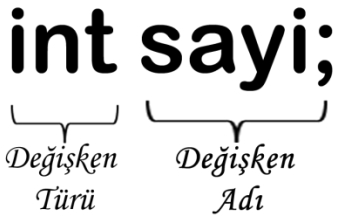
C#(Sharp) Programlama Dili

Değişkenler

Program yazarken her zaman sabit verilerle çalışmayız, çoğu zaman programımızda bir verinin kullanıcının davranışına göre değişmesi gerekir. Kullanıcıdan bir metin alıp bunu ekrana yazdıran bir program buna örnek verilebilir. Değişken kısaca bellek gözeneklerinin programlamadaki karşılıklarıdır.

C#'ta değişken tanımlama

Çoğu programlama dilinde değişkenler kullanılmaya başlanmadan önce tanımlanırlar. Aşağıdaki değişken tanımlama şekli gösterilmiştir.



Yukarıdaki şekilde C#' ta değişken tanımlamanın nasıl yapıldığı anlatılmıştır. Değişken türü bellekte ayrılan gözeneğin büyüklüğünü belirtmemizi sağlar. Değişken adı da bu gözeneğe verdiğimiz adı belirtir. Doğal olarak bu gözenekteki veriye erişmek istediğimizde veya bu gözenekteki veriyi değiştirmek istediğimizde bu adı kullanacağız. Yukarıdaki şekilde -2,147,483,648 ile 2,147,483,647 arasında (sınırlar dâhil) bir değer tutabilen ve adı "sayi" olan bir bellek gözeneği oluşturduk.

Değişkenlere değer atama

Çoğu programlama dilinde değişkenler tanımlandıktan sonra direkt olarak programda kullanılabilirler. Ancak C#' ta değişkeni tanımladıktan sonra ayrıca bir de ilk değer atamak zorundayız. Aksi bir durumda değişkeni programımız içinde kullanamayız. Değişkenlere değer atama şöyle yapılır:

```
a=5;
```

Burada a değişkenine 5 değerini atadık. Bu en basit değer atama yöntemidir. Ayrıca şunlar da mümkündür:

```
int a=5;
int b, c, d, e;
int f=10, g, m=70;
```

Birinci satırda tanımlama ve değer vermeyi aynı satırda yaptık. İkincisinde aynı türden birden fazla değişken tanımladık. Üçüncü satırda ise tanımladığımız değişkenlerin bazılarına değer verirken bazılarına vermedik.

Değişken türleri

Yukarıda değişken tanımlarken değişken türü için **int** kullanmıştık. C#' ta bunun gibi farklı kapasitelere sahip bir hayli daha değişken türü vardır. Ayrıca bazı değişken türleri sayısal, bazıları da metinseldir. Sayısal türler aşağıdaki tabloda listelenmiştir:

Tür	Boyut	Kapasite	Örnek
byte	1 bayt	0, ..., 255 (tam sayı)	`byte a=5;`
sbyte	1 bayt	-128, ..., 127 (tam sayı)	`sbyte a=5;`
short	2 bayt	-32768, ..., 32767 (tam sayı)	`short a=5;`
ushort	2 bayt	0, ..., 65535 (tam sayı)	`ushort a=5;`
int	4 bayt	-2147483648, ..., 2147483647 (tam sayı)	`int a=5;`
uint	4 bayt	0, ..., 4294967295 (tam sayı)	`uint a=5;`
long	8 bayt	-9223372036854775808, ..., 9223372036854775807 (tam sayı)	`long a=5;`
ulong	8 bayt	0, ..., 18446744073709551615 (tam sayı)	`ulong a=5;`
float	4 bayt	$\pm1.5*10^{-45}$, ..., $\pm3.4*10^{38}$ (reel sayı)	`float a=5F;` veya `float a=5f;`
double	8 bayt	$\pm5.0*10^{-324}$, ..., $\pm1.7*10^{308}$ (reel sayı)	`double a=5;` veya `double a=5d;` veya `double a=5D;`
decimal	16 bayt	$\pm1.5*10^{-28}$, ..., $\pm7.9*10^{28}$ (reel sayı)	`decimal a=5M;` veya `decimal a=5m;`

Dikkat ettiyseniz bazı değişken türlerinde değer atarken değerin sonuna bir karakter eklenmiş, bu değişken türlerindeki değişkenlere değer atarken siz de bunlara dikkat etmelisiniz. Sıra geldi metinsel türlere:

Tür	Boyut	Açıklama	Örnek
char	2 bayt	Tek bir karakteri tutar.	`char a='h';`
string	Sınırsız	Metin tutar.	`string a="Ben bir zaman kaybıyım, beni boşver hocam";`

String türüne ayrıca char ve/veya string sabit ya da değişkenler + işaretiyle eklenip atanabilir. Örnekler:

```
char a='g';
string b="deneme";
string c=a+b+"Viki"+'m';
```

C#'ta hem metinsel hem de sayısal olmayan türler de vardır:

bool

Koşullu yapılarda kullanılır. Bool türünden değerlere `true`, `false` veya `2<1` gibi ifadeler örnek verilebilir. Örnekler:

```
bool b1=true;
bool b2=false;
bool b3=5>4;
```

object

Bu değişken türüne her türden veri atanabilir. Örnekler:

```
object a=10;
object b='k';
object c="metin";
object d=12.7f;
```

Aslında C#' taki bütün değişken türleri **object** türünden türemiştir. Bu yüzden **object** türü diğerlerinin taşıdığı bütün özellikleri taşır. Ancak şimdilik bunu düşünmenize gerek yok. Bu, nesneye dayalı programlamanın özellikleriyle ilgili.

Değişkeni Programımız İçinde Kullanma

Şimdiye kadar değişkenleri tanımlayıp ilk değer verdik. Şimdi değişkenleri programımızda kullanmanın zamanı geldi. Bir örnek:

```
using System;
class degiskenler
{
    static void Main()
    {
        byte a=5;
        Console.WriteLine(a);
    }
}
```

Burada a değişkeninin değerini ekrana yazdırdık. Başka bir örnek:

```
using System;
class degiskenler
{
    static void Main()
    {
        byte a=5;
        byte b=8;
        Console.WriteLine(a+b);
    }
}
```

Bu programda iki değişkenimizin değerlerinin toplamını ekrana yazdırdık.

```
using System;
class degiskenler
{
    static void Main()
    {
        string a="Viki", b="kitap";
        Console.WriteLine(a+b);
    }
}
```

Bu programda aynı satırda iki tane **string** değişkeni tanımladık ve değer verdik. Bu değişkenlerin değerlerini **WriteLine** metoduyla ekrana yan yana yazdırdık. **WriteLine** metodu + işaretini gördüğünde sayısal değişken ve değerleri toplar, **string** türünden değişken ve değerleri yan yana yazar, **char** türünden değişken ve değerlerin Unicode karşılıklarını toplar. Ancak tabii ki + karakterinin ayırdığı değişken veya değerler **char** ile **string** se **char** karakterle **string** metni yan yana yazar.

```
using System;
class degiskenler
{
    static void Main()
    {
        string a;
        a=Console.ReadLine();
        Console.WriteLine(a+" metnini yazdınız.");
    }
}
```

Sanırım şimdiye kadar yazdığımız en gelişmiş program buydu. Bu program kullanıcıdan bir metin alıp bunu ekrana "... metnini yazdınız." şeklinde yazıyor. Geçen derste **ReadLine** metodunu kullanıcı enter'a basana kadar programı bekletmek için kullanmıştık. Aslında **ReadLine** metodunun en yaygın kullanımı kullanıcının bilgi girişi yapmasını sağlamaktır. Dikkat ettiyseniz programımızda kullanıcının girdiği bilgi a değişkenine atanıyor. Sonra da **WriteLine** metoduyla ekrana bu değişken ve " metnini yazdınız." metni yan yana yazdırılıyor. Burada asıl önemsememiz gereken şey **Console.ReadLine()** ifadesinin **string** türünden bir değer gibi kullanılabilmesidir. C#'ta bunun gibi birçok metot bir değer gibi kullanılabilir. Tahmin edebileceğiniz üzere **WriteLine** gibi birçok metot da bir değer gibi kullanılamaz. Başka bir örnek:

```
using System;
class degiskenler
{
    static void Main()
    {
        int a, b;
        a=20;
        b=a;
        Console.WriteLine(b);
```

```
    }
}
```

Bu programda da görebileceğiniz gibi değişkenlere değer olarak başka bir değişkeni atayabiliriz. Ancak değerini atadığımız değişkene daha önceden bir değer atanmış olması gerekiyor. Burada `b` değişkenine `a` değişkeninin değeri atanıyor. Ancak aşağıdaki gibi bir kullanım kesinlikle hatalıdır.

```
using System;
class degiskenler
{
    static void Main()
    {
        Console.ReadLine()="metin";
        string a=Console.ReadLine();
        Console.WriteLine(a);
    }
}
```

Burada `Console.ReadLine()` ifadesi bir değişkenmiş gibi kullanılmaya çalışılıyor, ancak hatalı. Çünkü `Console.ReadLine()` ifadesi yalnızca bir değermiş gibi kullanılabilir.

Değişken adlandırma kuralları

Şimdiye kadar değişkenlere `ad`, `a` veya `b` gibi basit adlar verdik. Ancak aşağıdaki kuralları ihlal etmemek şartıyla değişkenlere istediğiniz adı verebilirsiniz.

- Değişken adları boşluk, simge içeremez.
- Değişkenler bir numerik karakterle başlayamaz.
- C#'ın diğer bütün komut, metot ve benzerlerinde olduğu gibi değişken adlarında büyük-küçük harf duyarlılığı vardır. Yani `degisken` isimli bir değişkenle `Degisken` isimli bir değişken birbirinden farklıdır.
- Değişken adları Türkçe karakterlerden(ğ,ü,ş,ö,ç,ı) oluşamaz. (Yeni versiyonlarda Türkçe karakterler kullanılabilir.)

Sık yapılan hatalar

C#'ta değişkenlerle ilgili sık yapılan hatalar şunlardır:

- Aynı satırda farklı türden değişkenler tanımlamaya çalışma. Örneğin aşağıdaki örnek hatalıdır:

```
int a, string b;
```

- Değişkene uygunsuz değer vermeye çalışma. Örnek:

```
int a;
a="metin";
```

- Değişkeni tanımlamadan ve/veya değişkene ilk değer vermeden değişkeni kullanmaya çalışma. Aşağıdaki örnekte iki değişkenin de kullanımı hatalıdır.

```
using System;
class degiskenler
{
    static void Main()
    {
        int b;
        Console.WriteLine(a);
        Console.WriteLine(b);
    }
}
```

- Değişken tanımlaması ve/veya değer vermeyi yanlış yerde yapma. Örnek:

```
using System;
class degiskenler
{
    int a=5;
    static void Main()
    {
        Console.WriteLine(a);
    }
}
```

Diğer `using` dışındaki bütün komutlarda da olduğu gibi değişken tanım ve değer vermelerini de `Main` bloğunun içinde yapmalıyız.

- Bazı değişken türlerindeki değişkenlere değer verirken eklenmesi gereken karakteri eklememek. Örnek:

```
using System;
class degiskenler
{
    static void Main()
    {
        float a=12.5;
        Console.WriteLine(a);
    }
}
```

- Ondalık sayıların ondalık kısmını ayırırken nokta (.) yerine virgül (,) kullanmak. Örnek:

```
using System;
class degiskenler
{
   static void Main()
   {
      float a=12,5f;
      Console.WriteLine(a);
   }
}
```

- Metinsel değişkenlerle matematiksel işlem yapmaya çalışmak. Örnek:

```
using System;
class degiskenler
{
   static void Main()
   {
      string a="1", b="2";
      int c=a+b;
      Console.WriteLine(a);
   }
}
```

- Bir değişkeni birden fazla kez tanımlamak. Örnek:

```
using System;
class degiskenler
{
   static void Main()
   {
      string a;
      string a="deneme";
      Console.WriteLine(a);
   }
}
```

- Değişkenlere değer verirken yanlış şekilde değer vermek. Örnek:

```
using System;
class degiskenler
{
   static void Main()
   {
      string a=deneme;
      Console.WriteLine(a);
   }
}
```

Sabit değişkenler

Programımızda bazen değeri hiç değişmeyecek değişkenler tanımlamak isteyebiliriz. Örneğin **pi** isimli float türünden bir değişken tanımlayıp buna 3.14 değerini verip programımızda pi sayısına ihtiyaç duyulduğunda bu değişkeni kullanabiliriz. Sabit değişkenlerin normal değişkenlerden farkı değişkeni değiştirmek istediğimizde ortaya çıkar, sabit olarak belirtilen değişkeni değiştirirsek derleyici hata verip programımızı derlemez. Bu daha çok uzun program kodlarında işe yarayabilir. Ayrıca sabit değişkenlere tanımlandığı satırda değer vermeliyiz. Herhangi bir değişkeni sabit olarak belirtmemiz için değişken türünden önce **const** anahtar sözcüğü kullanılır. Örnekler:

```
const int a = 5;
const string b ="deneme";
const char c ='s';
```

Aşağıdaki gibi bir durum, değişkene tanımlandığı satırda değer verilmediği için hatalıdır.

```
const int a;
a=5;
```

Sabit değişkenlere değer olarak sabit, sabit değişken ya da sabit ve/veya sabit değişkenlerden oluşan matematiksel ifadeler verilebilir. Örnek:

```
const int a=5;
const int b=a+7;
const int c=a*b;
```

Aşağıdaki gibi bir durum hatalıdır.

```
int a=5;
const int b=a+8;
```