

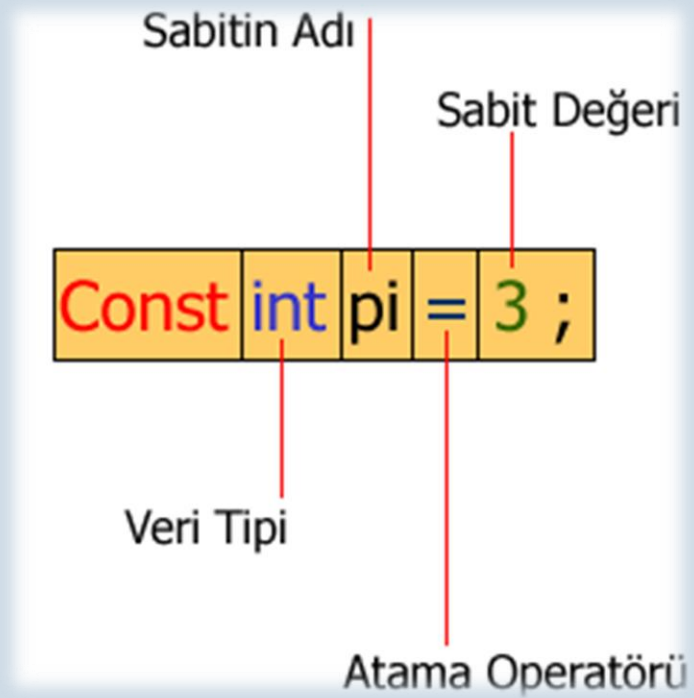
WEB PROGRAM LAMA

Adem AKKUŞ

Bilgisayar Mühendisi
Bilişim Teknolojileri Öğrt.
Eğitmen

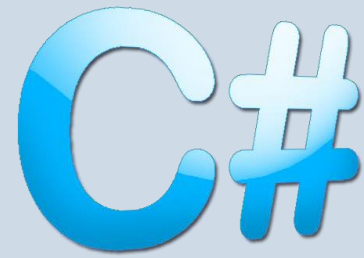
Sabit (Constant) Tanımlama

Bir program boyunca değerinin değişmeyeceğini düşündüğümüz veriler sabit (Constant) olarak tanımlanır.



- Sabit değişkenlerin değerlerini programın çalışması sırasında değiştirmeye çalışmak hataya yol açar.
- Sabitlere ilk değer verilirken sabitler kullanılır.
- Sabitler içsel tasarım olarak static olduklarından, ayrıca static olarak tanımlamaya çalışmak hataya yol açar.

Sabitlerle ilgili Örnek



```
using System;
public class faaliyet_alani
{
    static void Main()
    {
        int x=5,y;
        y=10;
        const int t=x+y;
    }
}
```

C:\WINDOWS\system32\cmd.exe

F:\c#\bolun2\compiled>csc sabitler.cs

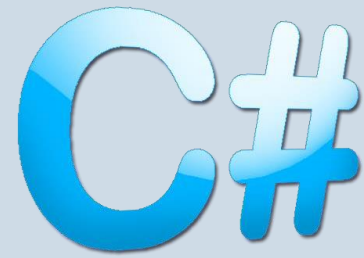
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.1433

for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727

Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

sabitler.cs(8,14): error CS0133: The expression being assigned to 't' must be constant

F:\c#\bolun2\compiled>



Sabitlerle ilgili Örnek

```
using System;
public class faaliyet_alani
{
    static void Main()
    {
        const int x=5,y=10;
        const int t=x+y;
        Console.WriteLine(t);
    }
}
```

```
C:\WINDOWS\system32\cmd.exe
02/19/2008 10:31 PM <DIR> .
02/19/2008 10:31 PM <DIR> ..
02/19/2008 10:10 PM      181 faaliyet_alani.cs
02/19/2008 10:11 PM    3,072 faaliyet_alani.exe
02/19/2008 03:45 PM     103 ilk_program1.cs
02/19/2008 04:01 PM    3,072 ilk_program1.exe
02/19/2008 04:38 PM     111 ilk_program2.cs
02/19/2008 04:38 PM    3,072 ilk_program2.exe
02/19/2008 04:39 PM     182 ilk_program3.cs
02/19/2008 04:39 PM    3,072 ilk_program3.exe
02/19/2008 10:27 PM     130 sabitler.cs
02/19/2008 10:31 PM     153 sabitler2.cs
      10 File(s)      13,148 bytes
       2 Dir(s)  51,602,219,008 bytes free

F:\c#\bolun2\compiled>csc sabitler2.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.1433
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

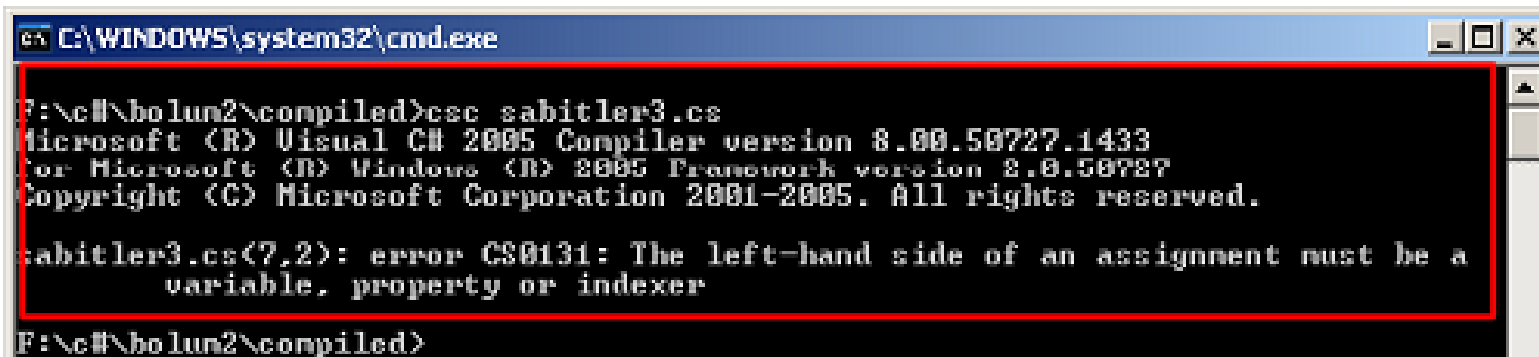
F:\c#\bolun2\compiled>sabitler2
15

F:\c#\bolun2\compiled>_
```

Sabitlerle ilgili Örnek

```
using System;
public class faaliyet_alani
{ static void Main()
{
    const int x=5,y=10;
    x+=2;
    const int t=x+y;
    Console.WriteLine(t);
}
}
```

Derleme gerçekleşmemektedir.
Neden?

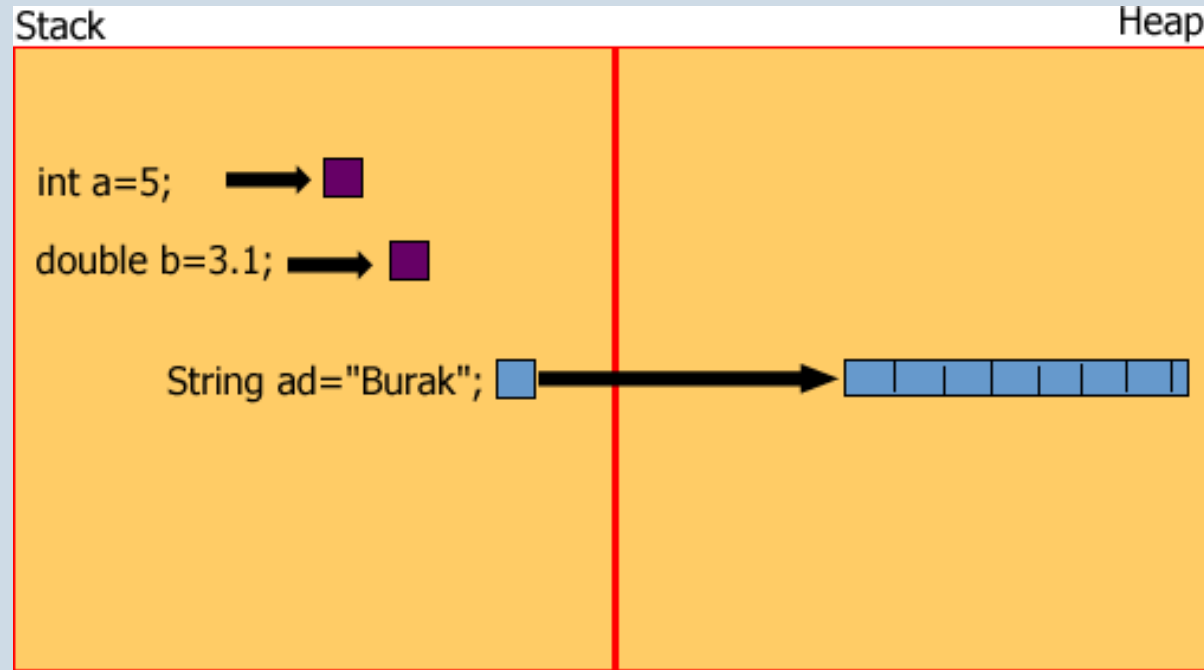
A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The command prompt shows the command 'csc sabitler3.cs' being executed. The output shows the Visual C# 2005 Compiler version 8.00.50727.1433 and the .NET Framework version 2.0.50727. The error message is 'sabitler3.cs(7,2): error CS0131: The left-hand side of an assignment must be a variable, property or indexer'. The error message is highlighted with a red rectangle.

```
C:\WINDOWS\system32\cmd.exe
F:\c#\bolun2\compiled>csc sabitler3.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.1433
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

sabitler3.cs(7,2): error CS0131: The left-hand side of an assignment must be a
variable, property or indexer

F:\c#\bolun2\compiled>
```

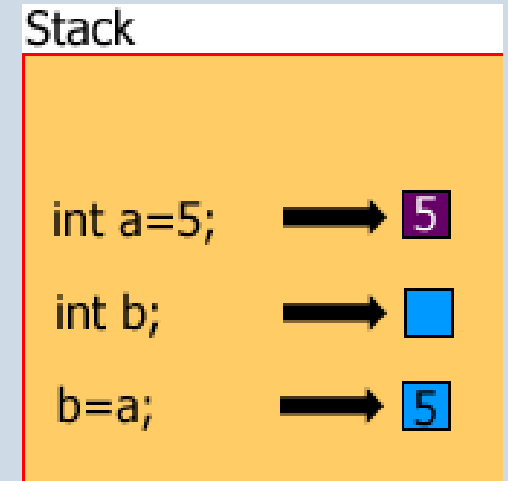
Değer ve Referans Tipleri




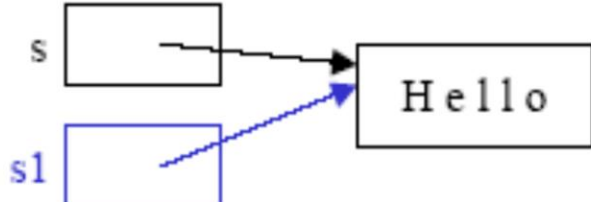
- Değer tipleri değişkenin değerini direkt bellek bölgesinden alırlar.
 - Referans tipleri ise başka bir nesneye referans olarak kullanılırlar. Diğer bir deyişle referans tipleri, heap alanında oluşturulan nesnelerin adreslerini saklarlar.
-
- Değer tipler, **stack** bölgesinde oluşturulurlar. Referans tipleri ise kullanımı biraz daha sınırlı olan **heap** bellek bölgesinde saklanırlar.

Değer ve Referans Tipleri

- Temel veri tipleri (int, double, float...) değer tipi; herhangi bir sınıf türü ise referans tipidir.
- İki değer tipi nesnesi birbirine eşitlenirken değişkenlerde saklanan değerler kopyalanarak eşitlenir ve bu durumda iki yeni bağımsız nesne elde edilmiş olur. Birinin değerini değiştirmek diğerini etkilemez.
- Fakat, iki referans tipi birbirine eşitlendiğinde bu nesnelerde tutulan veriler kopyalanmaz, işlem yapılan nesnelerin heap bölgesindeki adresleridir.
- İki nesne heap bölgesinde aynı yeri gösterdiği için, birinde yapılan değişiklik diğerini de etkileyecektir.



Değer ve Referans Tipleri

	Value Types	Reference Types
variable contains	value	reference
stored on	stack	heap
initialisation	0, false, '\0'	null
assignment	copies the value	copies the reference
example	<pre>int i = 17; int j = i;</pre> 	<pre>string s = "Hello"; string s1 = s;</pre> 

Değer ve Referans Tipleri

- CTS sayesinde, .NET platformu için geliştirilen bütün diller aynı veri tiplerini kullanırlar. Tek değişen veri türlerini tanımlama yöntemi ve söz dizimidir.
- C#'da önceden tanımlanmış temel veri tipleri 15 tanedir. (13 tanesi değer tipi, 2 tanesi ise referans tipidir)

Değer Tipleri (Value Types)

- Değer tiplerinin tamamı **Object** denilen bir nesneden türemiştir. C#'da her nesne yada veri tipi aslında **Object** tipidir.
- Değer tiplerinde bir nesnenin değeri direkt olarak saklıdır. Tanımlanan değer tiplerine aşağıdaki şekilde ilk değer ataması yapılabilir.

```
int a=3,b;  
b=a;
```

- Bu noktada **a** üzerindeki değişikliklerden **b** etkilenmeyecektir.

Değer Tipleri (Value Types)

- Değer tiplerine ilk değer verme;

```
a=new int(); //yapıcı çalışır.(referans tip)  
a=0;
```

- Yukarıdaki iki satırda aynı işlemi yapar.

```
float b;//derleyici hatası, atama yapılması gerekir.  
'Error2 Use of unassigned local variable b'
```

```
float b = new float(); //hata vermez yapıcı metot çalıştı
```

```
b=3.21f //yeni atama yapılıyor
```

Değer Tipleri (Value Types)

Veri Tipi	Varsayılan Değer
bool	False
byte	0
*** char	'\0'
*** decimal	0.0M
*** double	0.0D
enum	Enum sabitindeki, 0 indisli değer.
float	0.0F
Int	0
*** long	0L
SByte	0
Short	0
Struct	Yapıdaki tüm değer tipleri varsayılan değer, tüm referans tipleride null değere atanır.
UInt	0
Ulong	0
UShort	0

Değer Tipleri (Value Types)

C# Tipi	.NET Framework	Tanım	Değer Aralığı
object	System.Object	Tüm CTS türleri için temel sınıf	-
bool	System.Boolean	Mantıksal Doğru/Yanlış	true ya da false
byte	System.Byte	8 bit işaretli tamsayı	0 ~ 255
sbyte	System.SByte	8 bit işaretli tamsayı	128 ~ 127
char	System.Char	Karakterleri temsil eder	16 Unicode karakterleri
decimal	System.Decimal	128 bit ondalıklı sayı	$\pm 1,5 \cdot 10^{-28} \sim \pm 7,9 \cdot 10^{28}$
double	System.Double	64 bit çift kayan sayı	$\pm 5 \cdot 10^{-324} \sim \pm 1,7 \cdot 10^{308}$
float	System.Single	32 bit tek kayan sayı	$\pm 1,5 \cdot 10^{-45} \sim \pm 3,4 \cdot 10^{38}$
int	System.Int32	32 bit işaretli tamsayı	-2.147.483.648 ~ 2.147.483.647
uint	System.UInt32	32 bit işaretli tamsayı	0 ~ 4.294.967.295
long	System.Int64	64 bit işaretli tamsayı	9.223.372.036.854.775.808 ~ -9.223.372.036.854.775.807
ulong	System.UInt64	64 bit işaretli tamsayı	0 ~ 18.446.744.073.709.551.615
short	System.Int16	16 bit işaretli tamsayı	-32.768 ~ 32.767
ushort	System.UInt16	16 bit işaretli tamsayı	0 ~ 65.535
string	System.String	Karakter Dizisi	Unicode Karakter Dizisi

Referans Tipleri (Reference Types)

Önceden tanımlanmış iki referans türü vardır.

string veri türü

object veri türü

- Object türü C#'ta bütün türlerin türediği sınıftır. Diğer bir deyişle Object türünden bir nesneye her hangi bir veri türünden nesneyi atayabiliriz.
- Object türü özelleştirilerek farklı amaçlara yönelik kullanılabilirler. Object'e eşleştirme (Boxing) işlemi ve tersi, Object'i dönüştürme (Unboxing)

String Türü

- Referans türünden olan stringler, türü Unicode karakterlerden oluşan bir dizi gibi algılanmalıdır.

Strings1="Merhaba";

Strings2=".NET";

Strings3=s1+s2;

} Stringleri arka arkaya eklemek için + operatörü kullanılır.

String Türü

- Özel anlamlar içeren karakterleri ifade etmek için `\\` ifadesini kullanırız (escape). Örneğin:

`String path="C:\\WINDOWS\\assembly"`

- String içinde görünen ifadenin aynısını belirtmek için string ifadesinin önüne `@` işareti konulur. Örneğin:

`String path=@"C:\\WINDOWS\\assembly"`

Object Veri Türü

- Her nesne object türünden olduğu için bütün değerler ve nesneler object türünden bir değişkene atanabilir.

Örnek:

```
using System;
public class varsayilan_degerler
{
    static void Main()
    {
        object x;
        x=10;
        Console.WriteLine(x.GetType());
        x="B";
        Console.WriteLine(x.GetType());
        x=8.78F;
        Console.WriteLine(x.GetType());
        x=false;
        Console.WriteLine(x.GetType());
        x=5.489M;
        Console.WriteLine(x.GetType());
    }
}
```

Tür Dönüşümleri

Farklı türden değişkenleri aynı ifade içerisinde kullanabilmek için tür dönüşümleri yapılır

Bilinçli Tür Dönüşümü

Büyük Tür → Küçük Tür

Küçük Tür → Büyük Tür

Bilinçsiz Tür Dönüşümü

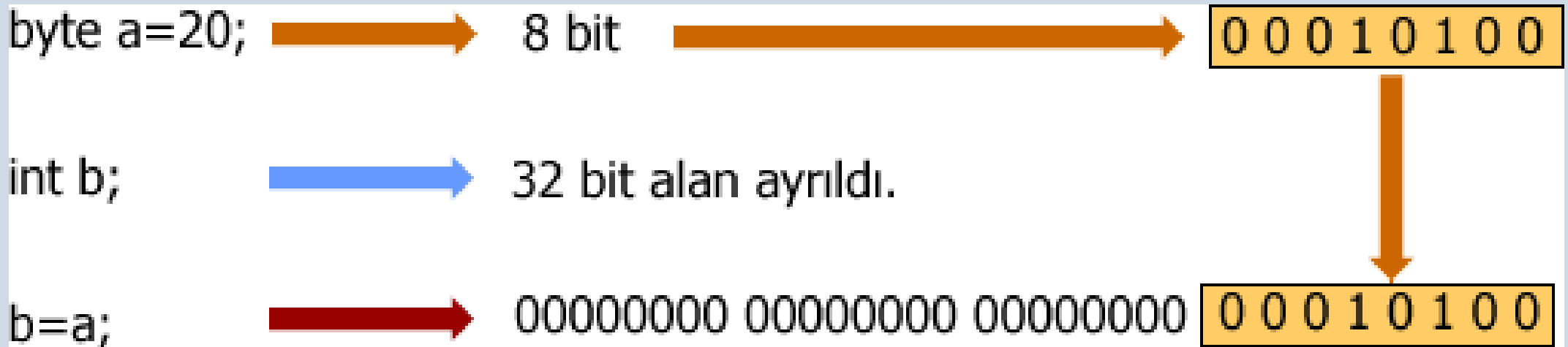
Büyük Tür → Küçük Tür
Yasaklanmıştır

Küçük Tür → Büyük Tür

Bilinçsiz (Implicit) Tür Dönüşümü

Küçük Tür -> Büyük Tür

- Tür dönüşümünde herhangi bir veri kaybı olmaz.
- Fazla olan bitler 0 ile beslenir.



Bilinçsiz (Implicit) Tür Dönüşümü

Küçük Tür -> Büyük Tür

- Byte türleri arasında aritmetiksel işlemler yapılırken sonuç mutlaka int yada daha büyük türden bir değişkene aktarılmalıdır.

```
byte a=5;  
byte b=3;  
byte c=a+b;
```



Cannot implicitly convert type 'int' to 'byte'

```
using System;  
public class Tur_donusumu1  
{  
    static void Main()  
    {  
        int x=5;  
        float a;  
        a=x;  
        Console.WriteLine(a);  
    }  
}
```

Bilinçsiz Tür Dönüşümü

Küçük Tür -> Büyük Tür

Tür	Dönüştürüleceği Tür
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

Bilinçsiz Tür Dönüşümü

Örnek 1

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        byte a=20;
        int b;
        b=a;
        Console.WriteLine(b);

        float f=20f;
        double d;
        d=f;
        Console.WriteLine(d);

        char c='a';
        decimal m;
        m=c;
        Console.WriteLine(m);
    }
}
```

Örnek 2

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        short b1=100;
        char c1=b1;
        bool b2=true;
        int i1=b2;
        double d1=10.2;
        int i2=d1;
        decimal m1=20.6M;
        double d2=m1;
        byte bt1=65;
        char c2=bt1;
        float f1=34.78F;
        decimal d3=f1;
    }
}
```

Bilinçsiz Tür Dönüşümü

Örnek 3

```
using System;
class Otomatik_tip
{
    public static void Main() {
        int a;
        float b=32.32f;
        double c;
        c=b;
        Console.WriteLine("b'nin değeri="+b+"\nc'nin değeri="+c;
    }
}
```



```
C:\Documents and Settings\SoNDuRaK\ConsoleApp...
b'nin değeri=32,32
c'nin değeri=32,3199996948242
Press any key to continue_
```

Bilinçsiz Tür Dönüşümü

Bazı türler arasında tür dönüşümü yapmak mümkün değildir.

Bunlar:

- a) Bool, decimal ve double türünden herhangi bir türe
- b) Herhangi bir türden char türüne
- c) Float ve decimal türünden herhangi bir türe (float türünden double türüne dönüşüm hariç)

Bilinçsiz Tür Dönüşümü

- Büyük türün küçük türe dönüştürülmesi: Büyük türlerin küçük türlere otomatik dönüştürülmesi C#'da yasaklanmıştır. Eğer bu tür bir dönüştürme (bilinçsiz olarak) mümkün olsaydı bir takım veri kayıpları yaşanacaktır.
- İstenmeyen durum. Ancak “**()**”cast operatörü ile yapılır.

Bilinçli (Explicit) Tür Dönüşümü

- Derleyicinin izin vermediği tür dönüşümlerinde yapılır.
- Veri kayıplarına sebep olabilir.
- Küçük Tür -> Büyük Tür dönüşümleri, bilinçsiz türde olduğu gibidir.
- Dönüşümler için tür dönüşüm operatörleri kullanılır. Tür dönüştürme operatörü parantez içinde değişken yada sabitten önce yazılır.

```
byte a = 10;  
int b;
```

```
b = (int)a;
```

tür dönüştürme operatörü.
byte tipindeki a değişkenini int tipine dönüştürür.

Bilinçli Tür Dönüşümü

Büyük Tür -> Küçük Tür

Bilinçli olarak büyük türler, küçük türlere çevrilirken veri kayıpları olabilir.

`int a = 256;` → 32 bit alan. → 00000000 00000000 00000000 1 00000000

`byte b;` → 8 bit alan tahsis edildi.

`b = (byte)a;` → int veri byte'a dönüştürüldü. → 00000000

Bilinçli Tür Dönüşümü

Büyük Tür -> Küçük Tür

Bilinçsiz yapılan tür dönüşümlerinde büyük türler, küçük türlere dönüştürülemezdi. Eğer tür dönüştürme operatörü kullanılırsa bu işlem mümkün olur. Örneğin;

```
using System;
public class Tur_donusumu1
{
    static void Main()
    {
        int a=400;
        byte b=(byte)a;
        Console.WriteLine(b);
    }
}
```

Programı çalıştırdığımızda ekrana 144 yazdırdı. Neden?

int a=400 : 00000000 00000000 00000001 10010000

(byte)i : 10010000

↓

144 Dec.

Tür Dönüşümleri

```
using System;
class Bilinçli_tip {
public static void Main() {
    double d1,d2; int i; byte b; char c; uint u; short s; d1=5.0;d2=4.0;
    //double int e dönüştü veri kaybı var,virgülden sonrası atılır
    i= (int) (d1/d2);Console.WriteLine("Double integere çevrildi="+i); Console.WriteLine();
    //int'i byte dönüştür, Veri kaybı yok.
    i=123; b=(byte) i; Console.WriteLine("i'nin değeri="+i+" iken b'nin değeri="+b);
    //Veri kaybı var.
    i=258; b=(byte) i; Console.WriteLine("i'nin değeri="+i+" iken b'nin değeri="+b);
    Console.WriteLine();
    //uint'i short'a dönüştür
    u=32146; s=(short) u; //Veri kaybı yok.
    Console.WriteLine("u'nun değeri="+u+" iken s'nin değeri="+s);
    u=35000; s=(short) u; //Veri kaybı var.
    Console.WriteLine("u'nun değeri="+u+" iken s'nin değeri="+s); Console.WriteLine();
    //int'i char'a dönüştür.
    i=90; c=(char) i; Console.WriteLine(i+"sayısının char'a dönüştürürsek="+c+" olur"); }
}
```

```
Double integere çevrildi=1
i'nin değeri=123 iken b'nin değeri=123
i'nin değeri=258 iken b'nin değeri=2
u'nun değeri=32146 iken s'nin değeri=32146
u'nun değeri=35000 iken s'nin değeri=-30536
90sayısının char'a dönüştürürsek=Z olur
Press any key to continue_
```

Checked ve Unchecked

- Bilinçli tür dönüşümleri sonucu oluşabilecek **veri kayıplarının önüne geçmek için**, tür dönüşümlerinin yer aldığı kodları checked blokları içine alırız. Böylece, exception türetilmesi sağlanır.
- Checked bloklarında tanımlanan değişkenler **blok dışında tanımlanamazlar**.
- Checked kod bloğunun içinde **veri kayıplarına neden olabilecek durumların göz ardı edilmesi gerekirse unchecked** blokları kullanılır.

```
// unchecked checked işlemini ters çevirir.  
using System;  
class turdonusum {  
    static void Main() {  
        int i=256;  
        byte b;  
        checked //Taşma olduğundan program hata verir.  
        {  
            b=(byte) i;  
        }  
        Console.WriteLine(b);  
    }  
}
```


Checked ve Unchecked

- Checked bir blok oluşturduğu için, içinde yapılan değişken tanımlamaları dış bloklarda kullanılamaz.

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

Checked ve Unchecked

- Normal şartlarda yapılan işlemler “unchecked” dir. Böyle bir ifadenin konmasının nedeni uzun “checked” blokların oluşturulması istenebilir.
- Bu durumlarda çok fazla blok oluşturmamak için “unchecked” ifadesi kullanılabilir.

```
using System;
class TurDonusumu11
{
    static void Main()
    {
        int i1 = 255;
        int i2 = 500;
        byte b, c;

        checked
        {
            b = (byte)i1;
            Console.WriteLine(b);

            unchecked
            {
                c = (byte)i2;
            }
            Console.WriteLine(c);
        }
    }
}
```


SORULAR

