C Sharp Programlama Dili/Akış kontrol mekanizmaları

Ders 7. Akış kontrol mekanizmaları

Program yazarken çoğu zaman herşey ak ve kara gibi net olmaz, çoğu zaman çeşitli koşullara göre farklı komutlar çalıştırmamız gerekir. Benzer şekilde çoğu komutun da yalnızca bir kez çalıştırılması bizim için yeterli gelmez, belli koşulları sağladığı sürece sürekli çalıştırılmasını istediğimiz komutlar olabilir. İşte bu gibi durumlar için C#'ta akış kontrol mekanizmaları vardır. Aslında en basitinden en karmaşığına kadar bütün programlama dillerinde bu mekanizmalar mevcuttur ve programlama dillerinin en önemli ögelerinden birisidir.

if else

`if else` deyimi sayesinde belli bir koşul sağlandığında söz konusu komutlar çalıştırılır, o belli koşullar sağlanmadığında çalıştırılmaz ya da başka komutlar çalıştırılır. Kullanılışı şu şekildedir:

```
if(koşul)
        komut1;
else
        komut2;
```

veya

```
if(koşul)
{
        komutlar1
}
else
{
        komutlar2
}
```

Yukarıdaki örneklerde eğer koşul sağlanırsa 1. komutlar, sağlanmazsa 2. komutlar çalıştırılır. if veya else'in altında birden fazla komut varsa bu komutları parantez içine almak gerekir. if veya else'in altında tek komut varsa bu komutları parantez içine almak gerekmez. Örnek bir program:

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		int a=5, b=7;
		if(a<b)
			Console.Write("a b'den küçük");
		else
			Console.Write("a b'den küçük değil");
	}
}
```

Başka bir örnek program:

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		int a=5, b=7;
		if(a<b)
		{
			Console.WriteLine("a b'den küçük");
			Console.Write(a);
		}
		else
		{
			Console.WriteLine("a b'den küçük değil");
			Console.Write(b);
		}
	}
}
```

if else yapılarında else kısmının bulunması zorunlu değildir. Bu durumda sadece koşul sağlandığında bir şeyler yapılacak, koşul sağlanmadığında bir şeyler yapılmayacaktır. Örnek:

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		int a=5, b=7;
		if(a<b)
			Console.WriteLine("a b'den küçük");
	}
}
```

if else bloklarının aşağıdaki gibi kullanımı da mümkündür:

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		Console.Write("Cinsiyetinizi girin (e, k): ");
		char cins=Convert.ToChar(Console.ReadLine());
		if(cins=='e')
			Console.Write("Erkeksiniz");
		else if(cins=='k')
			Console.Write("Kızsınız");
		else
			Console.Write("Lütfen cinsiyetinizi doğru giriniz!");
	}
}
```

Bu program kullanıcıdan cinsiyetini girmesi istemekte, eğer kullanıcının girdiği harf `e` ise ekrana `Erkeksiniz` yazmakta, eğer girdiği harf `e` değilse ise bu sefer kullanıcının girdiği harfi `k` mı değil mi diye incelemekte, eğer k girmişse ekrana `Kızsınız` yazmakta, bunların dışında bir harf girdiğinde de ekrana `Lütfen cinsiyetinizi doğru giriniz!` yazmaktadır. Bu şekilde bu bloklar daha da uzatılabilir. Yani `else if` satırının bir tane olması zorunlu değildir. Ancak tabii ki `else` satırının yalnızca bir tane olması gerekir. C# iç içe if else kullanılmasına izin verir:

```
if(koşul1)
{
	if(koşul2)
		komut1;
	else
	{
		komut2;
		komut3;
	}
}
else
	komut4;
```

switch

switch deyimi bazı if else deyimlerinin yaptığı işi daha az kodla yapar. Genellikle bazı karmaşık if else bloklarını kurmaktansa switch'i kullanmak programın anlaşılırlığını artırır. Ancak tabii ki basit if else bloklarında bu komutun kullanılması gereksizdir. Kuruluşu:

```
switch(ifade)
{
	case sabit1:
		komut1;
		break;
	case sabit2:
		komut2;
		break;
	default:
		komut3;
		break;
}
```

Bu switch deyimiyle ilgili bilmeniz gerekenler:

- İfadenin ürettiği değer hangi case sabitinde varsa o "case"deki komutlar işletilir. Eğer ifadenin ürettiği değer hiçbir case sabitinde yoksa default casedeki komutlar işletilir.
- Aynı birden fazla case sabiti olamaz. Örneğin:

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		int a=4;
		switch(a)
		{
			case 4:
				Console.Write("deneme1");
				break;
			case 4:
				Console.Write("deneme2");
				break;
			case 5:
				Console.Write("deneme3");
				break;
			default:
				Console.Write("deneme4");
				break;
		}
	}
}
```

Bu program hatalıdır.

> C#'ta herhangi bir case'e ait komutların `break;` satırı ile sonlandırılması gerekmektedir. Eğer `break;` satırı ile sonlandırılmazsa programımız hata verir. Örneğin aşağıdaki program hata vermez:

```
using System;
class AkisKontrolMekanizmalari
{
static void Main()
{
	int a=4;
	switch(a)
	{
		case 4:
			string b=Console.ReadLine();
			Console.Write(b);
			break;
		default:
			Console.Write("deneme4");
			break;
	}
}
```

Ancak şu program hata verir:

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		int a=4;
		switch(a)
		{
			case 4:
				string b=Console.ReadLine();
				Console.Write(b);
			default:
				Console.Write("deneme4");
				break;
		}
	}
}
```

Eğer programımızın bir case'deyken farklı bir case'e gitmesini istiyorsak `goto` anahtar sözcüğünü kullanırız. Örnek:

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		int a=5;
		switch(a)
		{
			case 4:
				string b=Console.ReadLine();
				Console.Write(b);
				break;
			case 5:
				Console.Write("Şimdi case 4'e gideceksiniz.");
				goto case 4;
			default:
				Console.Write("deneme4");
				break;
		}
	}
}
```

- goto satırı kullanılmışsa break; satırının kullanılmasına gerek yoktur.
- Eğer farklı case sabitlerinin aynı komutları çalıştırmasını istiyorsak şöyle bir program yazılabilir:

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		int a=5;
		switch(a)
		{
			case 4:
			case 5:
				string b=Console.ReadLine();
				Console.Write(b);
				break;
			default:
				Console.Write("deneme4");
				break;
		}
	}
}
```

Bu programda a değişkeni 4 olsa da 5 olsa da aynı komutlar çalıştırılacaktır.

- case anahtar sözcüğünün yanındaki ifade mutlaka ya sabit ya da sabitlerden oluşan bir ifade olmalıdır.
- default durumunu istediğiniz yere yazabilirsiniz, aynı şekilde istediğiniz case'leri de istediğiniz yere yazabilirsiniz. Yani case'lerin sırası önemli değildir.
- Bir switch blokunda default durumu bulunmak zorunda değildir.

- switch'in parantez içindeki ifadesi bir değişken olabileceği gibi, bir sabit ya da ifade de olabilir.

for döngüsü

Eğer programda -belli koşulları sağladığı sürece- birden fazla çalıştırılmasını istediğimiz kodlar varsa döngüler kullanılır. C#'ta en çok kullanılan döngü "for"dur. Kullanımı:

```
for(ifade1;kosul;ifade2)
        komut;
```

veya

```
for(ifade1;kosul;ifade2)
{
        komut1;
        komut2;
        .
        .
        .
}
```

for döngüsünün çalışma prensibi

```
for(ifade1;kosul;ifade2)
{
        komut1;
        komut2;
        .
        .
        .
}
```

1) ifade1 çalıştırılır.
2) Koşula bakılır. Eğer koşul sağlanıyorsa;

2.1) küme parantezleri içindeki ya da -küme parantezleri yoksa- kendinden sonra gelen ilk satırdaki komut çalıştırılır.

2.2) ifade2 çalıştırılır.

2.3) 2. adıma dönülür.

3) Eğer koşul sağlanmıyorsa küme parantezleri dışına ya da -küme parantezleri yoksa- kendinden sonra gelen ilk satırdaki komuttan hemen sonraki satıra çıkılır. Dolayısıyla döngüden çıkılmış olur.

for döngüsüyle ilgili örnekler

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		int toplam=0;
		for(int i=1;i<=3;i++)
			toplam+=i;
		Console.Write("toplam={0}",toplam);
	}
}
```

Bu program 1'den 3'e kadar olan tam sayıları (1 ve 3 dâhil) toplayıp toplamı ekrana yazacaktır.

```
using System;
class AkisKontrolMekanizmalari
{
	static void Main()
	{
		float s;
		int si,f;
		bool a=true;
		for(;a;)
		{
			Console.Write("Lütfen faktoriyelinin alınmasını
istediğiniz sayıyı girin: ");
			s=Convert.ToSingle(Console.ReadLine());
			si=(int)s;
			if(si!=s||s<1)
			{
				Console.WriteLine("Lütfen pozitif tam sayı
girin.");
				a=true;
			}
			else
			{
				a=false;
				for(f=1;si>1;si--)
				     f*=si;
				Console.Write("Faktöriyeli={0}",f);
			}
		}
	}
}
```

Bu program girdiğimiz herhangi bir sayının faktöriyelini bulup ekrana yazar. Eğer girdiğimizi sayı tam sayı değilse veya 1'den küçükse "Lütfen pozitif tam sayı girin." diyerek tekrar veri girişi ister.
UYARI: C# nokta (.) işaretini sayıların ondalık kısımlarını ayırmak için kullanırken, DOS aynı amaç için virgül (,) işaretini kullanır.

for döngüsüyle ilgili kurallar

- for döngüsünün parantezleri içindeki "for(int i=0;i<5;i++)" iki ifade ve bir koşulun istenirse bir tanesi, istenirse bazıları, istenirse de tamamı boş bırakılabilir; ancak noktalı virgüller mutlaka yazılmalıdır.
- Tahmin edebileceğiniz gibi for döngüsünün içinde veya "for(int i=0;i<5;i++)" kısmında tanımlanan herhangi bir değişken döngünün dışında kullanılamaz. Bir değişkeni döngünün dışında kullanabilmemiz için o değişkenin döngüden önce tanımlanıp ilk değer verilmesi ve değişkeni kullanacağımız yerde de faaliyet alanının devam etmesi gerekmektedir. Bu bütün döngüler ve koşul yapıları için geçerlidir. Örneğin:

```
using System;
class AkisKontrolMekanizmalari
{
       static void Main()
       {
             int a=0;
             for(int i=0;i<1;i++)
                 a=5;
             Console.Write(a);
       }
}
```

Bu program ekrana `5` yazacaktır. Ancak;

```
using System;
class AkisKontrolMekanizmalari
{
       static void Main()
       {
            int a;
            for(int i=0;i<1;i++)
                a=5;
            Console.Write(a);
       }
}
```

Bu program çalışmaz, çünkü a değişkeni döngüden önce tanımlanmasına rağmen ilk değer verilmiyor.
HATIRLATMA: Bir değişkenin faaliyet alanı tanımlandığı en iç blokun içidir.
NOT: `Console.Write("{0,3}",i);` gibi bir ifadede i değişkeni 3 birimlik bir genişlikte sağa yaslı olarak yazılır. `WriteLine` ile de kullanılabilir. i; değişken, sabit ya da ifade olabilirken 3 yalnızca sabit olmalıdır. i bir ifade olsa bile parantez içine almaya gerek yoktur.

while döngüsü

Komut ya da komutların bir koşul sağlandığı sürece çalıştırılmasını sağlar. Kuruluşu:

```
 while(koşul)
      komut;
veya
 while(koşul)
 {
      komut1;
      komut2;
      .
      .
      .
 }
```

while döngüsünün çalışma prensibi

```
 while(koşul)
 {
      komut1;
      komut2;
      .
      .
      .
 }
```

1) Koşula bakılır. Eğer koşul sağlanıyorsa;

1.1) küme parantezleri içindeki ya da -küme parantezleri yoksa- kendinden sonra gelen ilk satırdaki komut çalıştırılır.

1.2) 1. adıma dönülür.

2) Eğer koşul sağlanmıyorsa küme parantezleri dışına ya da -küme parantezleri yoksa- kendinden sonra gelen ilk satırdaki komuttan hemen sonraki satıra çıkılır. Dolayısıyla döngüden çıkılmış olur.

> Aslında while döngüsü for döngüsünün yalnızca koşuldan oluşan hâlidir. Yani `for(;i<0;)` ile `while(i<0)` aynı döngüyü başlatır.

do while döngüsü

Şimdiye kadar gördüğümüz döngülerde önce koşula bakılıyor, eğer koşul sağlanırsa döngü içindeki komutlar çalıştırılıyordu. Ancak bazen döngüdeki komutların koşul sağlanmasa da en az bir kez çalıştırılmasını isteyebiliriz. Bu gibi durumlar için C#'ta `do while` döngüsü vardır.

Kullanımı

```
 do
      komut;
 while(koşul);
veya
 do
 {
      komut1;
      komut2;
      .
      .
      .
 }while(koşul);
```

do while döngüsünün çalışma prensibi

```
 do
 {
      komut1;
      komut2;
      .
      .
      .
 }while(koşul);
```

1) Döngüdeki komutlar bir kez çalıştırılır.
2) Koşula bakılır.

2.1) Eğer koşul sağlanıyorsa 1. adıma dönülür.

2.2) Eğer koşul sağlanmıyorsa döngüden çıkılır.

Döngülerde kullanılan anahtar sözcükler

break

Hatırlarsanız break komutunu switch'teki case'lerden çıkmak için kullanmıştık. Benzer şekilde break komutu bütün döngülerden çıkmak için kullanılabilir. Örnek:

```
 using System;
 class AkisKontrolMekanizmalari
 {
      static void Main()
      {
           for(char a;;)
           {
                a=Convert.ToChar(Console.ReadLine());
                if(a=='q')
                     break;
           }
      }
 }
```

Bu program, kullanıcı "q" harfini girene kadar kapanmamaktadır.

continue

break sözcüğüne benzer. Ancak break sözcüğünden farklı olarak program continue'u gördüğünde döngüden çıkmaz, sadece döngünün o anki iterasyonu sonlanır. Örnek:

```
using System;
class AkisKontrolMekanizmalari
{
     static void Main()
     {
          for(int a=0;a<51;a++)
          {
               if(a%2==1)
                    continue;
               Console.WriteLine(a);
          }
     }
}
```

Bu program 0'dan 50'ye kadar (0 ve 50 dâhil) olan çift sayıları ekrana alt alta yazmaktadır.

goto

Nesneye yönelik programlamada pek hoş görülmese de kullanabileceğiniz başka bir komut "`goto`"dur. Aslında eskiden BASIC gibi dillerde her satırın bir numarası vardı ve bu sözcük satırlar arasında dolaşmayı sağlıyordu. Ancak böyle bir yöntem nesne yönelimli programlamaya terstir. O yüzden çok fazla kullanmamanız tavsiye edilir. Örnek kullanım:

```
using System;
class AkisKontrolMekanizmalari
{
     static void Main()
     {
               birinci:
                    Console.WriteLine("Birinci bölüm");
                    goto ucuncu;
               ikinci:
                    Console.WriteLine("İkinci bölüm");
               ucuncu:
                    Console.WriteLine("Üçüncü bölüm");
     }
}
```

Bu programda ikinci bölüm hiçbir zaman çalıştırılmayacaktır.

Döngülerle ilgili karışık örnekler

> 1'den 1000'e (sınırlar dâhil) kadar olan sayılar içerisinde 5'e tam bölünen, ancak 7'ye tam bölünemeyen sayıları alt alta listeleyen, bu sayıların kaç tane olduğunu ve toplamını yazan bir program yazınız.

```
using System;
class AkisKontrolMekanizmalari
{
     static void Main()
     {
          int toplam=0, sayi=0, i=5;
          for(;i<1001;i+=5)
          {
               if(i%35==0)
                    continue;
               sayi++;
               toplam+=i;
               Console.WriteLine(i);
          }
          Console.WriteLine("Sayısı: "+sayi);
          Console.WriteLine("Toplam: "+toplam);
     }
}
```

Girilen pozitif herhangi bir tam sayıyı ikilik düzene çeviren programı yazınız.

```
using System;
class AkisKontrolMekanizmalari
{
     static void Main()
     {
          string mod="";
          Console.Write("Lütfen ikilik sisteme dönüştürülmesini istediğiniz
sayınızı girin: ");
          float a=Convert.ToSingle(Console.ReadLine());
          int b=(int)a;
          if(a<=0||a!=b)
               mod="Bir pozitif tam sayı girmediğiniz için sayınız ikilik sisteme
dönüştürülmedi!";
          else
               for(;b>0;b/=2)
                    mod=b%2+mod;
         Console.Write(mod);
     }
}
```

Konsol ekranına girilen 0 ile 100 (sınırlar dâhil) arasındaki 10 notun en büyüğünü, en küçüğünü ve ortalamasını yazan programı yazınız.

```
using System;
class AkisKontrolMekanizmalari
{
     static void Main()
     {
          int bs=0, toplam=0, ks=0;
          for(int a=0, b;a<10;a++)
          {
               Console.Write(a+1+". notu giriniz: ");
               b=Convert.ToInt32(Console.ReadLine());
               if(b>100||b<0)
               {
                    Console.Write("Yanlış not girdiniz. Lütfen tekrar ");
                    a--;
                    continue;
               }
               if(a==0)
               {
                    bs=b;
                    ks=b;
               }
               else
               {
                    if(b>bs)
                         bs=b;
                    if(b<ks)
                         ks=b;
               }
               toplam+=b;
          }
          Console.Write("En büyük: {0}\nEn küçük: {1}\nOrtalama:
"+toplam/10,bs,ks);
     }
}
```