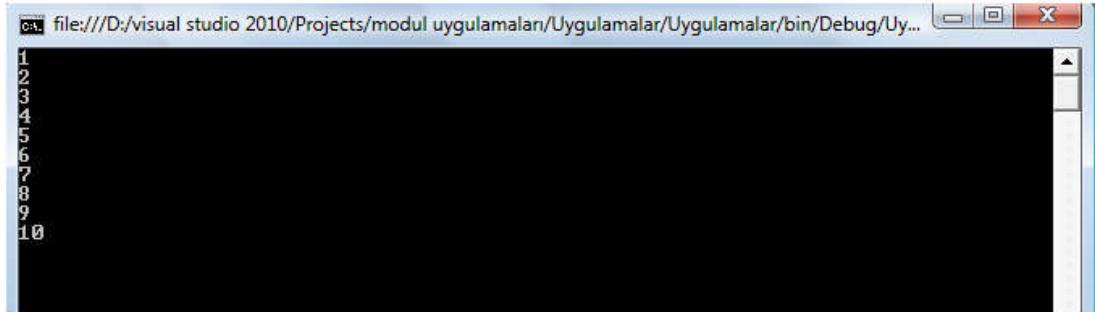
```
byte i;  
for(i=1;i<=10;i++)  
{  
    Console.WriteLine(i);  
}
```

Yukarıdaki kodu incelediğimizde;

- Döngü kontrol değişkenimiz olan i'ye 1 değerini atayarak başlangıç değerimizi,
- Döngümüzün ne zamana kadar döneceğini belirlediğimiz koşulumuzu $i \leq 10$ ifadesini,
- $i++$ ile de i değerimizi döngümüzün her dönüşünde 1 arttıracığımızı belirliyoruz.

Döngü her seferinde koşul kısmını kontrol eder ve buradaki koşul `false`(yanlış) olana kadar küme parantezleri ({ }) ile sınırlandırılan kod bloğunu çalıştırmaya devam edecektir.

Kod parçamızı çalıştırdığımızda aşağıdaki gibi bir ekran çıktısı alabiliriz.



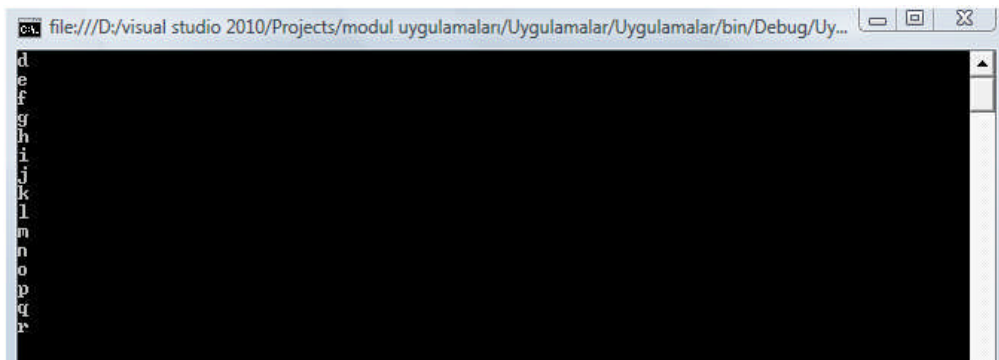
Resim 0-2. Örnek 2-1'in Ekran Çıktısı

For terimiyle döngü kurarken başlangıç değerimiz herhangi bir tam sayı olabileceği gibi char türünde bir değişkende olabilir.

Örnek 2-2: d'den r'ye kadar olan harfleri ekrana yazdırınız.

```
char i;  
for (i = 'd'; i <= 'r'; i++)  
{  
    Console.WriteLine(i);  
}
```

Yukarıdaki kodları çalıştırdığımızda da aşağıdaki gibi bir ekran çıktısıyla karşılaşırız.



Resim 0-3. Örnek 2-2'nin Ekran Çıktısı

For döngüsüyle sonsuz bir döngü oluşturulmak istenirse şu şekilde kodlanması gerekir;

```
for(;;)
{
    //.....
}
```

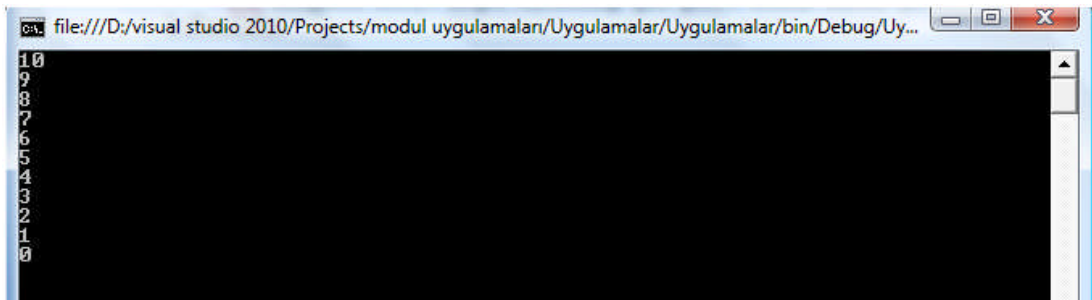
Uyarı: Bu şekilde bir sonsuz döngüyü bilgisayarınızda çalıştırdığınız zaman uygulamanız sonsuza kadar devam eder.

For döngüleri ileriye doğru sayabildiği gibi geriye dönük sayma işlemlerinde de kullanılırlar.

Örnek 2-3: 10'dan 0'a geriye doğru sayan ve sayıları ekrana yazdıran programı yazdırınız.

```
int i;
for (i=10;i>=0;i--)
    Console.WriteLine(i);
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran çıktısıyla karşılaşırız.



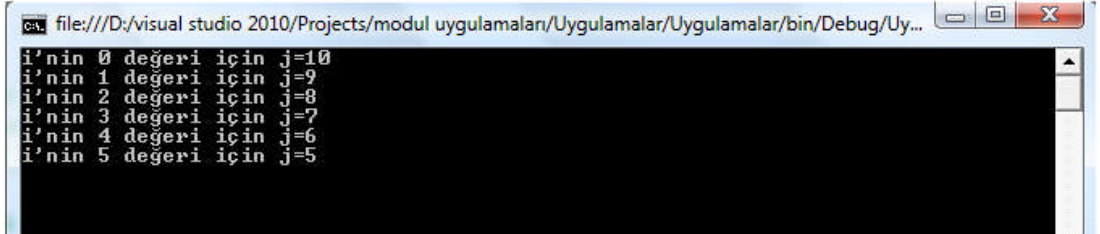
Resim 0-4. Örnek 2-3'ün Ekran Çıktısı

For döngüsü içerisinde birden fazla döngü kontrol değişken kullanma şansına da sahibiz.

Örnek 2-4: i=0'dan başlayacak ve j=10'dan başlayacak olan iki değişkendir. i ve j birbirine eşit olana kadar iki değişkenin durumlarını ekrana yazdıran programı yazınız.

```
int i, j;
for (i = 0, j = 10; i <= j; i++, j--)
    Console.WriteLine("i'nin "+i+" değeri için j="+j);
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran çıktısıyla karşılaşırız.



```
file:///D:/visual studio 2010/Projects/modul uygulamaları/Uygulamalar/Uygulamalar/bin/Debug/Uy...
i'nin 0 değeri için j=10
i'nin 1 değeri için j=9
i'nin 2 değeri için j=8
i'nin 3 değeri için j=7
i'nin 4 değeri için j=6
i'nin 5 değeri için j=5
```

Resim 0-5. Örnek 2-4'ün ekran çıktısı

Örnek 2-5: 0'dan klavyeden girilen sayıya kadar olan sayıların toplamını ekrana yazdıran programı yazınız.

```
int bitis,i,toplam;
Console.Write("Bir sayı giriniz:");
bitis = Convert.ToInt32(Console.ReadLine());
toplama = 0;
for (i = 0; i <= bitis; i++)
{
    toplama = toplama + i;
}
Console.WriteLine("Toplam={0}", toplama );
```

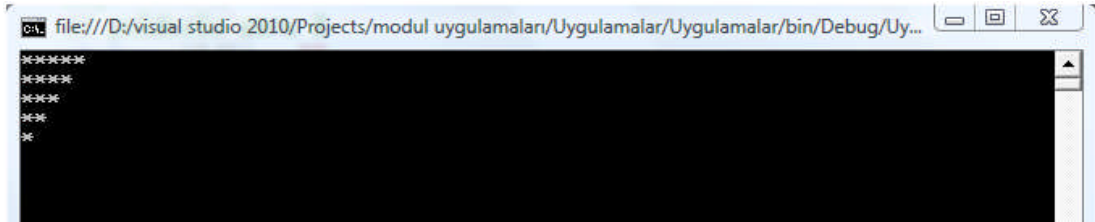
Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler	Ekran Çıktısı
15	
50	

Şimdiye kadar gördüğümüz örneklerde for döngüsünü hep tek başına kullandık. Aynı koşul kontrol mekanizmalarında olduğu gibi döngüler de iç-içe kullanılabilirler. İç-içe kullanılacak döngü sayılarında herhangi bir kısıtlama söz konusu değildir. İstedğimiz kadar sayıda döngüyü iç-içe kullanabiliriz

Sıradaki örneklerimizde de iç içe for döngüsü nasıl kullanılır buna göz atalım.

Örnek 2-6: Aşağıdaki ekran çıktısını verecek programın kodunu yazınız.



Resim 0-6. Örnek 2-6 Yıldız Sorusu

```
string yildiz = "";
for (int i = 1; i <= 5; i++)
{
    for (int k = 0; k <= 5-i; k++)
        yildiz = yildiz + "*";
    Console.WriteLine(yildiz);
    yildiz = "";
}
```

Örnek 2-7: 1'den 10'a kadar olan sayılar için çarpım tablosunu ekrana yazdıran programı yazınız.

```
int i,k;
for (i = 1; i <=10; i++)
{
    Console.WriteLine("-{0} ve Katları-",i);
    Console.WriteLine("-----");
    for (k = 1; k <= 10; k++)
    {
        int carpim = i * k;
        Console.WriteLine("{0} x {1} = {2}", i, k, carpim);
    }
    Console.WriteLine("-----");
}
```

2.1.2. While Döngüsü

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

Genel yazım şekli şöyledir.

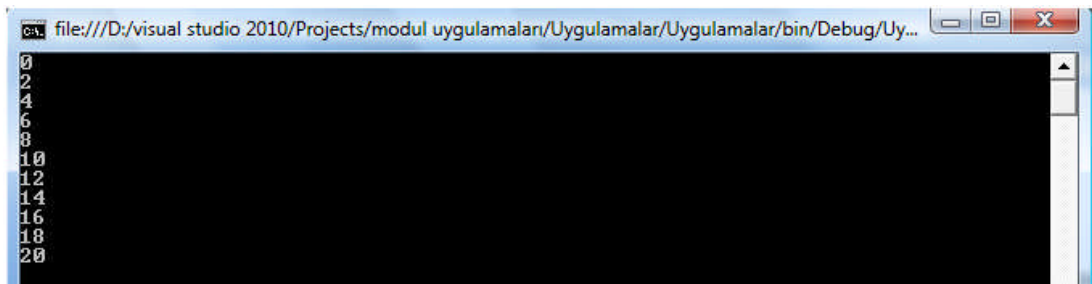
➤ Kullanımı:

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

Örnek 2-8: 0'dan 20'ye kadar olan çift sayıları ekrana yazdırınız.

```
int i=0;
while (i <= 20)
{
    Console.WriteLine(i);
    i = i + 2;
}
```

Kod parçamızı çalıştırdığımızda aşağıdaki gibi bir ekran çıktısı alabiliriz.



Resim 0-7. Örnek 2-8'in Ekran Çıktısı

Örnek 2-9: Bilgisayara rastgele ürettiğimiz 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");
    }
    Console.WriteLine("Kalan tahmin hakkınız:{0}", hak);
}
if(hak==0)
    Console.WriteLine("Tahmin hakkınız bitti. Sayımız:{0}",tutulan);
```

Yukarıdaki programda karşımıza çıkan Random komutu bize belirtilen bir aralıkta rastgele sayı üretmemizi sağlayan bir komuttur. Programımızda bizler 1-50 arasında bir sayı üretmesini sağladık.

Bir diğer dikkat etmemiz gereken komutumuz da break komutudur. Aynı bir önceki öğrenme faaliyetinde gördüğümüz select-case yapısındaki gibi sonlandırma işine yarayan break komutunun While döngüleriyle birlikte kullanımı oldukça yaygındır. Döngülerden istenilen koşulun sağlanmasını beklemeden çıkmak için kullanılır.

2.1.3. 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.

➤ Kullanımı:

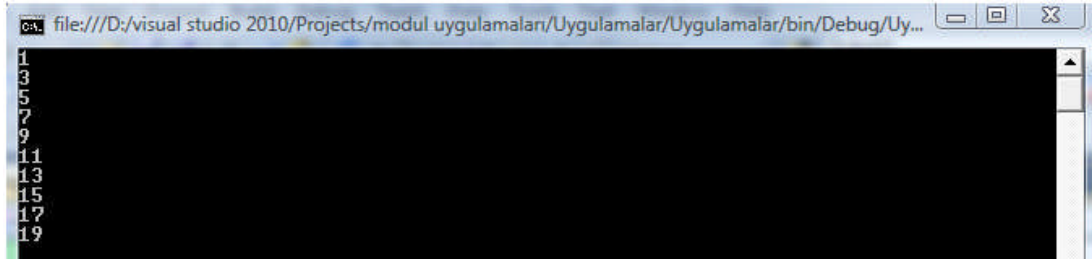
```
do
{
    yapılacak işler;
}
while(koşul);
```

Örnek 2-10: 1'den 20'ye kadar olan tek sayıları ekrana yazdırınız.

```
int i=1;
do{
    Console.WriteLine(i);
    i = i + 2;
} while (i < 20);
```

Yukarıdaki örnek kod incelendiğinde koşulun do-while döngüsünün en sonunda kontrol edildiğini görebilirsiniz.

Kod parçamızı çalıştırdığımızda aşağıdaki gibi bir ekran çıktısı alabiliriz.



Resim 0-8. Örnek 2-10'un Ekran Çıktısı

2.1.4. Foreach Döngüsü

foreach, dizi (Array) ve koleksiyon (collection) tabanlı nesnelerin elemanları üzerinden ilerleyen bir döngüdür.

Genel kullanım şekli şöyledir;

➤ **Kullanımı:**

```
foreach(tip değişken in koleksiyon)
{
    yapılacak işler;
}
```

- **Tip:** buradaki tip koleksiyonun veri tipi ile aynı veya uyumlu olmak zorundadır.
- **Değişken:** foreach döngüsü içerisinde koleksiyonda bulunan sıradaki elemanı temsil eder.
- **Koleksiyon:** ArrayList ya da dizi gibi aynı tip verileri barındıran koleksiyon.

Uyarı: Bir sonraki öğrenme faaliyeti olan Diziler konusunda foreach döngüsünün kullanımına yönelik daha fazla örnek gösterilecektir. Sadece ön bilgi amacıyla aşağıdaki örneği inceleyiniz.

Örnek 2-11: Gunler isimli dizi içerisindeki elemanları ekrana yazdıran programı yazınız.

```
string[] gunler=new string[7];
gunler[0]="Pazartesi";
gunler[1]="Salı";
gunler[2]="Çarşamba";
gunler[3]="Perşembe";
gunler[4]="Cuma";
gunler[5]="Cumartesi";
gunler[6]="Pazar";
foreach (string gun in gunler)
{
    Console.WriteLine(gun);
}
```

Bu kısımda dizinin tanımlanması ve içeriğine değer atama işlemi gerçekleştirilmektedir.

Burada, foreach, dizi boyunca her seferinde bir elemanı adımlar. Dizinin her bir elemanının değerini gun adındaki string değişkenine aktarır ve daha sonra döngüyü başlatır.

Foreach döngüsü ile ilgili dikkat edilmesi gereken bir önemli nokta ise, koleksiyondaki elemanın (örnek 2-11'deki gun) değerini değiştirememenizdir, aksi takdirde kodunuz derlenmeyecektir:

```
foreach (string gun in gunler)
{
    gun= gun+" günü";
    Console.WriteLine(gun);
}
```

Yukarıdaki kullanım hatalı bir kullanımdır ve program derlenmeyecektir.

Eğer bir koleksiyondaki elemanlar aracılığıyla döngüyü başlatmaya ihtiyaç duyulursa ve onların değerleri değiştirilecekse, foreach döngüsü yerine bir for döngüsü kullanımına ihtiyaç duyulacaktır.

2.2. Jump (Dallanma – Atlama) Komutları

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

Bunlar;

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

2.2.1. Break Anahtar Sözcüğü

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.

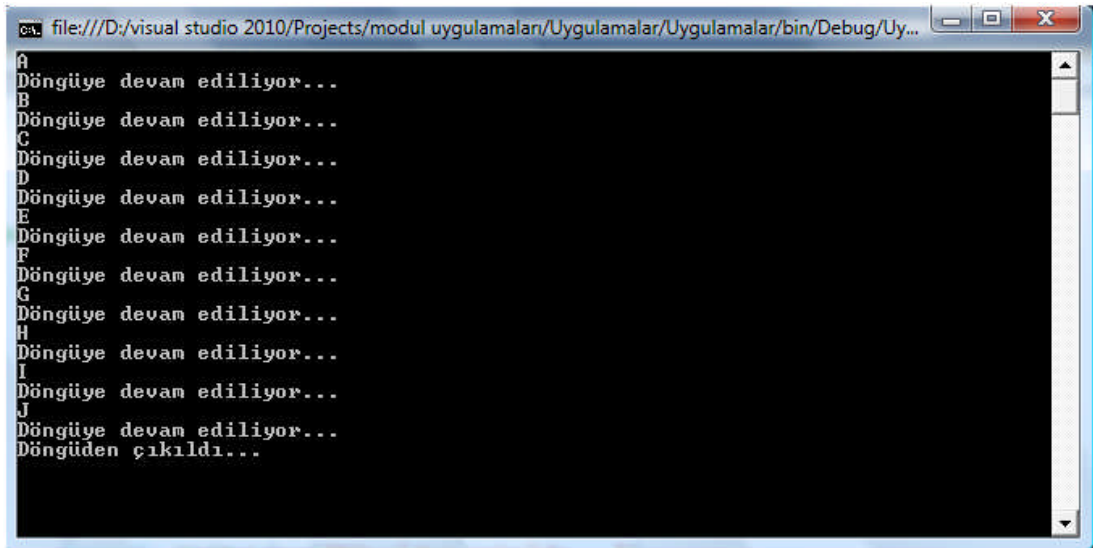
Break anahtar sözcüğü döngü bloklarının ya da switch bloklarının dışında kullanılamazlar.

Örnek 2-12: 'A' harfinden başlayıp 'Z'ye kadar devam eden bir döngü de 'K' harfine gelindiğinde döngüden çıkan programın kodunu yazınız.

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

Yukarıdaki örnek incelendiğinde döngümüz 'A' harfinden başlayarak teker teker harfleri yazmaktayken 'K' harfine geldikten sonra `break` komutuyla karşılaşır. Bu komutu gören program o anda içerisinde bulunduğu döngüyü sonlandırır ve programın akışına kaldığı yerden devam eder.

Aşağıdaki Resim 2-11'den de görüleceği üzere programımızdaki döngüde i değeri 'K' ya eriştiğinde döngü sonlandırılıp, programın kaldığı yerden devam etmesi sağlanıyor.



```
file:///D:/visual studio 2010/Projects/modul uygulamaları/Uygulamalar/Uygulamalar/bin/Debug/Uy...
A
Döngüye devam ediliyor...
B
Döngüye devam ediliyor...
C
Döngüye devam ediliyor...
D
Döngüye devam ediliyor...
E
Döngüye devam ediliyor...
F
Döngüye devam ediliyor...
G
Döngüye devam ediliyor...
H
Döngüye devam ediliyor...
I
Döngüye devam ediliyor...
J
Döngüye devam ediliyor...
Döngüden çıkıldı...
```

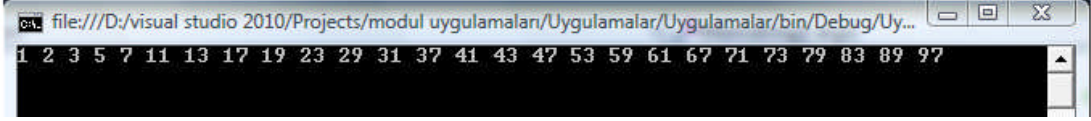
Resim 0-9. Örnek 2-12'nin Ekran Çıktısı

Örnek 2-13: 0'dan 100'e kadar sayılardan asal olanları ekrana yazdıran programın kodunu yazınız.

```
for(int i=1; i < 100; i++){
    bool asalMi = true;
    //Sayının asal olup olmadığı kontrol ediliyor
    for(int j=2; j < i; j++)
    {
        if(i % j == 0){
            asalMi = false;
            break;
        }
    }
    // asal olan sayılar ekrana yazdırılıyor
    if(asalMi)
        Console.Write(i + " ");
}
```

Yukarıdaki örnekte de `break` komutu `if` bloğundan çıkmak için kullanılmıştır.

Kodları çalıştırdığımızda karşımıza aşağıdaki gibi bir ekran çıktısı gelir.



Resim 0-10 Örnek 2-13'ün Ekran Çıktısı

2.2.2. Continue Anahtar Sözcüğü

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.

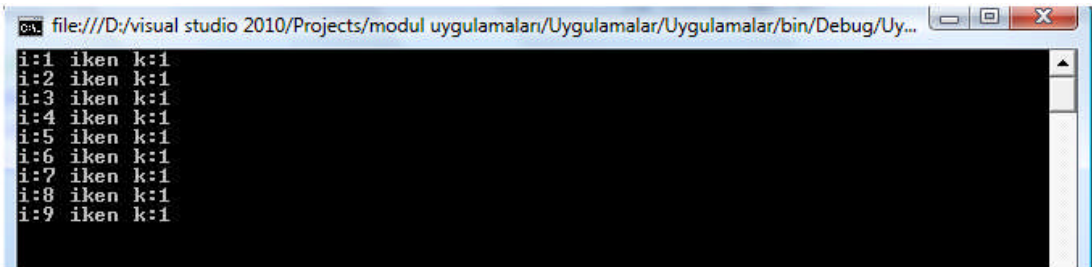
Şimdiki örneğimizde continue anahtar sözcüğünün kullanımını inceleyelim.

Örnek 2-14: Continue anahtar sözcüğünün kullanımı.

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

Yukarıdaki kod parçasını incelediğimizde döngümüzün koşulu, `i`'nin 10'dan küçük olan değerleri sağlanması durumunda `TRUE` değerini almasıdır.

`i` ve `k` değişkenlerimizin değerleri döngümüz içerisinde `i++` ve `k++` ifadeleriyle artırılmaktadır. Lakin programımızın ekran çıktısını (Resim 2-11) incelediğimizde yalnızca `i` değişkeninin değerinin artırıldığını görmekteyiz. Sebebi ise `k` değişkeninin değerinin `continue` anahtar sözcüğünden sonra artırılmasıdır. Program `continue` anahtar sözcüğünü görünce o satırdan tekrar döngünün başına döner ve böylece `k` değişkeninin değeri değiştirilemez.



Resim 0-11. Örnek 2-14'ün Ekran Çıktısı

2.2.3. Goto Anahtar Sözcüğü

Goto anahtar sözcüğü, koşulsuz atlama komutudur. Programın akışı esnasında goto anahtar sözcüğüyle karşılaşıldığı anda program goto ile belirlenen etiketin bulunduğu satıra atlama işlemi gerçekleştirir.

Goto anahtar sözcüğünün kullanımı nesne yönelimli programlama tekniğinde pek uygun görülmesi de bazı durumlarda (örneğin switch deyiminde case ifadeleri arasında dolaşma) gerekebilir.

Uyarı: Goto anahtar sözcüğü ile bir döngü ve koşul bloğu içerisine dallanma işlemi gerçekleştirilemez.

Sıradaki örneğimizde goto anahtar sözcüğünün kullanımını inceleyelim;

Örnek 2-15: Goto anahtar sözcüğünün kullanımı.

```
int sayi1 = 10;
int sayi2 = 20;
Console.WriteLine("{0} + {1} = {2} ", sayi1, sayi2, sayi1 +
sayi2);
goto son;
Console.WriteLine("{0} x {1} = {2} ", sayi1, sayi2, sayi1 *
sayi2);
son:
    Console.ReadLine();
```

Yukarıdaki örnek incelendiğinde program ilk goto satırıyla karşılaşınca “son” isimli etiketin bulunduğu satıra dallanma işlemini gerçekleştirir ve çarpma işleminin gerçekleştirdiği satırı (Console.WriteLine("{0} x {1} = {2} ", sayi1, sayi2, sayi1 * sayi2);) atlayarak dallanma işlemini gerçekleştirir.

2.2.4. Return Anahtar Sözcüğü

Return anahtar sözcüğü, metotlardan geriye bir değer döndürmek için kullanılır. Metotlarla ilgili ayrıntılı bilgiyi ve return anahtar sözcüğünün kullanımını bir sonraki Alt Programlar Modülü içerisinde inceleyeceğiz.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
<p>★ ★★ ★★★ ★★★★ ★★★★★</p> <p>Yandaki şekildeki gibi yıldızları klavyeden girilen sayı kadar ekrana yazdıran programın kodunu yazınız.</p>	<p>Döngünün bitiş değerini klavyeden girilen sayı olacak şekilde atayınız. Koşulunuzu belirlerken bunu göz önünde bulundurunuz.</p>
<p>Verilen bir mesajı istenilen sayıda ekrana yazdıran programı yazınız.</p>	<p>Girilen mesaj bilgisini bir değişkene atayınız İstenilen adede göre de döngünüzü kurunuz.</p>
<p>Klavyeden girilen sayının faktöriyelini hesaplayan programın kodunu yazınız.</p>	<p>Örneğin klavyeden 5 değeri girilmiş olsun; $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$ Döngünün bitiş değerini klavyeden girilen sayı olacak şekilde atayınız. Döngünüzü örneğe göre yazınız.</p>
<p>1’den 50’ye kadar olan sayılardan tek sayıların karesini çift sayıların da küpünü alıp ekrana yazdıran programın kodunu yazınız.</p>	<p>Bir sayının üssünü hesaplamak için; <code>Math.Pow(sayı, us)</code> fonksiyonunu kullanınız.</p>
<p>Kullanıcıdan klavyeden önceden belirlenmiş bir PIN kodunu girmesini isteyiniz. 3 defa PIN numarasını hatalı girerse “Uygulamanız bloke olmuştur” mesajı, kullanıcı 3 deneme hakkı içinde doğru olarak PIN numarasını girerse “Hoş geldiniz” mesajı veren programın kodunu while döngüsü ve yazınız.</p>	<p>Sayı tahmin örneğinden yapısal olarak faydalanabilirsiniz.</p>
<p>Klavyeden girilen 10 sayı içerisinde</p> <ol style="list-style-type: none"> 100-200 arasındaki sayıların adedini 100’den küçük sayıların toplamını 200’den büyük sayılardan da 4’e kalansız bölünebilenlerini <p>ekrana yazdıran programı do-while döngüsü ve if komutlarıyla yazınız.</p>	<p>Bir sayaç değişkeni belirleyiniz ve girilen sayı miktarını tutunuz. Toplam, bolunenSayisi ve adet değişkenlerinizi oluşturunuz. Girilen sayının yandaki hangi duruma uyacağını belirleyerek gerekli işlemleri gerçekleştiriniz.</p>
<p>Klavyeden girilen sayının asal olup olmadığını kontrol eden programı yazınız.</p>	<p>Yalnızca 1 ve kendisine bölünebilen sayılar asal sayılardır.</p>

ÖLÇME VE DEĞERLENDİRME

Bu faaliyet kapsamında kazandığınız bilgileri, aşağıdaki soruları cevaplayarak belirleyiniz.

1. Aşağıdakilerden hangisi bir döngü deyimi değildir?
A) IF.....ELSE
B) FOR
C) FOREACH
D) DO...WHILE
2. Aşağıda for döngüsü ile ilgili verilen kullanımlardan hangisi yanlıştır?
A) `for (i=0 ; i<100 ; i=i+5)`
B) `for (i=0 , j=100 ; i<j ; i++ , j--)`
C) `for (; ;)`
D) `for (i="A" ; i<"Z" ; i++)`
3. İç-içe döngüler ile ilgili aşağıda verilenlerden hangisi doğrudur?
A) İç-içe en fazla 2 adet döngü kullanabiliriz.
B) Bir if koşulu içerisinde do-while döngüsü kullanamayız.
C) İç-içe kullanacağımız döngü sayılarında herhangi bir kısıtlama söz konusu değildir.
D) Hem do-while hem de while döngüsünü iç-içe kullanamayız.
4. Aşağıdaki anahtar sözcüklerden hangisi belirtilen bir komut satırına dallanma için kullanılır?
A) continue
B) goto
C) break
D) return
5. Dallanma komutlarıyla ilgili aşağıda verilenlerden hangisi doğrudur?
A) Dallanma işlemi bir programda yalnızca ileriye dönük yapılabilir.
B) Dallanma işlemi bir programda yalnızca geriye dönük yapılabilir.
C) Dallanma işlemi bir programda hem ileri hem de geriye dönük yapılabilir.
D) Dallanma komutlarıyla ile bir döngünün içerisine atlama işlemi yapılabilir.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-3

AMAÇ

Bu modül ile dizilerle çalışabilecek ve programlarınızda uygulayabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız

- Aynı veri türüne sahip birden fazla veriyi tek bir seferde nasıl saklarız? Araştırınız.
- Farklı veri türlerine sahip birden fazla veriyi tek bir seferde nasıl saklarız? Araştırınız.

3. DİZİLER

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. Bu modül içerisinde dizi oluşturma, diziye değer girme, diziye yazdırma, dizilerde arama, dizilerde sıralama, dinamik diziler yapmayı öğreneceğiz.

Bir dizi, aynı tipe ait bir miktar eleman içeren bir veri yapısıdır. Dizileri hep bir arada yer alan değişkenler listesi gibi düşünebiliriz. Örneğin 5 tane sebze ismini tek bir liste içerisinde tutmak istersek bir dizi kullanabiliriz.

3.1. Dizi Oluşturma

Bir dizi, boş parantezler ve bir değişken ismi tarafından takip edilen dizi içindeki elemanların tipini tanımlayarak bildirilir;

- **Tanımlanması:**

1.Yol:

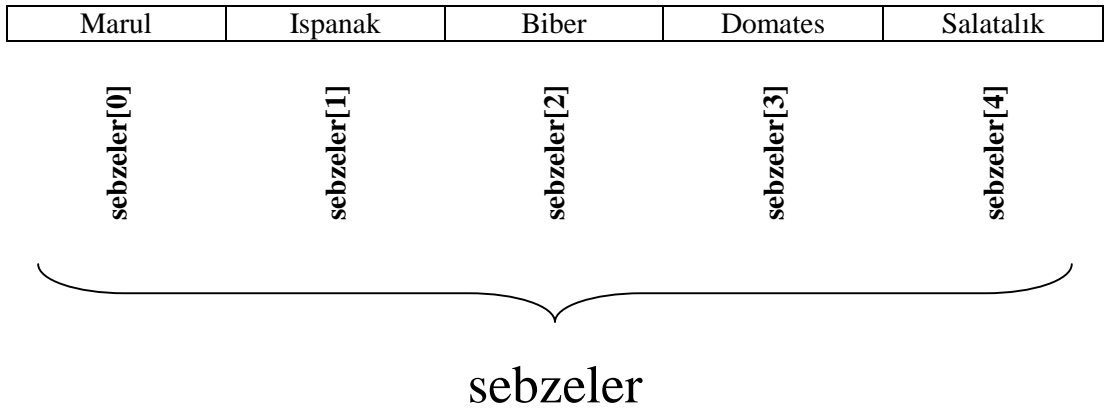
```
tip[] dizi-ismi=new tip[eleman-sayısı];
```


Burada tip, dizinin temel veri türünü belirlemek için kullanılır. Temel veri türü, dizi içerisinde saklanacak olan verinin türünü belirler. Tip ifadesinden hemen sonra köşeli parantezler ([]) geldiğine dikkat edin. Köşeli parantezler, burada tek boyutlu bir dizinin tanımlandığını belirtirler. Eleman-Sayısı ile de dizinin içerisine ne kadar eleman tutulacağını belirtir.

Örneğin, metinsel ifade (string) türde elemanlarını içeren bir dizi aşağıdaki gibi bildirilir;

```
string[] sebzeler=new string[5];
```

Yukarıda tanımlaması gerçekleştirilen *sebzeler* isimli dizi içerisinde 5 adet *string* türünde veri tutabiliriz. Oluşturmuş olduğumuz dizi kavramsal olarak şu şekilde görünür;



Resim 0-1. sebzeler Dizisinin Kavramsal Görünüşü

Görmüş olduğumuz şekilde dizi tanımlaması gerçekleştirilebileceği gibi, aşağıdaki gibi de dizi tanımlama işlemi gerçekleştirebilir.

2.Yol:

```
int[] notlar;  
rakamlar=new int[10];
```

Eğer tam sayı (integer) türünde eleman içeren bir dizi tanımlamak istiyorsak yukarıdaki yolu da izleyebiliriz.

Yukarıdaki tanımlamada da *rakamlar* isimli dizi içerisinde 10 adet *int* türünde veri tutabiliriz.

Dizi tanımlama işlemlerinde üçüncü bir yol da dizinin ilk değerlerinin küme parantezleri ({}) içerisinde dizinin istenilen boyutu kadar başlangıçta belirtilmesiyle tanımlanmasıdır;

3.Yol:

```
int[] notlar={65,76,85};
```

Bu tanımlama yöntemiyle tek boyutlu 3 elemandan oluşan *int* türünde bir dizi tanımlamış olduk.

Dikkat ederseniz yukarıdaki dizinin tanımlanması esnasında herhangi bir boyut (eleman sayısı) belirtilmedi. Bu durumlarda ilk anda kaç adet eleman girişi yapıldıysa dizinin boyutu da o kadar olur.

İstenirse aşağıdaki gibi dizinin boyutu belirtilerek de tanımlama gerçekleştirilebilir;

```
int[] notlar=new int[3]{65,76,85};
```

3.2. Diziye Değer Girme

Bir dizi tanımlandıktan sonra sıra o diziye değer girmeye gelir. Bir diziye değer girişleri tanımlama esnasında yapılabildiği gibi, programın akışı esnasında da gerçekleştirilebilir.

Dizi tanımlandıktan sonra, dizinin her bir elemanı için indeks değerleriyle elemana erişerek değer ataması şu şekildedir;

```
int[] notlar=new int[3];  
notlar[0]=65;  
notlar[1]=76;  
notlar[2]=85;
```

Bir diğer yöntemde bir önceki konuda gördüğümüz, dizi oluşturulurken değer girmeyi tekrar hatırlarsak;

```
int[] notlar=new int[3]{65,76,85};
```

veya

```
int[] notlar= {65,76,85};
```

şeklinde tanımlama esnasında değer girişi yapabiliriz.

Yukarıda her iki örnekte de verilen notlar dizisinin kavramsal gösterimi şu şekildedir;

notlar[0]	notlar[1]	notlar[2]
65	76	85

Resim 0-2. notlar Dizinin Kavramsal Görünüşü

Char (karakter) türündeki bir dizinin ilk kullanımına hazırlanması da şu şekillerde gerçekleştirilir;

```
char[] harfler = new char[]{'r','T','h','Y'};
```

veya

```
char[] harfler=new char[4];  
harfler[0] = 'r';  
harfler[1] = 'T';  
harfler[2] = 'h';  
harfler[3] = 'Y';
```

String (metinsel) türdeki bir dizinin ilk kullanımına hazırlanması da şu şekillerde gerçekleştirilir;

```
string[] sebzeler = new string[] { "Marul", "Ispanak", "Biber",  
"Domates", "Salatalık" };
```

veya

```
string[] sebzeler=new string[5];  
sebzeler[0] = "Marul";  
sebzeler[1] = "Ispanak";  
sebzeler[2] = "Biber";  
sebzeler[3] = "Domates";  
sebzeler[4] = "Salatalık";
```

Bir dizi içerisindeki elemanlara tek tek dizi indeksi yardımıyla erişilebilir. Dizi indeksi (array index), bir elemanın dizi içerisindeki konumunu ifade eder. Genellikle programlama dillerinde dizilerin ilk elemanının indeksi sıfır (0)'dır. Örneğin 10 elemanlı bir dizi varsa, bu dizinin indeks numaraları 0-9 arasındadır.

Dizinin tüm elemanlarına değil de bir kısmına değer girişi yapmamız isteniyorsa, ilgili değerın barındırılacağı indeksine değer atama işlemi gerçekleştirilir.

```
int[] plakalar=new int[10];  
plakalar[2] = 43;  
plakalar[5] = 16;  
plakalar[6] = 66;  
plakalar[9] = 6;
```

Yukarıda tanımlanan plakalar isimli dizinin 2,5,6 ve 9 numaralı indeks konumlarına değer ataması gerçekleştirilmiş ve kavramsal görüntüsü şu şekilde oluşmuştur.

plakalar[0]	0
plakalar[1]	0
plakalar[2]	43
plakalar[3]	0
plakalar[4]	0
plakalar[5]	16
plakalar[6]	66
plakalar[7]	0
plakalar[8]	0
plakalar[9]	6

Başlangıçta eleman sayısı belli fakat değerleri daha sonra girilecekse tanımlama şu şekilde yapılabilir. Örneğin sebzeler dizisinin eleman sayısı 5 değil de 8 olsun, başlangıçta da 4 adet değer girilecek olsun;

```
string[] sebzeler = new string[] { "Marul", "Ispanak", "Biber", "Domates", "", "", "", "" };
```

veya

```
string[] sebzeler = new string[] { "Marul", "Ispanak", "Biber", "Domates", null, null, null, null };
```

Yukarıda tanımlaması gerçekleştirilen sebzeler isimli dizinin kavramsal görüntüsü de şu şekildedir;

sebzeler[0]	Marul
sebzeler[1]	Ispanak
sebzeler[2]	Biber
sebzeler[3]	Domates
sebzeler[4]	
sebzeler[5]	
sebzeler[6]	
sebzeler[7]	

Dikkat ederseniz int türündeki dizilerde boş kalan dizi hücrelerine sıfır(0) değeri, string türündeki dizilerde de boş kalan hücrelere boş (null-“”) değer yüklenmektedir.

Örnek 3-1: Rakamlar isimli dizi içerisinde 0-9 arası rakamları tersten bir döngü yardımıyla yükleyiniz.

```
int[] rakamlar = new int[10];
int i;

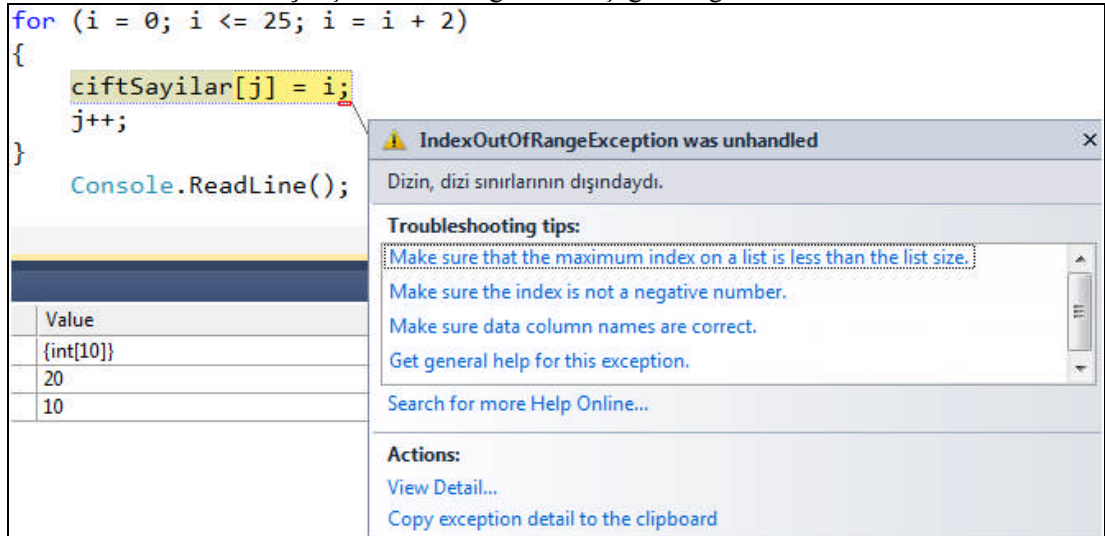
for (i = 0; i <= 9; i++)
    rakamlar[i] = 9 - i;
```

Dizilerle çalışırken dikkat etmemiz gereken noktalardan en önemlisi dizi sınırlarına sadık kalmaktır. Eğer 10 elemanlı bir dizi tanımlamışsak ve bu dizi tanımlanırken belirlenen eleman sayısından fazla sayıda eleman değeri atamaya çalışırsak hata alırız.

Örnek 3-2: 10 elemanlı çiftSayılar isimli bir dizi tanımlayınız. İçerisine 0-25 arasındaki çift sayıları ekleyen kodu yazınız.

```
int[] çiftSayilar = new int[10];
int i, j;
j = 0;
for (i = 0; i <= 25; i = i + 2)
{
    çiftSayilar[j] = i;
    j++;
}
```

Yukarıdaki kodu çalıştırmak istediğimizde aşağıdaki gibi bir hata alırız.



Resim 0-3. Örnek 3-2’deki Dizi Sınırı Aşım Hatası

Resim 3-3’te görülen hata mesajı dizi sınırının aşıldığını belirten hata mesajıdır.

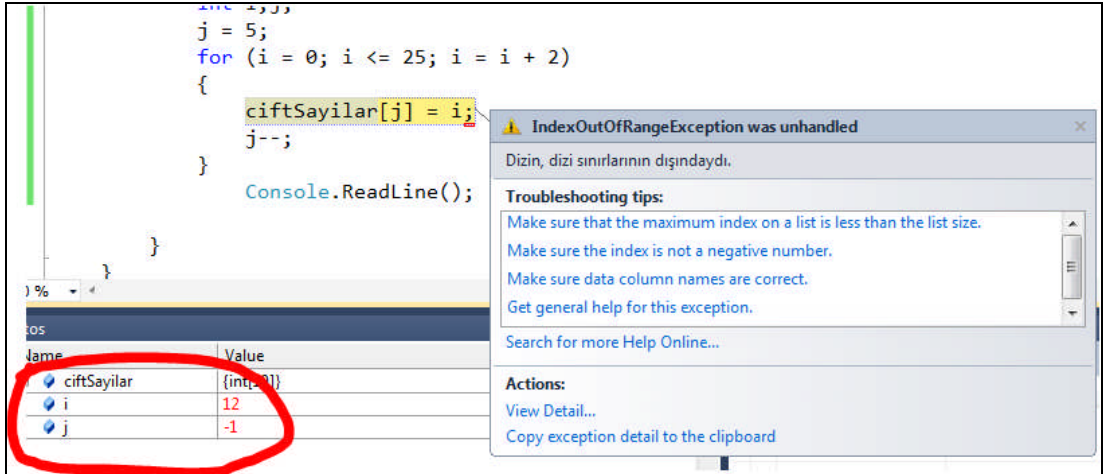
Örnek3-2’de verilen kodlar incelendiğinde çiftSayılar isimli dizinin başlangıçta 10 elemanlı bir dizi olarak tanımlandığı görülmektedir. Ancak 0-25 arasındaki çift sayılar bu diziye bir for döngüsü yardımıyla (i değişkeninin değeri olarak) eklenirken, dizinin 10. elemanı eklendikten sonra (j değişkeninin değerinin 10 olmasından sonra) döngünün devam etmesi sebebiyle 11. eleman eklemeye çalışılınca Resim 3-3’teki “Dizi Sınırlarının Dışı” hata mesajı ile karşılaşırız.

Dizi eleman sayısının üst limitini aşmak gibi alt sınır değerinin altına girilmeye çalışılması da hata mesajı almamıza sebebiyet verir.

Örnek 3-3: Örnek 3-2’deki kodlarda biraz değişiklik yapıp programın dizinin alt sınır değerinden daha düşük indeksine erişmeye çalışalım;

```
int[] çiftSayılar = new int[10];
int i, j;
j = 5;
for (i = 0; i <= 25; i = i + 2)
{
    çiftSayılar[j] = i;
    j--;
}
```

j değişkenimizin başlangıç değerini 5 yapıp bu sefer döngümüz vasıtasıyla elemanlarımızı yazdırmaya çalıştığımızda, j indeksimizin j-- ifadesiyle döngü içerisinde her seferinde değerinin 1 azaltıldığını görüyoruz. Ne zaman ki j’nin değeri 0’ın altında negatif bir sayıya ulaştığında, programımız Resim 3-4’teki gibi bir hata mesajını bizlere verir.



Resim 0-4. Dizi Sınır Değeri Aşımı Hatası

3.3. Diziyi Yazdırma

Bir dizinin elemanlarına indeks numaraları vasıtasıyla erişebileceğimizi daha önce de bahsetmiştik. Erişilen bu elemanlarla ilgili işlemlerden birisi de ekrana yazdırma işlemidir. Erişilen değerlerinin ekrana yazdırılması işlemi şu şekilde gerçekleştirilir;

```
int[] plakalar=new int[10];  
plakalar[2] = 43;  
plakalar[5] = 16;  
plakalar[6] = 66;  
plakalar[9] = 6;
```

Aşağıdaki işlemler yukarıda tanımlanmış olan diziye göre gerçekleştirilmektedir.

```
Console.WriteLine(plakalar[2]);  
Console.WriteLine(plakalar[3]);  
Console.WriteLine(plakalar[4]);  
Console.WriteLine(plakalar[5]);
```

Yukarıdaki kod parçası çalıştırıldığında ekrana

```
43  
0  
0  
16
```

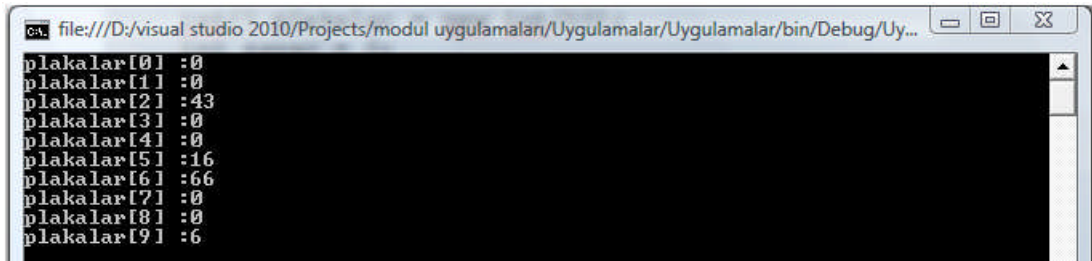
değerleri yazılır.

Bir dizi içerisindeki tüm değerleri ekrana yazdırmak istiyorsak döngü kullanmak gayet mantıklı olacaktır. Örneğin 200 elemanlı bir dizinin tüm elemanlarını ekrana yazdırmak istersek alt alta 200 satır kod yazmamız mümkün değildir.

Örnek 3-4: Plakalar isimli dizi içerisinde bulunan bütün elemanları ekrana yazdıran programın kodunu yazınız.

```
int[] plakalar = new int[10];  
int sayac = 0;  
plakalar[2] = 43;  
plakalar[5] = 16;  
plakalar[6] = 66;  
plakalar[9] = 6;  
foreach (int note in plakalar)  
{  
    Console.WriteLine("plakalar["+sayac+"] : "+note);  
    sayac++;  
}
```

Yukarıdaki kodlar çalıştırıldığı zaman aşağıdaki gibi bir ekran çıktısı ile karşılaşır;



```
file:///D:/visual studio 2010/Projects/modul uygulamaları/Uygulamalar/Uygulamalar/bin/Debug/Uy...
plakalar[0] :0
plakalar[1] :0
plakalar[2] :43
plakalar[3] :0
plakalar[4] :0
plakalar[5] :16
plakalar[6] :66
plakalar[7] :0
plakalar[8] :0
plakalar[9] :6
```

Resim 0-5. Örnek3-4'ün Ekran Çıktısı

Örnek 3-5: Örnek3-4'teki plakalar dizisini bir de for döngüsüyle ekrana yazalım.

```
int[] plakalar = new int[10];
int sayac = 0;
plakalar[2] = 43;
plakalar[5] = 16;
plakalar[6] = 66;
plakalar[9] = 6;
for(sayac=0;sayac<10;sayac++)
    Console.WriteLine("plakalar["+sayac+"] :"+plakalar[sayac]);
```

Yukarıdaki kod parçası çalıştırıldığı zaman karşımıza Resim 3-5'te ekran görüntüsünün aynısı karşımıza çıkar.

Ancak, burada dikkat etmemiz gereken husus for döngüsünün bitiş değerini dizimizin eleman sayısını bildiğimiz için buna göre belirledik.

3.4. Bazı Dizi Özellikleri ve Metotları

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

3.4.1. Length

Dizinin saklayabileceği toplam eleman sayısını veren ve int türünde bir değer veren özelliktir.

➤ **Kullanımı:**

```
dizi-adi.Length;
```

Örnek 3-6: ciftSayilar isimli dizinin içerisinde kaç adet eleman olduğunu ekrana yazan programın kodunu yazınız.

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


Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsü alırız;



Resim 0-6. Örnek 3- 6'nın Ekran Çıktısı

3.4.2. Clear(dizi,baslangic,adet)

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.

➤ Kullanımı:

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

Örnek 3-7: Yeni tanımladığımız ve içeriğini oluşturduğumuz bir dizinin elemanlarının nasıl silindiğini inceleyelim;

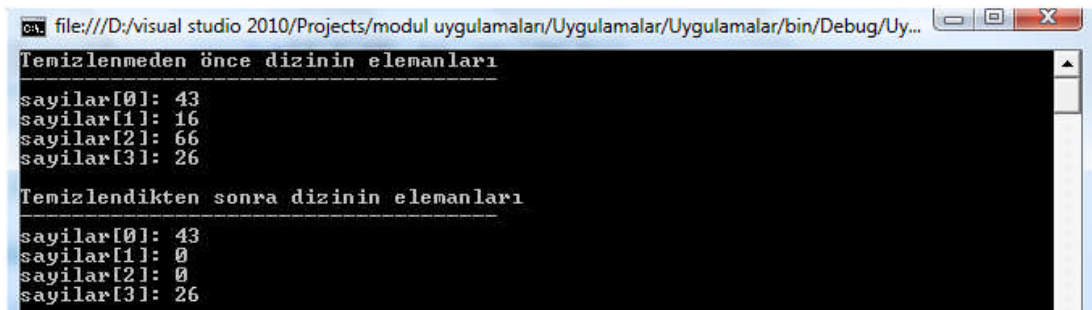
```
int[] sayilar = new int[] { 43, 16, 66, 26 };
Console.WriteLine("Temizlenmeden önce dizinin elemanları");
Console.WriteLine("-----");
for (int i = 0; i < sayilar.Length; i++)
    Console.WriteLine("sayilar[{0}]: {1}", i,sayilar[i]);

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]);
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsü alırız;



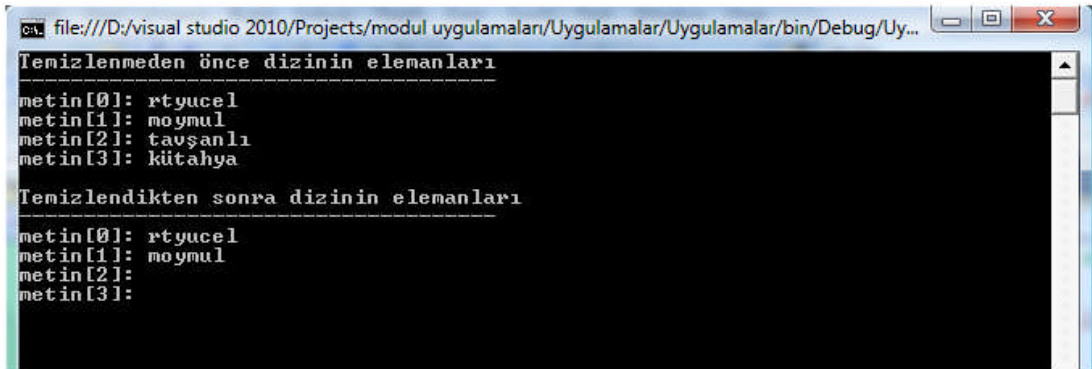
Resim 0-7. Örnek 3-7'in Ekran Çıktısı

Yukarıdaki Örnek 3.7'yi incelediğimizde `Array.Clear(sayilar,1,2);` komut satırı ile sayilar dizisinin 1 nolu indeksinden başlayarak 2 adet dizi elemanının değerini temizlemiş olduk. Yani dizi elemanlarının değerlerinin yerine 0 değeri verilmiş oldu.

Örnek 3-8: Clear metodunu bir de string bir dizi üzerinde deneyip sonuçlarını inceleyelim.

```
string[] metin = new string[] { "rtyucel", "moymul",  
"tavşanlı", "kütahya" };  
Console.WriteLine("Temizlenmeden önce dizinin elemanları");  
Console.WriteLine("-----");  
for (int i = 0; i < metin.Length; i++)  
    Console.WriteLine("metin[{0}]: {1}", i, metin[i]);  
  
Array.Clear(metin, 2, 2);  
  
Console.WriteLine("\nTemizlendikten sonra dizinin elemanları");  
Console.WriteLine("-----");  
  
for (int i = 0; i < metin.Length; i++)  
    Console.WriteLine("metin[{0}]: {1}", i, metin[i]);
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsü alırız;



Resim 0-8. Örnek 3-8'in Ekran Çıktısı

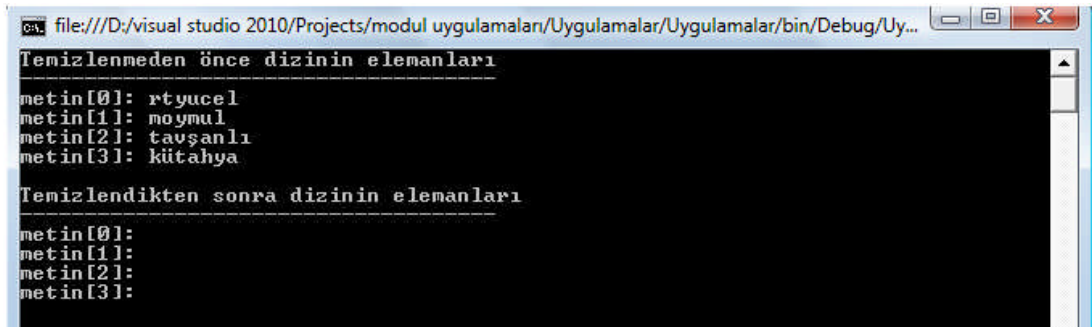
Yukarıdaki örnek 3.8'i incelediğimizde `Array.Clear(metin,2,2)` satırıyla metin isimli dizinin 2 numaralı indeksinden başlayarak 2 adet kaydın silinmesi işlemini gerçekleştirdik. Burada yeni değerlerin "" (boş metin) olduğuna dikkat ediniz.

Dizideki tüm elemanları silmek için sıradaki örneğimizi inceleyelim.

Örnek 3-9: Bir dizi içerisindeki tüm elemanları silen programı yazınız.

```
string[] metin = new string[] { "rtyucel", "moymul",  
"tavşanlı", "kütahya" };  
Console.WriteLine("Temizlenmeden önce dizinin elemanları");  
Console.WriteLine("-----");  
for (int i = 0; i < metin.Length; i++)  
    Console.WriteLine("metin[{0}]: {1}", i, metin[i]);  
  
Array.Clear(metin, 0, metin.Length);  
  
Console.WriteLine("\nTemizlendikten sonra dizinin elemanları");  
Console.WriteLine("-----");  
  
for (int i = 0; i < metin.Length; i++)  
    Console.WriteLine("metin[{0}]: {1}", i, metin[i]);
```

Yukarıdaki kodlar çalıştırılınca aşağıdaki ekran görüntüsü karşımıza çıkar;



Resim 0-9. Örnek 3-9'un Ekran Çıktısı

Yukarıdaki kodlardan `Array.Clear(metin, 0, metin.Length);` satırı incelendiğinde; Clear metodunda dizinin 0. elemanından itibaren eleman sayısı (`metin.Length`) kadar verinin silineceğini görebiliriz. Bu şekilde bir dizi içerisindeki tüm elemanları silme işlemi gerçekleştirilir.

3.4.3. Reverse(Dizi)

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

➤ Kullanımı:

```
Array.Reverse(diziAdi);
```

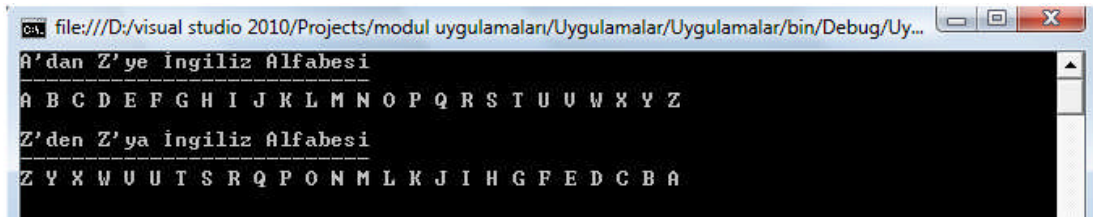
Örnek 3-10: Alfabe adlı dizi içerisine girilen A-Z'ye harfleri tersten ekrana yazdıran programın kodunu yazınız.

```

char[] alfabe = new char[26];
char harf;
int i=0;
Console.WriteLine("A'dan Z'ye İngiliz Alfabeti");
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 A'ya İngiliz Alfabeti");
Console.WriteLine("-----");
for (i = 0; i < 26; i++)
    Console.Write(alfabe[i] + " ");

```

Yukarıdaki kodlar çalıştırılınca aşağıdaki ekran görüntüsü karşımıza çıkar;



Resim 0-10 Örnek 3-10'un Ekran Çıktısı

3.4.4. Sort(Dizi)

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 3-11: 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);

```

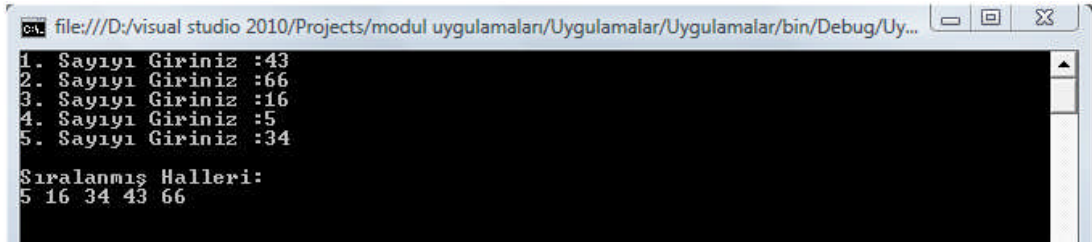
```

Console.WriteLine("");
Console.WriteLine("Sıralanmış Halleri: ");
foreach (int sayi in sayilar)
    Console.Write(sayi + " ");

```

Bu örnekte aynı zamanda foreach döngüsünün kullanımını da görmüş olduk. Bundan sonra dizilerle ilgili örneklerimizde foreach döngüsünü sıkça kullanacağız.

Yukarıdaki kod parçası çalıştırıldığı zaman aşağıdaki gibi bir ekran çıktısıyla karşılaşırız;



Resim 0-11. Örnek 3-11'in Ekran Çıktısı

3.4.5. IndexOf(Dizi, arananDeger)

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 3-12: Daha önceden değerleri girilmiş olan bir dizi içerisinde istenilen değeri bulan programın kodunu yazınız.

```

string[] iller = new string[] { "Ankara", "İstanbul",
"Kütahya", "İzmir", "Yozgat" };
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);

```

Yukarıdaki kod parçası çalıştırıldığı zaman aşağıdaki gibi bir ekran çıktısıyla karşılaşırız;

```
file:///D:/visual studio 2010/Projects/modul uygulamaları/Uygulamalar/Uygulamalar/bin/Debug/Uy...
iller[0]: Ankara
iller[1]: İstanbul
iller[2]: Kütahya
iller[3]: İzmir
iller[4]: Yozgat
illeri dizisi içerisinde Kütahya'nın indeksi: 2
```

Resim 0-12. Örnek 3-12'nin Ekran Çıktısı

Uyarı: Eğer aranan="KÜTAHYA" veya aranan="kütahya" yazarsanız Array.IndexOf(iller,aranan) ifadesi geriye -1 değerini döndürür.

Örnek 3-13: 1-49 arasında 6 adet rastgele sayı üreten bir Sayısal Loto Programı hazırlayınız.

```
// Boyutu 6 olan int array' i tanımlayın.
int[] sayilar = new int[6];
//Random tipinden bir değişken oluşturun.
Random r = new Random();
//int tipinden bir değişken oluşturun ve ilk değerini 0 olarak
atayın.
int counter = 0;
//Bir while döngüsü tanımlayın ve koşul olarak counter<6 olarak
belirtin.
while (counter < 6)
{
    //int tipinden bir değişken oluşturun ve değerini Random
    değişkenin 1 ile 49 arası ürettiği tamsayıya eşitleyin.
    int sayi = r.Next(1, 50);
    //Sayi adlı değişkenin değerinin sayilar adlı dizide var
    olup olmadığını Array.IndexOf metodu ile kontrol edin.
    if (Array.IndexOf(sayilar, sayi) == -1)
    {
        //Eğer sayi adlı değişkenin değeri sayilar dizisinde
        yoksa dizinin counter numaralı ögesine sayi değerini eşitleyin.
        sayilar[counter] = sayi;
        //counter'ı 1 arttırın.
        counter++;
    }
}
Array.Sort(sayilar);
//Sayilar dizisini ekrana yazdırın
Console.WriteLine("Bu haftanın şanslı sayıları: ");
foreach (int i in sayilar)
    Console.Write(i + " ");
```

Yukarıdaki kod parçası çalıştırıldığı zaman aşağıdaki gibi bir ekran çıktısıyla karşılaşırız;

```
file:///D:/visual studio 2010/Projects/modul uygulamaları/Uygulamalar/Uygulamalar/bin/Debug/Uy...
Bu haftanın şanslı sayıları: 5 6 8 13 20 47
```

Resim 0-13. Örnek 3-13'ün Ekran Çıktısı

Uyarı: Random() komutu sayesinde program her çalıştırıldığında farklı sayılar üreteceğinden Resim 3-13'teki sayıların sizin programınızda da çıkma olasılığı oldukça düşüktür.

3.5. Dinamik Diziler

Şimdiye kadar gördüğümüz 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-daraltılamazdı.

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.

İşte dizilerde sıkça karşılaşılan bu kısıtlamalar ArrayList sınıfı ile çözümlenir. ArrayList, büyüklüğü, dinamik olarak artıp azalabilen nesne referanslarından oluşan değişken uzunlukta bir dizedir. Bu veri yapısı .NET sınıf kütüphanesinin **System.Collections** isim alanında bulunur.

ArrayList yapısının, bu dinamik boyut dışında bizlere sunduğu bir diğer avantaj da bir dizi içerisinde saklanacak olan verilerin tür sınırlandırmasını ortadan kaldırmasıdır. Örneğin bir dizi içerisinde hem int türünden veriler, hem string türünden veriler, hem char türünden veriler hem de bool türünden veriler saklamak mümkündür.

ArrayList ile dinamik bir dizi şu şekilde tanımlanır;

➤ **Tanımlanması:**

```
ArrayList diziAdi=new ArrayList();
```

ArrayList'leri örneklerimizde kullanmadan önce sizlere ArrayList'ler ile sıkça kullandığımız bazı metotlardan ve özelliklerden bahsetmemizde fayda olacaktır.

Özellikler:

Capacity: ArrayList'in kapasitesini int türünde verir.

Count: ArrayList içerisindeki eleman sayısını int türünde verir.

Metotlar:

Add: Bir nesneyi ArrayList'in sonuna eklemeye yarar.

Insert: Belirtilen indeks pozisyonuna nesneyi eklemeye yarar.

Remove: Belirtilen nesne ArrayList içerisinde varsa siler.

RemoveAt: İndeks değeriyle belirtilen pozisyondaki elemanı siler.

Sort: ArrayList içerisindeki elemanları sıralar.

3.5.1. Capacity Özelliği:

ArrayList'in kapasitesini int türünde veren özelliktir.

➤ **Kullanımı:**

```
int kapasite=liste.Capacity;
```

3.5.2. Count Özelliği:

ArrayList içerisinde bulunana eleman sayısını int türünde veren özelliktir.

➤ **Kullanımı:**

```
int elemanSayisi=liste.Count;
```

3.5.3. Add Metodu:

Bir nesneyi ArrayList'in sonuna eklemeye yarar.

➤ **Kullanımı:**

```
ArrayList liste=new ArrayList();
liste.Add(123); //int türünde değer ekleme
liste.Add("Tevfik"); //string türünde değer ekleme
liste.Add('H'); //char türünde değer ekleme
liste.Add(true); //bool türünde değer ekleme
liste.Add(3.14d); //double türünde değer ekleme
liste.Add(3.666f); //float türünde değer ekleme
```

Örnek 3-14: 0-50 arasında 3'e kalansız bölünebilen sayıları ArrayList içerisine ekleyen programın kodunu yazınız.

```
ArrayList liste=new ArrayList();
for (int sayi = 0; sayi < 100; sayi++)
{
    if (sayi % 3 == 0)
    {
        liste.Add(sayi);
        Console.WriteLine(sayi+" listeye eklendi.");
    }
}
```

Yukarıdaki kodu çalıştırdığımızda aşağıdaki gibi bir ekran çıktısı alırız.

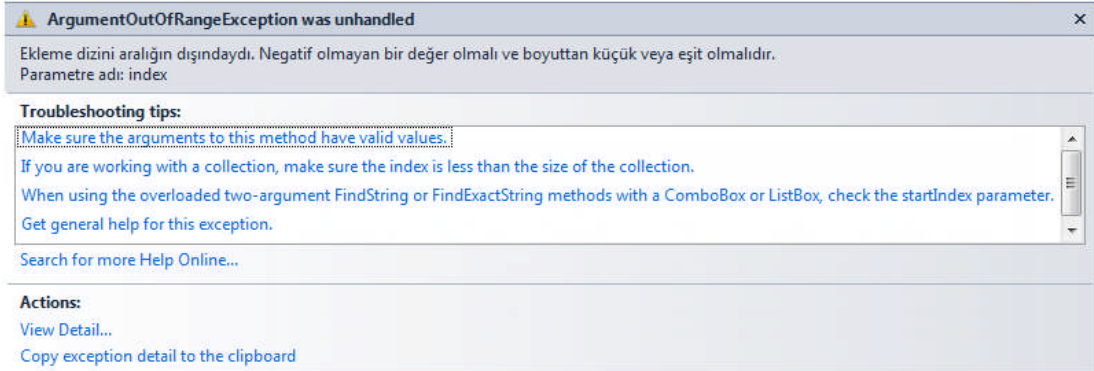

```
file:///D:/visual studio 2010/Projects/modul uygulamaları/Uygulamalar/Uygulamalar/bin/Debug/Uy...
0 listeye eklendi.
3 listeye eklendi.
6 listeye eklendi.
9 listeye eklendi.
12 listeye eklendi.
15 listeye eklendi.
18 listeye eklendi.
21 listeye eklendi.
24 listeye eklendi.
27 listeye eklendi.
30 listeye eklendi.
33 listeye eklendi.
36 listeye eklendi.
39 listeye eklendi.
42 listeye eklendi.
45 listeye eklendi.
48 listeye eklendi.
```

Resim 0-14. Örnek 3-14'ün Ekran Çıktısı

3.5.4. Insert Metodu:

Parametre olarak belirtilen indeks değerine yine parametre olarak verilen nesneyi ekler. Ekleme işleminden önce o indeksteki ve o indeksten sonraki tüm değerler birer sonraki indekslere kaydırılır.

Dikkat edilmesi gereken nokta; araya eklenmek istenilen indeks değerinden en az bir önceki konumda veri bulunması gerekir. Aksi takdirde Resim3-15'teki hata mesajını alırız.



Resim 0-15. ArrayList Indeks Sınır Dışı Hatası

➤ Kullanımı:

```
liste.Insert(5,123); //5 nolu indekse 123 değerini ekler
```

Örnek 3-15: Oluşturacağınız bir ArrayList'e 5 adet nesne ekleyiniz. Daha sonra insert metodunu kullanarak 10. indekse başka bir nesne eklemeye çalışınız. Aldığınız ekran çıktısını arkadaşlarınızla paylaşınız.

Örnek 3-16: 0'dan 9'a kadar rakamları barındıran bir ArrayList'in aşağıda verilen değerleri sırasıyla 5.indeksine ekleyen kodu yazınız.

- 123,	- Tefvik,	- H,
- true,	- 3.14d,	- 3.666f

```
ArrayList liste = new ArrayList();

for (int i = 0; i < 10; i++)
    liste.Add(i);

Console.WriteLine("Insert işleminden önce liste:");

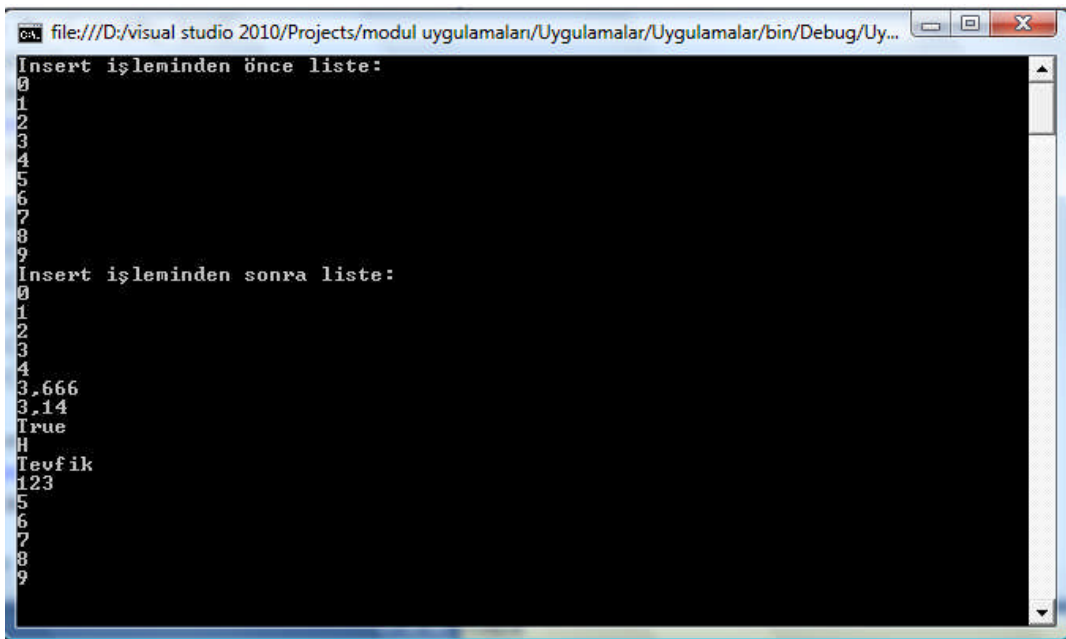
foreach (object obj in liste)
    Console.WriteLine(obj);

liste.Insert(5, 123);
liste.Insert(5, "Tevfik");
liste.Insert(5, 'H');
liste.Insert(5, true);
liste.Insert(5, 3.14d);
liste.Insert(5, 3.666f);

Console.WriteLine("Insert işleminden sonra liste:");

foreach (object obj2 in liste)
    Console.WriteLine(obj2);
```

Yukarıdaki kod parçası çalıştırıldığı zaman aşağıdaki gibi bir ekran çıktısı alırız.



```
file:///D:/visual studio 2010/Projects/modul uygulamaları/Uygulamalar/Uygulamalar/bin/Debug/Uy...
Insert işleminden önce liste:
0
1
2
3
4
5
6
7
8
9
Insert işleminden sonra liste:
0
1
2
3
4
3.666
3.14
True
H
Tevfik
123
5
6
7
8
9
```

Resim 0-16. Örnek 3-16'nın Ekran Çıktısı

3.5.5. Remove Metodu:

Belirtilen nesne ArrayList içerisinde varsa siler.

➤ **Kullanımı:**

```
liste.Remove(nesne);
```

Örnek 3-17: Aşağıda verilen değerleri sırasıyla bir ArrayList'e ekledikten sonra Tevfik,123 ve 3.14 değerlerini silen kodu yazınız.

- 123,	- Tevfik,	- H,
- true,	- 3.14d,	- 3.666f

```
ArrayList liste = new ArrayList();
liste.Add(123);
liste.Add("Tevfik");
liste.Add('H');
liste.Add(true);
liste.Add(3.14d);
liste.Add(3.666f);

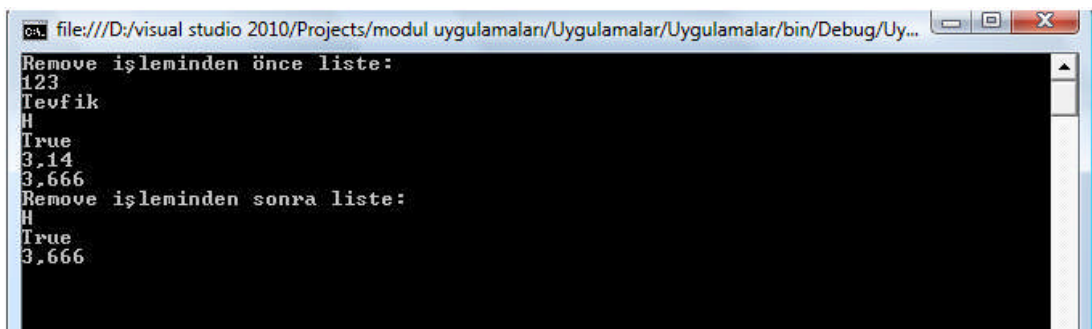
Console.WriteLine("Remove işleminden önce liste:");

foreach (object obj in liste)
    Console.WriteLine(obj);

liste.Remove("Tevfik");
liste.Remove(123);
liste.Remove(3.14d);

Console.WriteLine("Remove işleminden sonra liste:");
foreach (object obj2 in liste)
    Console.WriteLine(obj2);
```

Yukarıdaki kod parçası çalıştırıldığı zaman aşağıdaki gibi bir ekran çıktısı alırız.



```
file:///D:/visual studio 2010/Projects/modul uygulamaları/Uygulamalar/Uygulamalar/bin/Debug/Uy...
Remove işleminden önce liste:
123
Tevfik
H
True
3.14
3.666
Remove işleminden sonra liste:
H
True
3.666
```

Resim 0-17. Örnek 3.17'nin Ekran Çıktısı

Yukarıdaki kod parçasını incelersek ve Resim 3.17’den de görüldüğü gibi liste dizisinden “Tevfik”,123 ve 3.14d nesnelerinin silindiğini görebiliriz.

3.5.6. RemoveAt Metodu:

Parametre olarak verilen indeks konumundaki elemanı siler.

➤ Kullanımı:

```
liste.RemoveAt(indeks); //indeks olarak verilen konumda bulunan elemanı siler.
```

Örnek 3-18: Aşağıda verilen değerleri sırasıyla bir ArrayList’e ekledikten sonra 2. ve 4. indeksteki elemanları silen kodu yazınız.

- 123,	- Tevfik,	- H,
- true,	- 3.14d,	- 3.666f

```
ArrayList liste = new ArrayList();
liste.Add(123);
liste.Add("Tevfik");
liste.Add('H');
liste.Add(true);
liste.Add(3.14d);
liste.Add(3.666f);

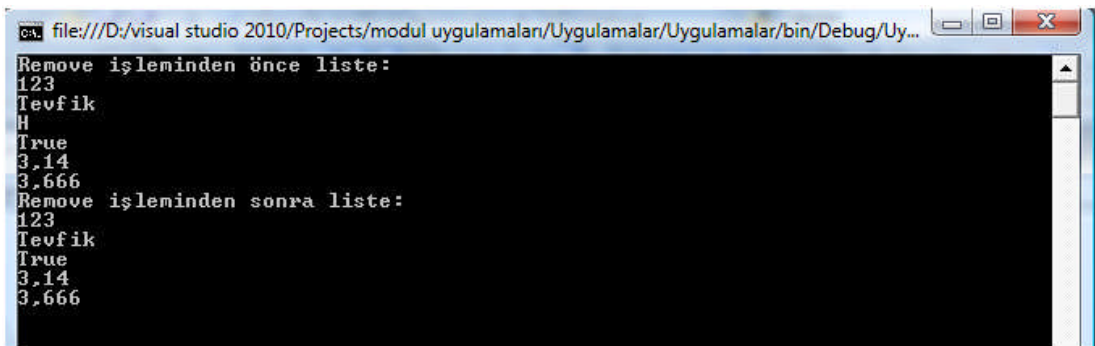
Console.WriteLine("Remove işleminden önce liste:");

foreach (object obj in liste)
    Console.WriteLine(obj);

liste.RemoveAt(2);

Console.WriteLine("Remove işleminden sonra liste:");
foreach (object obj2 in liste)
    Console.WriteLine(obj2);
```

Yukarıdaki kod parçası çalıştırıldığı zaman aşağıdaki gibi bir ekran çıktısı alırız.



```
file:///D:/visual studio 2010/Projects/modul uygulamaları/Uygulamalar/Uygulamalar/bin/Debug/Uy...
Remove işleminden önce liste:
123
Tevfik
H
True
3.14
3.666
Remove işleminden sonra liste:
123
Tevfik
True
3.14
3.666
```

Resim 0-18. Örnek 3-18'in Ekran Çıktısı

3.5.7. Sort Metodu:

ArrayList içerisindeki elemanları küçükten büyüğe sıralar.

➤ **Kullanımı:**

```
liste.Sort();
```

Örnek 3-19: Klavyeden girilen 5 adet ismi bir ArrayList içerisine kaydedin ve bunları A'dan Z'ye sıralayan kodu yazınız.

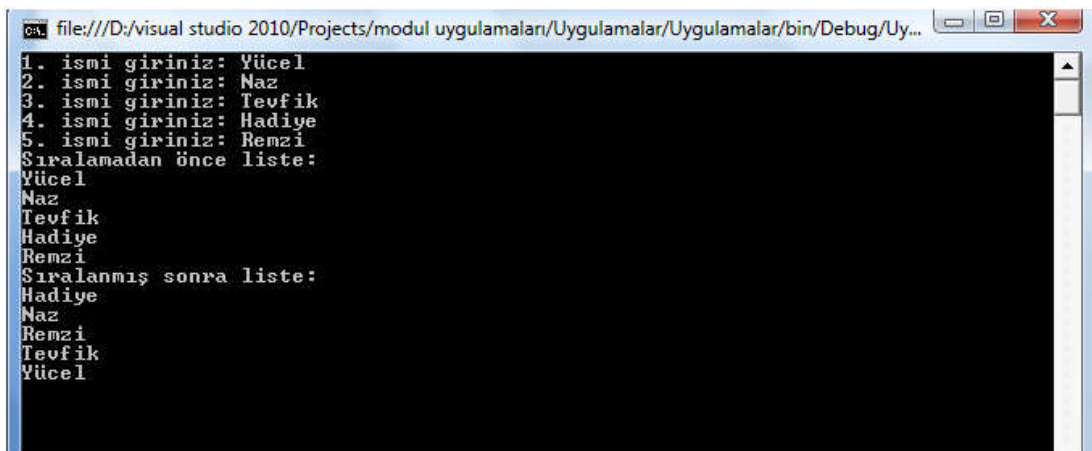
```
ArrayList liste = new ArrayList();
string isim;
for (int i = 1; i <= 5; i++)
{
    Console.Write(i+". ismi giriniz: ");
    isim = Console.ReadLine();
    liste.Add(isim);
}

Console.WriteLine("Sıralamadan önce liste:");

foreach (object obj in liste)
    Console.WriteLine(obj);

liste.Sort();

Console.WriteLine("Sıralanmış sonra liste:");
foreach (object obj2 in liste)
    Console.WriteLine(obj2);
```



Resim 0-19. Örnek 3-19'un Ekran Çıktısı

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
0'dan klavyeden girilen sayıya kadar olan sayılardan <ul style="list-style-type: none"> ➤ tek olanları tekSayılar dizisinde, ➤ çift olanları da çiftSayılar dizisinde saklayan daha sonra bu dizileri ayrı ayrı ekrana yazdıran programı yazınız. 	<ul style="list-style-type: none"> ➤ Döngünün bitiş değerini klavyeden girilen sayı olarak belirleyiniz. ➤ Döngü sayacını kontrol edip tek ise tekSayılar dizisine, çift ise çiftSayılar dizisine ekleyiniz. ➤ Herhangi bir döngü yardımıyla dizileri ayrı ayrı ekrana yazdırınız.
Bir öğrenciye ait 3 yazılı ve 2 sözlü notunu bir dizi içerisinde saklayan ve daha sonra bu notların ortalamasını hesaplayan <ul style="list-style-type: none"> ➤ Ortalama 45'e eşit ya da büyükse ekrana "Dersi Geçtin", ➤ Değilse "Dersten Kaldın" yazdıran programın kodunu yazınız.	<ul style="list-style-type: none"> ➤ Notları bir dizi içerisinde saklayınız. ➤ Dizi içerisinde bir döngü ile notları okuyup ortalamasını bulunuz. ➤ Ortalamanın 45'e eşit ya da büyük olup olmadığını kontrol ediniz. ➤ Ortalamanın durumuna göre ekrana ilgili durumu yazdırınız.
Sınıftaki arkadaşlarınızın <ul style="list-style-type: none"> ➤ isim, ➤ telefon, ➤ e-posta adresi bilgilerini saklayacağınız diziler oluşturup bu bilgileri bir döngü yardımıyla bu dizilere kaydeden, <p>Daha sonra klavyeden girilen isme ait bilgileri ekrana getiren, Eğer klavyeden girilen isim kayıtlarda bulunamazsa ekrana "Kayıt bulunamadı" mesajı veren programın kodunu yazınız</p>	<ul style="list-style-type: none"> ➤ İsimler, telefon ve e-posta adresleri için ayrı ayrı diziler oluşturunuz. ➤ Bir kişiye ait bilgilerin indeks numaralarının 3 dizide de aynı olmasına dikkat ediniz. ➤ Dizilerde arama metodu ile ilgili örneği inceleyin ve isim dizisinde klavyeden girilen isme göre arama yaptırınız. ➤ Dönen sonuca göre ilgili bilgileri/mesajı ekrana yazdırınız.
Klavyeden bir dizi boyutu girin. Daha sonra dizi boyutu kadar ondalıklı sayıyı bu dizi içerisine ekleyin. Diziye eklenen bu sayıların <ul style="list-style-type: none"> ➤ Toplamını, ➤ Aritmetik ortalamasını, ➤ En büyük değerini ➤ En küçük değerini ekrana yazdıran programı yazınız.	<ul style="list-style-type: none"> ➤ Klavyeden girilen sayıyı dizinin boyutu olarak belirleyiniz. ➤ Girilen sayıya göre bir döngü oluşturunuz. ➤ Döngü yardımıyla kullanıcının girmiş olduğu sayıları diziye ekleyiniz. ➤ Dizi içerisindeki değerlerin toplamını bulunuz. ➤ Dizi içerisindeki değerlerin aritmetik ortalamasını bulunuz. ➤ Diziyi sıralayınız. ➤ En büyük ve en küçük elemanları bulunuz.

ÖLÇME VE DEĞERLENDİRME

Bu faaliyet kapsamında kazandığınız bilgileri, aşağıdaki soruları cevaplayarak belirleyiniz.

1. Tek boyutlu dizilerle ilgili aşağıda verilenlerden hangisi **yanlıştır**?
A) Diziler, aynı tipteki değişkenleri tek bir adla saklayabildiğimiz veri yapılarıdır.
B) Diziler tanımlanırken kapasiteleri belirlenmelidir.
C) Programın akışı esnasında dizilerin kapasiteleri değiştirilebilir.
D) Programın akışı esnasında dizilerin türleri değiştirilemez.
2. Dizilerin kapasiteleri tanımlanırken kullanılan karakterler aşağıdaki hangi şıkta doğru olarak verilmiştir?
A) {}
B) []
C) ()
D) <>
3. `int[] dizi=new int[10]` şeklinde tanımlanan bir dizi için aşağıda verilenlerden hangisi **kesinlikle yanlıştır**?
A) Dizinin son elemanı 10. indekse sahiptir.
B) Dizinin ilk elemanı 0. indekse sahiptir.
C) Dizinin son elemanı 9. indekse sahiptir.
D) Dizi maksimum 10 eleman barındırabilir.
4. Aşağıda verilen dizilerle ilgili metotlardan hangisi dizi içerisindeki elemanları silmeye yarar?
A) IndexOf
B) Reverse
C) Sort
D) Clear
5. Dinamik diziler (ArrayList) ile ilgili verilenlerden hangisi **yanlıştır**?
A) Dinamik dizilerin kapasiteleri programın akışı esnasında değiştirilebilirler.
B) Count özelliği dinamik dizilerin içerisindeki eleman sayısını verir.
C) Capacity özelliği dinamik dizilerin içerisindeki eleman sayısını verir.
D) Dinamik diziler içerisinde farklı türlerde veri depolayabiliriz.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru “Modül Değerlendirme”ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. () Bir switch bloğunda aynı sabite sahip birden fazla case ifadesi yer alabilir.
2. () Float veri tipi bir case sabiti olarak kullanılabilir.
3. () default bloğu bir Switch-Case yapısında bulunmasa da olur.
4. () İç-içe birden fazla if ifadesi kullanılamaz.
5. () Eğer koşulun sağlanmaması durumunda işlem yapılması isteniyorsa, else ifadesine gerek duyulur.
6. () Eğer If veya Else ifadelerinden sonra sadece bir komut yazılacak ise küme parantezleri ({}) kullanılmayabilir.
7. () Bir case bloğu içerisinde break komutu kullanılmazsa, hata mesajı alırız.
8. default anahtar kelimesi switch-case yapısı içerisinde her zaman case ifadelerinden önce yer almalıdır.
9. () For döngüsünde sayaç char türünde de olabilir.
10. () For döngüsünde koşul sağlanmadığı müddetçe döngü çalıştırılmaz.
11. () Döngüler yalnızca ileri dönük sayma işlemi gerçekleştirir, geriye doğru sayma işlemi gerçekleştiremezler.
12. () While döngüsü koşul yanlış (false) iken küme parantezleri ({}) ile belirlenen alandaki işlemleri tekrarlar.
13. () Foreach, dizi (Array) ve koleksiyon (collection) tabanlı nesnelerin elemanları üzerinden ilerleyen bir döngüdür.
14. () Foreach döngüsünde belirtilen değişken tipinin koleksiyon ile aynı veya uyumlu bir veri tipi olması gerekir.
15. () Goto anahtar sözcüğü ile bir döngü ve koşul bloğu içerisine dallanma işlemi gerçekleştirilebilir.
16. () Dallanma işlemi bir programda yalnızca ileriye dönük yapılabilir.
17. () Tek boyutlu diziler yalnızca tanımlanırken belirlenen türde veri depolayabilirler.
18. () Diziler başlangıçta belirlenen kapasitelerinden fazla veri depolayabilirler.
19. () Bir dizinin indeks değeri negatif (-) bir tam sayı olabilir.
20. () Length özelliği dizinin kapasitesini verir.
21. () Sort metodu, dizi içerisindeki elemanları büyükten küçüğe sıralamak için kullanılır.
22. () Reverse metodu, parametre olarak verilen dizinin eleman sırasını tersine çevirir.
23. () Dinamik dizilerin kapasiteleri, programın akışı esnasında değiştirilebilir.
24. () Dinamik dizilerde yalnızca aynı türdeki verileri depolayabiliriz.
25. () Insert metoduyla bir dinamik diziyeye veri eklendiğinde, o indekste bulunan elemanın üzerine yazılır.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1	D
2	C
3	D
4	A
5	B

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1	A
2	D
3	C
4	B
5	C

ÖĞRENME FAALİYETİ-3'ÜN CEVAP ANAHTARI

1	C
2	B
3	A
4	D
5	C

MODÜL DEĞERLENDİRME SORULARI CEVAP ANAHTARI

1	Y	14	D
2	Y	15	Y
3	D	16	Y
4	Y	17	D
5	D	18	Y
6	D	19	Y
7	D	20	D
8	Y	21	Y
9	D	22	D
10	D	23	D
11	Y	24	Y
12	Y	25	Y
13	D		

KAYNAKÇA

- ALGAN Sefer, **Her Yönüyle C#**, Pusula Yayıncılık, 1.Baskı, İstanbul, Türkiye, (2003)
- SCHILDT Herbert, **Herkes İçin C#**, Alfa Yayınevi, 1.Baskı, İstanbul, Türkiye, (2002)
- ASLAN KAAAN, **A'dan Z'ye C Kılavuzu**, Pusula Yayıncılık, 8.Baskı, İstanbul, Türkiye, (2002)
- Butow, E., Ryan, T.: **“Your Visual Blueprint For Building .NET Application”**
- MSDN : **“Introduction to C# Programming for the Microsoft® .NET Platform (Prerelease)Workbook”**
- Hejlsberg, A., Wiltamuth, S.: **“C# Language Reference”**
- Turtshi, A., Werry J., Hack, G., Albahari, J., Nandu S.: **“C#.NET Web Developer's Guide”**, Syngress Publishing, Inc., Rockaland, USA (2002)
- Microsoft: **“C# Language Specification”**
- MSDN Yardım Dokümanları