PROGRAMLAMA TEMELLERİ KONU ÖZETİ

TANIMLAMALAR

Programlamada kullanılacak tüm verilerin mutlaka daha önceden bildirimlerinin yapılması gerekir. Bu bildirimde kullanılacak ve verinin türü, adı ve eğer varsa başlangıç değeri belirlenir. Farklı programlama dillerine göre ver türlerinde bazı farklılıklar olsa da temel türler ortaktır. Verilerin Sayısal, Metinsel ve Mantıksal olarak üç ana türdedir. Sayısal veriler Tam sayılar ve Ondalıklı sayılar, Metinsel veriler de Metin ve Karakter olmak üzere iki temel türdedir. Aşağıda bu türlerde ki veri tipleri verilmiştir.

SAYISAL VERİ TİPLERİ

Grup	Adı	Kapasite
Tam Sayılar	sbyte	-128 … 127
	byte	0 … 255
	short	-32768 … 32767
	ushort	0 … 65536
	İnt	-2147483648 … 2147483647
	uint	0 … 4294967295
	long	-9223372036854775808 …9223372036854775808
	ulong	0 … 18446744073709551615
Ondalık Sayılar	float	-3,402823 10^{38} … 3,402823 10^{38}
	double	-1,79769313486232 10^{308} … 1,79769313486232 10^{308}

METİNSEL VERİ TİPLERİ

Adı	Kapasite
String	Birden fazla karakterden oluşan veriler
Char	Tek karakterden oluşan veriler (16 bit uzunlukta)

MANTIKSAL VERİ TİPLERİ

Bool	**True** ya da **False** değeri alır

OPERATÖRLER

Programlama dillerinde komutlar genellikle sözel ifadeler şeklindedir. Bu ifadeler bir İngilizce kelime ya da kısaltma şeklinde olabilir. Ancak matematiksel denklem ifadelerinin oluşturulmasında ise sözel komutlar yerine simgesel komutlar kullanılır. Bu simgesel komutlara operatör adı verilir. C# dilinde kullanılan önemli operatörler aşağıda kategoriler halinde verilmiştir.

Aritmetiksel Operatörler

Operatör	İşlevi	Örnek Kullanım
=	Atama	a=b
+	Toplama	a=x+y
-	Çıkarma	a=x-y
*	Çarpma	a=x*100
/	Bölme	a=x/2
%	Mod Alma	a=x%3
++	Bir Artırma	a++, a=b++
--	Bir Eksiltme	b--, b=a--

İşlemli Özel Atama Operatörleri

Operatör	İşlevi	Kullanımı	Eşdeğeri
+=	Toplayarak atama	a+=b	a=a+b
-=	Çıkartarak atama	a-=b	a=a-b
*=	Çarparak atama	a*=b	a=a*b
/=	Bölerek atama	a/=b	a=a/b
%=	Mod alarak atama	a%=b	a=a%b

Mantıksal Operatörler

Operatör	Anlamı	Örnek Kullanım
==	Eşit mi?	A == B
<	Küçük mü?	A < B
>	Büyük mü?	A > B
<=	Küçük ya da eşit mi?	A <= B
>=	Büyük ya da eşit mi?	A >= B
!=	Farklı mı?	A != B

Operatörlerde İşlem Öncelikleri

Aritmetik denklemler düzenlenirken önemli bir noktada işlem öncelikleridir. Operatörlerin gerçekleştirilmesinin birbirlerine göre öncelikleri vardır. Denklemin doğru tanımlanabilmesi için bu önceliklere dikkat edilmesi gerekir. Denklemlerde öncelikle parantez içleri uygulanır bu nedenle doğru denklemin tanımlanması için parantezler kullanmak gerekebilir. Aşağıdaki tabloda işlem öncelikleri sırası ile verilmiştir. Aynı işlem önceliğine sahip operatörler soldan sağa doğru gerçekleştirilir.

İşlem Önceliği	Operatör
1	A++, A--
2	++A, --A
3	*, /, %
4	+, -
5	<, <=, >, >=
6	== , !=
7	&&
8	||
9	=, +=, -=, *=, /=, %=

PROGRAM DENETİM DEYİMLERİ

Programlamada işlemlerin tekrar edilmesi ya da bir koşula göre farklı işlemlerin tercih edilerek gerçekleştirilmesi gerekebilir. Bu akışın sağlanabilmesi için Program denetim deyimleri dediğimiz komut yapılarından yararlanırız. Programların işleyişini düzenleyen en önemli yapılar program denetim deyimleridir. Programın akışını belirleyen bu yapılar iki temel gruba ayrılabilir.

- Koşul (Şart) İfadeleri
- Tekrar (Döngü) İfadeleri

KOŞUL (ŞART) İFADELERİ

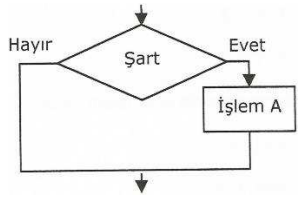
Program dallandırmak için kullanılan yapılardır. Belirli bir şarta bağlı olarak farklı işlemlerin yapılmasını sağlarlar. Dolayısı ile koşul ifadeleri programın akışını kontrol eden ifadelerdir. Üç farklı koşul ifadesi vardır;

- İf yapısı
- Switch yapısı
- ?: operatörü

1) IF/ ELSE

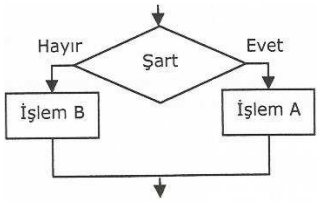
Verilen şartın doğru ya da yanlış oluşuna göre farklı işlemlerin yapılmasını sağlayan yapıdır. Bu yapı if/ else şeklinde kullanılabildiği gibi tek başına if olarak ta kullanılabilir. Tek başına if olarak kullanıldığında koşulun yanlış olduğu durum için herhangi bir işlem yapılmaz.

IF Yapısı: Koşul Doğru olduğunda işlem gerçekleştirilecek yanlış olduğunda herhangi bir işlem yapılmayacaktır.




```
if (Şart)
{
    İşlem A;
}
```

IF / ELSE Yapısı: Koşul Doğru olduğunda bir işlem yanlış olduğunda başka bir işlem gerçekleştirilecektir.



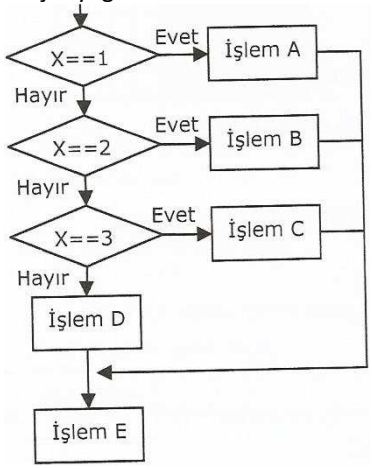

```
if (Şart)
{
    İşlem A;
}
else
{
    İşlem A;
}
```

If ve else yapılarından sonra kullanılan küme parantezleri deyimlerin etki alanlarını gösterir. Koşulun sağlandığı durumda işletilecek olan deyimler küme parantezinin içinde yer alır. Eğer şart durumları için çalıştırılacak komutlar tek satırlık ise parantez kullanılmayabilir.

2) SWİTCH

Bir seçici ile birden fazla seçeneğin içerisinden bir tanesinin seçilerek işletilmesini sağlayan yapıdır. Seçici değişken, seçimi seçenekler ile karşılaştırılarak hangi seçeneğe eşit olduğunu bulur ve o seçenek ile verilen işlemler yerine getirilir.

Akış Diyagramı



Kullanımı

```
switch(seçici)
{
  case 1: işlem A;
          break;
  case 2: işlem B;
          break;
  case 3: işlem C;
          break;
  …
  case N: işlem N;
          break;
  default: işlem X;
}
```

Switch deyiminde yukarıda gösterildiği gibi seçici değişken program içerisinden gelen bir değerdir ve case ile belirtilen seçeneklerde aranır. Bulunduğunda o işlem gerçekleştirilir. Eğer seçici değişken hiçbir seçenekte yok ise default ile bildirilen işlem gerçekleştirilir. Seçenekte verilen işlemler bir satırdan fazla ise küme parantezleri ({}) arasında yazılırlar.

3.) ?: **OPERATÖRÜ**

If/else yapısı yerine kullanılan bir operatördür.

Kullanımı: **(koşul)?** İşlem A: işlem B;

Eğer koşul doğrusu ise işlem A, yanlış ise işlem B gerçekleştirilir.

TEKRAR (DÖNGÜ) İFADELERİ

Bir kod ya da kod bloğunun program içerisinde 1'den fazla çalıştırmak gerektiğinde tekrarlı yapılar kullanılır. Bu bir işlemin tekrarı ya da bir matematiksel denklemin işletilmesi amacı ile yapılabilir. Döngüler birbiri ile ilişkili program kodlarını ardı ardına tekrarlayarak çalıştırmak için kullanılır. Örneğin 0 ile 100 aralığındaki sayıları toplamak için her seferinde bir sonraki sayıyı toplamın üzerine eklememiz gerekir. Bu durumda bu sayıları sayacak bir döngü kurmamız gerekir. Temel olarak iki grupta incelenebilir;

- Sabit Döngüler; **for**
- Şartlı (Koşullu) Döngüler; **While** / **Do – while**

1) SABİT DÖNGÜ: FOR

Belirli bir kod öbeğinin belli sayıda tekrar edilmesini sağlayan tekrar yapısıdır. Döngüye girilmeden önce döngünün tekrar sayısı bellidir. Dört önemli unsuru bulunur.

I. Döngü değişkeni: Belli aralıkta değer artışını sağlayacak değişkendir. Sayısal tipte bir veri olması gerekir.

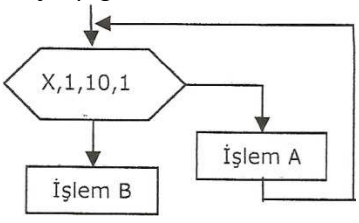
II. Başlangıç değeri: Döngünün başlayacağı değerdir.

III. Döngü başlangıç ve devam koşulu: Döngünün başlayabilmesi ve devam edebilmesi için sağlanması gereken koşuldur. Döngü değişkenin aldığı değer bu koşulu sağladığı sürece döngü tekrar etmeye devam eder. Koşulun sağlanmadığı ilk değer döngünün de bittiği yerdir.

IV. Değişim miktarı: Döngü değişkeninin değişim miktarını belirler. Bu değişim 1'er 1'er olabileceği gibi farklı değerlerle de değişim sağlanabilir. Değişim artma ya da azalma şeklinde olabilir.

Aşağıda verilen durumlara ve değerlere göre;
Döngü değişkeni :X
Başlangıç değeri :1
Bitiş Değeri :10(X<=10)
Değişim miktarı :+1 (X++)

Akış Diyagramı



Kullanımı

```
for(X=1;X<=10;X++)
{
    //for çalışma bloğu
    İşlem A;
    …;
}
```

Eğer **for** bloğu içerisindeki işlemler tek satırlık ise küme parantezleri kullanılmayabilir.

2) KOŞULLU DÖNGÜLER

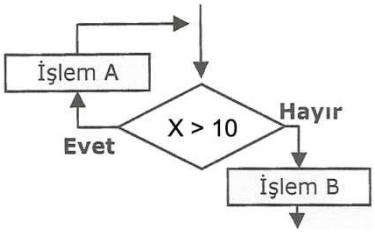
Bu tip döngülerde belirli bir tekrar sayısı ya da aralığı söz konusu değildir. Yani döngüye girilmeden önce tekrar sayısı belli olmayabilir. Belirtilen koşul sağlandığı müddetçe döngü bloğundaki komutlar tekrarlanmaya devam edilir. Koşul sağlanmadığı anda döngü sonlanır. Koşulun başta ve sonda sınanmasına göre iki farklı şekilde kullanılır.

● **While döngüsü (Giriş kontrollü Döngü):** Koşul başta kontrol edilir, sonra döngü komutları çalıştırılmaya başlanır. Önce verilen koşula bakılır. Koşul sağlanıyorsa komutlar çalıştırılır ve tekrar koşula bakılmak üzere geri dönülür. Koşul doğru sonucunu verdiği sürece tekrar yapısı çalışmaya devam eder. Koşul sağlanmadığında döngü sonlanır. İki unsuru bulunur;

Koşul Değişkeni: koşul içerisinde yer alan değişkendir. Bu değişkenin alacağı değere göre koşul sağlanır ya da sağlanmaz.

Koşul: Döngünün devamı için gerekli şarttır. Değişkenin aldığı değer bu koşul ile sınanır.

Akış Diyagramı

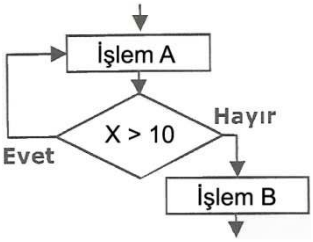


Kullanımı

```
while (koşul)
{
   İşlem A;
}
```

● **Do – while (Çıkış Kontrollü Döngü):** Döngü komutları bir kez işletildikten sonra koşul sınanır. Eğer koşul sağlanıyorsa döngü deyimleri yeniden çalıştırılır ve tekrar koşula bakılır. Koşul sağlandığı sürece döngü tekrar etmeye devam eder. Koşul sağlanmadığı noktada döngüden çıkılır. While deyim için gerekli unsurlar do-while içinde geçerlidir.

Akış Diyagramı



Kullanımı

```
do
{
   İşlem A;
} while ( koşul )
```

DÖNGÜ DENETİMİ

Break: Döngünün sonlandırılması için kullanılır. Herhangi bir noktada döngü kesilmek isteniyorsa break deyimi kullanılır.

Continue: Özellikle for döngüsünde başlayan adımı sonlandırmak için kullanılır. Başlatılan bir tekrar sonlandırılması, döngünün başına dönülerek döngü değişkenin diğer sıradaki değeri alıp yeni bir tekrara başlaması sağlanır.

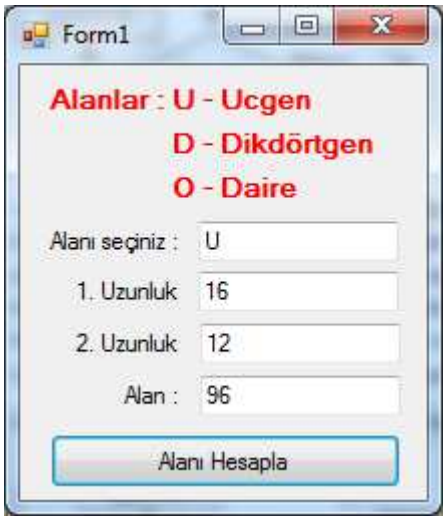
ÖRNEK UYGULAMALAR

Örnek 1. (if/else yapısı) : Kullanıcı tarafından girilen boy ve kilo bilgilerine göre Vücut Kitle Indexini hesaplayıp, İndex değeri ile birlikte 25 sınır değerinin üstü için Şişmansınız, altı için Normalsiniz Mesajı veren programı yazınız.


```
private void button1_Click()
{
   double kilo, boy, vki;

   kilo =Convert.ToDouble( textBox1.Text);
   boy =Convert.ToDouble(textBox2.Text);
   vki = kilo / (boy * boy);
   label5.Text =Convert.ToString(vki);
   if (vki >= 25)
     label6.Text = "Şişmansınz";
   else
     label6.Text = "Normalsiniz";
}
```

Örnek 2. (switch yapısı): Kullanıcının yapmış olduğu seçime göre istediği şeklin alanını hesaplayan programı yazınız.



```
private void button1_Click()
{
   char alan_sec;
   double u1, u2, alan;
   const double  pi=3.14;
   alan_sec = Convert.ToChar(textBox1.Text);

   switch (alan_sec)
   {
     case 'U':
     {
       u1 = Convert.ToDouble(textBox2.Text);
       u2 = Convert.ToDouble(textBox3.Text);
       alan = (u1 * u2) / 2;
       textBox4.Text = Convert.ToString(alan);
     } break;
     case 'D':
     {
       u1 = Convert.ToDouble(textBox2.Text);
       u2 = Convert.ToDouble(textBox3.Text);
       alan = u1 * u2;
       textBox4.Text = Convert.ToString(alan);
     } break;
     case 'O':
     {
       u1 = Convert.ToDouble(textBox2.Text);
       alan = pi * (u1 * u1);
       textBox4.Text = Convert.ToString(alan);
     } break;
     default: textBox4.Text = "Hatalı Giriş!";
     break;
   }
}
```

Örnek 3. (for yapısı): Kullanıcının belirlediği aralıktaki sayıları ve bu sayıların toplamını veren programı yazınız.


```
private void button1_Click()
{
   int say, bas, bit, toplam=0;
   bas = Convert.ToInt32(textBox1.Text);
   bit = Convert.ToInt32(textBox2.Text);
   label1.Text = "Sayılar";

   for (say = bas; say <= bit; say++)
   {
     label1.Text += " \n " + say;
     toplam += say;
   }
   label4.Text = "Sayıların Toplamı =" +
                          toplam.ToString();
}
```

Örnek 4. (while yapısı): Belli miktar parası olan bir banka müşterisi, bu paranın belirlenecek bir yıllık faiz oranı ile istediği miktara ne kadar sürede ulaşacağını ve bu süre sonunda parasının net olarak kaç lira olacağını hesaplayan programı yazınız.


```
private void button1_Click()
{
   double ap, sp, faiz, yil=0;
   ap = Convert.ToDouble(textBox1.Text);
   sp = Convert.ToDouble(textBox2.Text);
   faiz = Convert.ToDouble(textBox3.Text);

   while (ap < sp)
   {
     ap += (ap * faiz / 100);
     yil++;
   }
   label4.Text = "Süre = "+yil.ToString()+"
yıl \nSon Para ="+ap.ToString()+ "TL";
}
```

Örnek 5. (do/while yapısı): Kullanıcı 0 girinceye kadar girmiş olduğu sayıların toplamını ve adedini hesaplayıp ekrana yazan programı yazınız.


```
private void button1_Click()
{
   int toplam = 0, sayi, adet=0;
   do
   {
     sayi =
Convert.ToInt32(Microsoft.VisualBasic.Interact
ion.InputBox("Giriş", "Toplanacak Sayı", "",
200, 200));
     toplam += sayi;
     adet++;
   } while (sayi != 0);
     label1.Text = adet.ToString() + " tane
sayı girdiniz";
     label2.Text= "Sayıların Toplamı
:"+toplam.ToString();
}
```