C Sharp Programlama Dili/Tür dönüşümü

< C Sharp Programlama Dili
Gezinti kısmına atla Arama kısmına atla
Ders 4. Tür dönüşümü

Modern programlamada birçok kez değişkenlerde tür dönüşümüne ihtiyaç duyulur. Örneğin string türündeki sayılarla ("5" veya "2" gibi) matematiksel işlem yapmamız gerektiğinde tür dönüşümü yapmamız gerekir. Aslında bahsettiğimiz tam olarak tür dönüşümü değildir, sadece bir değişkenin değişik türdeki hâlinin başka bir değişkene atanmasıdır. Tür dönüşümleri bilinçli tür dönüşümü ve bilinçsiz tür dönüşümü olmak üzere ikiye ayrılır.

Bilinçsiz tür dönüşümü

Aslında bunu önceki derste görmüştük. Bu derste sadece adını koyuyoruz ve tanımlıyoruz: C#'ta düşük kapasiteli bir değişken, sabit ya da değişken ve sabitlerden oluşan matematiksel ifade daha yüksek kapasiteli bir değişkene atanabilir. Buna bilinçsiz tür dönüşümü denir, bunun için herhangi bir özel kod gerekmez.

```
using System;
class TurDonusumu
{
	static void Main()
	{
		byte a=5;
		short b=10;
		sbyte c=30;
		int d=a+b+c;
		string e="deneme";
		char f='k';
		object g=e+f+d;
		long h=d;
		float i=h;
		double j=i;
		double k=12.5f;
		Console.WriteLine(j+k);
	}
}
```

Bilinçsiz tür dönüşümüyle ilgili ilginç bir durum söz konusudur. `char` türünü kendisinden daha kapasiteli bir sayısal türe bilinçsiz olarak dönüştürebiliriz. Bu durumda ilgili karakterin Unicode karşılığı ilgili sayısal değişkene atanacaktır.

```
using System;
class TurDonusumu
{
	static void Main()
	{
		char a='h';
		int b=a;
		Console.WriteLine(b);
	}
}
```

Bilinçsiz tür dönüşümüyle ilgili diğer ilginç bir nokta ise `byte`, `sbyte`, `short` ve `ushort` türündeki değişkenlerle yapılan matematiksel işlemlerdir. Bu tür durumda oluşan matematiksel ifade intleşir. Yani aşağıdaki durumda programımız hata verir:

```
using System;
class TurDonusumu
{
	static void Main()
	{
		byte a=5, b=10;
		short d=2, e=9;
		byte f=a+b;
		short g=d+e;
		Console.WriteLine(f+g);
	}
}
```

Çünkü artık 8. satırdaki a+b ifadesi intleşmiş, ayrıca da 9. satırdaki d+e ifadesi de intleşmiştir, ve bunları en az int türdeki bir değişkene atayabiliriz. Size yardımcı olması açısında bilinçsiz tür dönüşümü yapılacabilecekler tablosu aşağıda verilmiştir:

Kaynak	Hedef
sbyte	short, int, float, long, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long, ulong	float, double, decimal
char	ushort, int, uint, long, ulong, float, double, decimal
float	double

Bilinçli tür dönüşümü

Bilinçli tür dönüşümü genellikle derleyicinin izin vermediği durumlarda kullanılır. Bilinçli tür dönüşümüyle küçük türün büyük türe dönüştürülmesi sağlanabilse de aslında bu gereksizdir, çünkü aynı şeyi bilinçsiz tür dönüşümüyle de yapabilirdik. Aşağıdaki programda bilinçli tür dönüşümü gerçekleştirilmektedir.

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

Programımızda da görebileceğiniz gibi `(byte)a` ifadesi, a değişkeninin byte hâlini tuttu. Aşağıdaki şekil bilinçli tür dönüşümünü anlatmaktadır.

- Eğer değişken adı kısmında tek bir değişken yoksa, bir ifade varsa parantez içine alınması gerekir.
- Bu şekilde tür dönüşümü değişkenlere uygulanabildiği gibi sabitlere de uygulanabilir:

```
using System;
class TurDonusumu
{
	static void Main()
	{
		byte b=(byte)12.5f;
		Console.WriteLine(b);
	}
}
```

- Reel türler tam sayı türlere dönüşürken ondalık kısım atılır.
- Bilinçsiz tür dönüşümüyle yalnızca küçük türler büyük türlere dönüşebiliyordu, yani veri kaybı olması imkansızdı. Halbuki bilinçli tür dönüşümünde veri kaybı gerçekleşebilir, eğer dönüşümünü yaptığımız değişkenin tuttuğu değer dönüştürülecek türün kapasitesinden büyükse veri kaybı gerçekleşir. Bu gibi durumlar için C#'ın `checked` ve `unchecked` adlı iki anahtar sözcüğü vardır.

checked

Kullanımı aşağıdaki gibidir:

```
using System;
class TurDonusumu
{
	static void Main()
	{
		int i=256;
		byte b;
		checked
		{
			b=(byte)i;
		}
		Console.WriteLine(b);
	}
}
```

i, byte'a çevrilirken veri kaybı gerçekleştiği için programımız çalışma zamanında hata verecektir. Ayrıca yeri gelmişken bir konuya değinmek istiyorum. Bu programımızda b değişkeni `checked` bloklarının içinde tanımlansaydı `checked` bloklarının dışında bu b değişkenine erişemezdik, çünkü değişkenler yalnızca tanımlandıkları en iç bloğun içinde aktiftirler. Buradaki "en iç" sözüne dikkat etmenizi öneririm.

```
{
	int b;
	{
		int a;
	}
}
```

Burada a değişkeni yalnızca en iç blokta aktif olmasına karşın, b değişkeni hem dıştaki hem içteki blokta aktiftir.

```
{
        {
                int a=0;
        }
        {
                int b=0;
        }
         Console.WriteLine(a+b);
}
```

Burada her iki değişken de `WriteLine` satırının olduğu yerde geçersiz olduğu için bu kod da geçersizdir. Çünkü her iki değişken de tanımlandıkları en iç bloğun dışında kullanılmaya çalışılıyor.

unchecked

Eğer programımızda veri kaybı olduğunda programın hata vermesini istemediğimiz kısımlar varsa ve bu kısımlar `checked` bloklarıyla çevrilmişse `unchecked` bloklarını kullanırız. `unchecked` , `checked` 'ın etkisini bloklarının içeriği çerçevesinde kırar. Programımızın varsayılan durumu `unchecked` olduğu için, yalnızca eğer programımızda `checked` edilmiş bir kısım var ve bu kısım `unchecked` etmek istediğimiz alanı kapsıyorsa gereklidir, diğer durumlarda gereksizdir. Örnek:

```
using System;
class TurDonusumu
{
        static void Main()
        {
                int i=256;
                int a=300;
                byte b, c;
                checked
                {
                        b=(byte)i;
                         unchecked
                        {
                                c=(byte)a;
                        }
                }
                Console.WriteLine(b);
        }
}
```

Programımız çalışma zamanında hata verir, ancak bunun sebebi `c=(byte)a;` satırı değil, `b=(byte)i;` satırıdır.

object türüyle ilgili kurallar

C#'taki en ilginç türün `object` olduğunu geçen derste söylemiştik, işte bu ilginç türün daha başka kuralları:

object türündeki bir değişkene başka herhangi bir türdeki değişken ya da sabit (string dışında) + işaretiyle eklenemez. Örneğin aşağıdaki gibi bir durum hatalıdır:

```
using System;
class TurDonusumu
{
	static void Main()
	{
		object a='5';
		int b=4;
		Console.WriteLine(a+b);
	}
}
```

object türündeki bir değişkene herhangi bir türdeki değişken ya da sabit atanabilir (bilinçsiz tür dönüşümü). Ancak object türündeki herhangi bir değişkeni başka bir türe çevirmek için tür dönüştürme işlemi kullanılır. Ayrıca da dönüştürülen türlerin uyumlu olması gerekir. Yani object türüne nasıl değer atandığı önemlidir. Örneğin aşağıdaki gibi bir durum hatalıdır:

```
using System;
class TurDonusumu
{
	static void Main()
	{
		object a="5";
		int b=(int)a;
		Console.WriteLine(b);
	}
}
```

Ancak aşağıdaki gibi bir kullanım doğrudur.

```
using System;
class TurDonusumu
{
	static void Main()
	{
		object a='k';
		char b=(char)a;
		Console.WriteLine(b);
	}
}
```

Aynı satırda çifte dönüşüm yapılamaz. Aşağıdaki gibi bir kullanım hatalıdır.

```
using System;
class TurDonusumu
{
	static void Main()
	{
		object a='c';
		int b=(int)a;
		Console.WriteLine(b);
	}
}
```

Burada önce objectin chara, sonra da charın inte dönüşümü ayrı ayrı yapılmalıydı. Aşağıdaki gibi bir kullanım hatalıdır:

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

Çünkü derleyici harfsiz tam sayı sayıları int sayar, dolayısıyla da burada uygunsuz bir tür dönüşümünden bahsedebiliriz. Yani 7. satırda a inte dönüştürülmeliydi. Ancak aşağıdaki gibi bir kullanım doğrudur:

```
using System;
class TurDonusumu
{
	static void Main()
	{
		byte m=5;
		object a=m;
		byte b=(byte)a;
		Console.WriteLine(b);
	}
}
```

Bu program hata vermez. Çünkü derleyici a'nın gizli türünün byte olduğunu biliyor.

string türüyle ilgili dönüşümler

Dikkat ettiyseniz şimdilik sadece sayısal türleri, char ve object (uygun şekilde değer verilmiş olması şartıyla) türünü kendi aralarında dönüştürebiliyoruz. Şimdi göreceğimiz metotlarla string türünü bu türlere ve bu türleri de string türüne dönüştürebileceğiz.

x.ToString()

Bu metot x değişken ya da sabitini stringe çevirip tutar. Örnekler:

```
using System;
class TurDonusumu
{
	static void Main()
	{
		string b=3.ToString();
		Console.WriteLine(b);
	}
}
```

veya

```
using System;
class TurDonusumu
{
	static void Main()
	{
		int a=6;
		string b=a.ToString();
		Console.WriteLine(b);
	}
}
```

veya

```
using System;
class TurDonusumu
{
	static void Main()
	{
		string b=12.5f.ToString();
		Console.WriteLine(b);
	}
}
```

veya

```
using System;
class TurDonusumu
{
	static void Main()
	{
		string b='k'.ToString();
		Console.WriteLine(b);
	}
}
```

`ToString()` metodu `System` isim alanında olduğu için programımızın en başında `using System;` satırının bulunması bu metodu kullanabilmemiz için yeterlidir. string türüyle ilgili bilmeniz gereken bir şey daha var:

```
using System;
class TurDonusumu
{
       static void Main()
       {
              string b=5+"deneme"+'k'+12.5f;
              Console.WriteLine(b);
       }
}
```

Böyle bir kod mümkündür. Ayrıca şöyle bir kod da mümkündür:

```
using System;
class TurDonusumu
{
       static void Main()
       {
              int a=5;
              char m='k';
              string r="deneme";
              float f=12.5f;
              string b=a+m+r+f;
              Console.WriteLine(b);
       }
}
```

Burada dikkat etmemiz gereken şey yan yana gelen sayısal ve char değer ve değişkenlerle matematiksel toplama yapılmasına rağmen bunların yan yana gelmemesi durumunda da normal yan yana yazma olayının gerçekleşmesidir. Ancak tabii ki şöyle bir durum hatalıdır:

```
using System;
class TurDonusumu
{
       static void Main()
       {
              int a=5;
              char m='k';
              float f=12.5f;
              string b=a+m+f;
              Console.WriteLine(b);
       }
}
```

Yani bir sabit ve/veya değişken grubunun bir stringe atanabilmesi için ifadenin içinde en az bir string olmalıdır.

System.Convert sınıfı ile tür dönüşümü

`System` isim alanının altındaki `Convert` sınıfı içinde tür dönüşümü yapabileceğimiz birçok metot bulunur. Bu metotlarla hemen hemen her türü her türe dönüştürebiliriz. Ancak bunun için değişken türlerinin CTS karşılıklarını bilmeliyiz. Değişken türleri ve CTS karşılıkları aşağıdaki tabloda listelenmiştir.

Tür	CTS karşılığı
bool	Boolean
byte	Byte
sbyte	Sbyte
short	Int16
ushort	UInt16
int	Int32
uint	UInt32
long	Int64
ulong	UInt64
float	Single
double	Double
decimal	Decimal
char	Char

Şimdi sıra geldi bu metotlara:

- Convert.ToBoolean(x)
- Convert.ToByte(x)
- Convert.ToSbyte(x)
- Convert.ToInt16(x)
- Convert.ToUInt16(x)
- Convert.ToInt32(x)
- Convert.ToUInt32(x)
- Convert.ToInt64(x)
- Convert.ToUInt64(x)
- Convert.ToSingle(x)
- Convert.ToDouble(x)
- Convert.ToDecimal(x)
- Convert.ToChar(x)

Bu metotlar x'i söz konusu türe dönüştürüp o türde tutarlar. x bir değişken, sabit ya da ifade olabilir. x, ifade olduğu zaman önce bu ifade mevcut türe göre işlenir, sonra tür dönüşümü gerçekleşir. Kullanımına bir örnek:

```
using System;
class TurDonusumu
{
       static void Main()
       {
              string s1, s2;
              int sayi1, sayi2;
              int Toplam;
              Console.Write("Birinci sayıyı girin: ");
              s1=Console.ReadLine();
              Console.Write("İkinci sayıyı girin: ");
              s2=Console.ReadLine();
              sayi1=Convert.ToInt32(s1);
              sayi2=Convert.ToInt32(s2);
              Toplam=sayi1+sayi2;
              Console.Write("Toplam= "+Toplam);
       }
}
```

Bu programla kullanıcıdan iki sayı alıp bunların toplamını ekrana yazdırdık. Ayrıca burada şimdiye kadar kullanmadığımız `Write` metodunu kullandık. Bu metodun `WriteLine` 'dan tek farkı ekrana yazıyı yazdıktan sonra imlecin alt satıra inmemesi.