

BİL 201

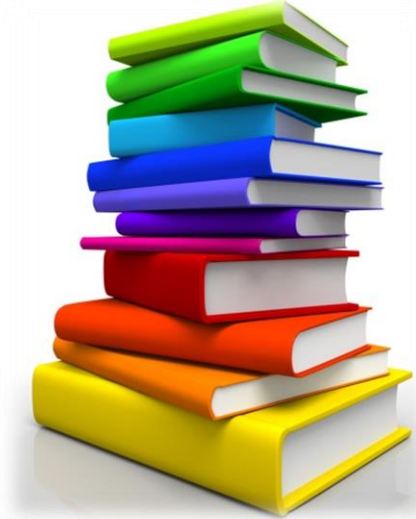
NESNE YÖNELİMLİ PROGRAMLAMA (NYP)

DERS #2

Öğretim Üyesi: Doç. Dr. Deniz Kılınç

BÖLÜM 2 - C# Programlama Dili Elemanları

- Bu bölümde aşağıdaki konular anlatılacaktır
 - Sabitler
 - Değişkenler
 - Tip Dönüşümü
 - Operatörler
 - Döngüler
 - Örnekler



Sabitler

- Sabitler, içeriği sabit olan değer ve ifadelerin saklanması için kullanılan değişken türleridir.
- Başlangıçta sabit değişkene bir değer vererek, ihtiyaç duyulduğunda bu değişkeni kullanabiliriz.
- Değerini sonradan değiştirmek istediğimizde derleyici, **derleme zamanı** hatası verir.
- Sabit değişkenlerinin **tanımlandığı satırda değer verilmek zorundadır.**
- C# programlama dilinde **const** anahtar sözcüğü kullanılarak tanımlanır.

```
const double pi = 3.141592;  
const int sayi = 9;  
const char ilkKarakter = 'a';
```

Değişkenler

- Program içerisinde üretilen değerleri **bellette geçici olarak** saklamak için kullanılırlar.
- İlk değerlerinin atanması **zorunlu değildir**. Program içerisinde de sonradan değer ataması yapılabilir.
- Aynı satırda aynı tipte birden fazla değişken tanımlaması ve değer ataması yapılabilir.
- Global olarak tanımlanmayan değişkenler **sadece tanımlandıkları metot içerisinde** kullanılabilirler.
- Değişkenlerin tipini öğrenmek için **GetType()** metodu kullanılır.

[değişkenin tipi] [değişkenin adı] = [ilk değer] ;

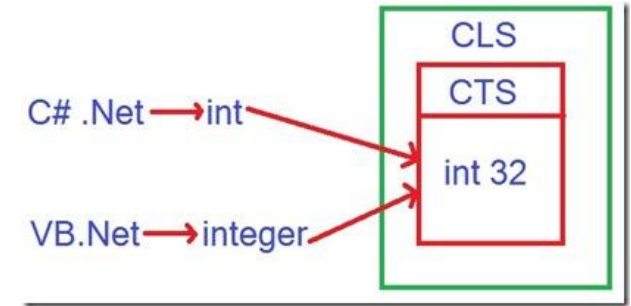
```
int yariCap = 12;  
float alan;  
int yukseklik = 2, taban = 4;  
string ad = "Ada", soyad = "Kılınç";
```

Değişken Türleri

Değişken Türü	Boyutu	CTS	Değer Aralığı	Örnek
byte	1 bayt	System.Byte	0, ..., 255 (tam sayı)	byte a=5;
sbyte	1 bayt	System.Byte	-128, ..., 127 (tam sayı)	sbyte a=5;
short	2 bayt	System.Int16	-32768, ..., 32767 (tam sayı)	short a=5;
ushort	2 bayt	System.UInt16	0, ..., 65535 (tam sayı)	ushort a=5;
int	4 bayt	System.Int32	-2147483648, ..., 2147483647 (tam sayı)	int a=5;
uint	4 bayt	System.UInt32	0, ..., 4294967295 (tam sayı)	uint a=5;
long	8 bayt	System.Int64	-9223372036854775808, ..., 9223372036854775807 (tam sayı)	long a=5;
ulong	8 bayt	System.UInt64	0, ..., 18446744073709551615 (tam sayı)	ulong a=5;
float	4 bayt	System.Single	$\pm 1.5 \cdot 10^{-45}$, ..., $\pm 3.4 \cdot 10^{38}$ (reel sayı)	float a=5F; veya float a=5f;
double	8 bayt	System.Double	$\pm 5.0 \cdot 10^{-324}$, ..., $\pm 1.7 \cdot 10^{308}$ (reel sayı)	double a=5; veya double a=5d; veya double a=5D;
decimal	16 bayt	System.Decimal	$\pm 1.5 \cdot 10^{-28}$, ..., $\pm 7.9 \cdot 10^{28}$ (reel sayı)	decimal a=5M; veya decimal a=5m;
bool	1 bayt	System.Boolean	True ya da False	bool cevap = False;
char	1 bayt	System.Char	16 Bit Unicode Karakter	char harf = 'a';

Değişken Türleri (devam...)

- Değişken tanımlamaları istenirse programlama diline ait değişken tipi tanımlamasıyla, istenirse **.NET** uygulama geliştirme platformunun bir özelliği olan **CTS** (Common Type System) tanımlaması ile yapılabilir.
- .NET uygulama geliştirme platformunda **programlama dilleri arasındaki standardizasyonu** CTS dediğimiz sistem sağlar.
- C#.NET ile program yazarken tamsayı veri tipinde değişken tanımlamak için **int** ifadesini kullanırız.
- VB.NET'te ise tamsayı veri tipi **Integer** olarak geçer.
- Her iki dil farklı olsa dahi CTS dediğimiz sistem sayesinde **int** ve **Integer** tanımlamalar MSIL'e çevrilirken CTS sisteminde tanımlanan **System.Int32** olarak ara-dile çevrilecektir.



```
System.Int32 yariCap = 4;  
System.Char ilkKarakter = 'a';
```

```
int yariCap = 12;  
char ilkKarakter = 'a';
```

Değişken Tip Dönüşümleri (Type-Casting)

- Üstü Kapalı (**Implicit**) Tip Dönüşümü
- Açık (**Explicit**) Tip Dönüşümü
- **Tostring()** Metodu İle Tip Dönüşümü
- **Convert** Sınıfı İle Tip Dönüşümü

Üstü Kapalı (Implicit) Tip Dönüşümü

- Değişkenin tanımlandığı veri tipinden daha yüksek kapasiteli bir veri tipindeki değişkene atanabilir. Bu duruma **Üstü Kapalı (Implicit) Tür Dönüşümü** denir.

```
int sayi1 = 10;  
float sayi2 = sayi1; //sayi1 değişkeni bulunduğu int veri tipinden daha kapasiteli float veri tipine dönüştü  
  
char cinsiyet = 'K';  
int Cinsiyet = cinsiyet; //cinsiyet değişkeni bulunduğu char veri tipinden daha kapasit. int veri tipine dönüş.  
  
short sayi3 = 10882;  
long sayi4 = sayi3; //sayi3 değişkeni bulunduğu short veri tipinden daha kapasiteli long veri tipine dönüştü
```

Düşük kapasiteden yüksek kapasiteye dönüşürken sorun yok.

Açık (Explicit) Tip Dönüşümü

- **Açık Tip Dönüşümü**, derleyicinin izin vermediği durumlarda kullanılır.
[Değişken] = (hedef tür) [değişken adı]

```
int a = 120;  
byte b;
```

```
//Aşağıdaki kodu çalıştırdığımızda aşağıdaki gibi bir hata alırız  
//Cannot implicitly convert type 'int' to 'byte'...  
b = a;
```

```
/*a değişkenini daha az kapasiteli byte'a tipine çeviriyoruz.  
Şanslıyız çünkü değer 0-255 arasında. Yani veri kaybımız olmadı.*/  
b = (byte)a;
```

```
/*a değişkenini yine daha az kapasiteli byte'a tipine çeviriyoruz.  
Değer 255'den büyük olduğu için veri kaybımız yaşadık.*/
```

```
a = 498;  
b = (byte)a;
```

ToString() Metodu İle Tip Dönüşümü

- **ToString()** metodu Bir değişken veya sabitin değerini **string** veri tipine dönüştürerek tutar.
- Kullanımı: **[değişken adı]** . ToString();

```
bool a = true;  
Console.WriteLine(a.ToString());
```

```
int sayi = 98;  
Console.WriteLine(sayi.ToString());
```

```
float sayi2 = 98.875f;  
Console.WriteLine(sayi2.ToString());
```



True
98
98,875

Convert Sınıfı İle Tip Dönüşümü

- **System** isim alanının altındaki **Convert** sınıfı tür dönüşümü yapılabilen metotları içeren bir sınıftır.
- Bu metodlar ile hemen hemen her türü, her türe **CTS** karşılıklarını kullanarak dönüştürülür.
- Kullanımı: *[değişken] = Convert . [hedef tür] ([değişken]);*

```
bool a = true;  
Console.WriteLine(Convert.ToString(a));
```

```
int sayi = 98;  
Console.WriteLine(Convert.ToString(sayi));
```

```
float sayi2 = 98.875f;  
Console.WriteLine(Convert.ToString(sayi2));
```



True
98
98,875

Operatörler

- Programlama dillerinde tek başlarına herhangi bir anlamı olmayan ancak programın işleyişine katkıda bulunan karakter ya da karakter topluluklarına **operatör** denir. Örneğin $a+b$ ifadesinde $+$ işareti bir operatördür.
- Operatörlerin etki ettikleri sabit ya da değişkenlere ise **operand** denir. Örneğin $a+b$ ifadesinde a ve b birer operanddır.

Operatörler
()
! + - ++ --
* / %
+ -
< <= > >= is as
== !=
&
^
&&
?:
= *= /= %= += -= &=
^= !=

En Yüksek



En Düşük

Operatörlerde
İşlem önceliği

Operatörler (devam...)

- a) Aritmetik Operatörler
- b) Atama Operatörü
- c) Mantıksal Operatörler
- d) Karşılaştırma Operatörleri
- e) «as» ve «is» Operatörleri
- f) typeof operatörü
- g) Bitsel Operatörler

Operatörler (devam...)

Operatör	Açıklama
+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
%	Mod Alma

Operatör	Açıklama
<	Küçüktür
>	Büyüktür
<=	Küçük Eşittir
>=	Büyük Eşittir
==	Eşittir
!=	Eşit Değildir
-->	Uzaklaşma Operatörü

Operatör	Açıklama
&&	Ve
	Veya
!	Değil
^	Özel Veya
??	Null coalescing
?:	Koşul

Operatör	Açıklama
+=	Eşitliğin sağındaki ve solundaki değeri toplar ve solundaki değişkene atar
-=	Eşitliğin sağındaki ve solundaki değeri çıkarır ve solundaki değişkene atar
*=	Eşitliğin sağındaki ve solundaki değeri çarpar ve solundaki değişkene atar
/=	Eşitliğin sağındaki değeri solundaki değerine bölerek solundaki değişkene atar

for Döngüsü

- Bir veya birden fazla kod satırının belirtilen koşulları sağladığı sürece **artış** ve ya **azalış** değeri kadar tekrarlanmasını sağlayan döngüdür.
- Başlangıç ve bitiş değerleri olmak zorundadır.
- **Başlangıç değeri**, **koşul**, **artış veya azalış** kısımları farklı bir yerde tanımlanmak istenirse boş bırakılabilir. Fakat noktalı virgüller kesinlikle olmalıdır.
- Genel yazım biçimi şu şekildedir:

```
for ( [başlangıç değeri] ; [koşul] ; [artış veya azalış] )  
{  
    . . .  
    [işleme alınacak komutlar];  
    . . .  
}
```

foreach Döngüsü

- **foreach** döngüsü bir dizinin ve ya koleksiyonun **her elemanı için bulundurduğu komutları çalıştıran** döngüdür.
- Dizinin her elemanının değerini geçici bir değişkene atayarak bu değişken üzerinde işlemler yapmamızı sağlar.
- **foreach** döngüsü ile dizi veya koleksiyondaki elemanların değerleri değiştirilemez. Sadece «**read – only**» işlemler yapılabilir.
- Dizi veya koleksiyonların eleman sayıları bilinmediğinde kullanılır.
- Genel yazım biçimi şu şekildedir:

```
foreach ( [değişkenVeriTipi] [değişkenAdı] in [dizi/koleksiyon]
{
    . . .

    [işleme alınacak komutlar];

    . . .
}
```


foreach Döngüsü (devam...)

Örnek: Bir dizinin tüm elemanlarının ekrana yazdırılması.

```
int[] sayilar = { 1, 3, 5, 7 };  
foreach (var sayi in sayilar)  
{  
    Console.WriteLine("Okunan sayı:" + sayi.ToString());  
}
```

Çıktı:

```
Okunan sayı:1  
Okunan sayı:3  
Okunan sayı:5  
Okunan sayı:7
```

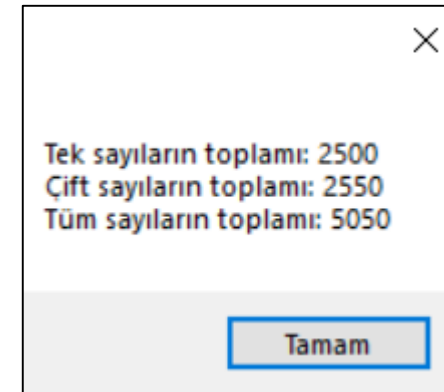
ÖRNEKLER

Örnekler

Örnek 1: 1-100 arasındaki Tek ve Çift sayıların Toplamı

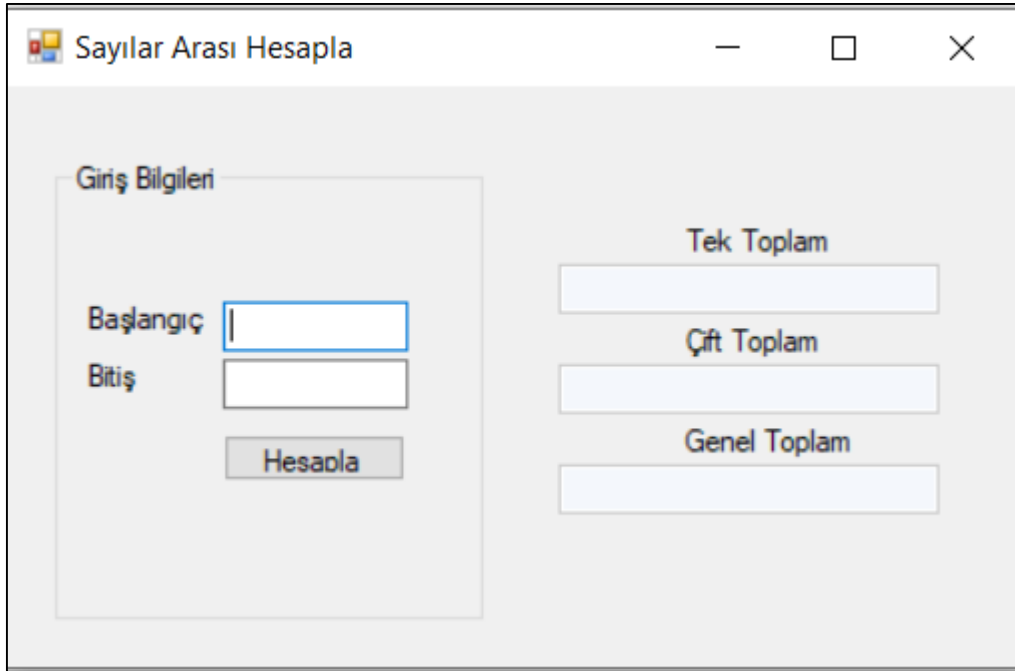
```
int tekToplam = 0, ciftToplam = 0;  
for (int i = 1; i <= 100; i++)  
{  
    if ((i % 2) == 0)  
        ciftToplam += i;  
    else  
        tekToplam += i;  
}
```

```
MessageBox.Show("Tek sayıların toplamı: " + tekToplam +  
    "\nÇift sayıların toplamı: " + ciftToplam +  
    "\nTüm sayıların toplamı: " + (tekToplam + ciftToplam));
```



Örnekler

Örnek 1: 1-100 arasındaki Tek ve Çift sayıların Toplamı,



The screenshot shows a Windows application window titled "Sayılar Arası Hesapla". Inside the window, there is a section titled "Giriş Bilgileri" (Input Information) which contains two text input fields labeled "Başlangıç" (Start) and "Bitiş" (End), and a "Hesapla" (Calculate) button. To the right of the input section, there are three text input fields for the results, labeled "Tek Toplam" (Odd Sum), "Çift Toplam" (Even Sum), and "Genel Toplam" (Total Sum).

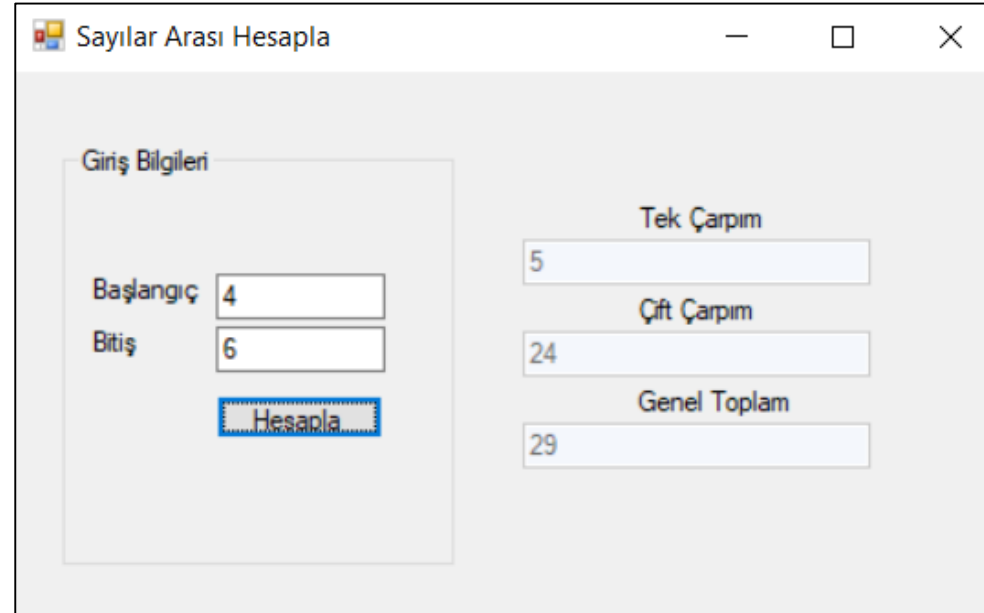
Yeni Versiyon

- Yandaki gibi bir grafik arayüze (GUI) sahip olmalı.
- Sadece sayısal giriş olmalı.
- Başlangıç bitiş değerinden küçük olmalı
- Hesaplama aşağıdaki prototipe sahip bir fonksiyonda yapılmalı.

`int` SonucDon(enToplamTur tur, `int` bas, `int` bit)

Örnekler

Örnek 2: 1-100 arasındaki Tek ve Çift sayıların Çarpımı (Diğer Formu Kopyalayarak Yapalım)



The screenshot shows a Windows application window titled "Sayılar Arası Hesapla". The window has a standard Windows title bar with minimize, maximize, and close buttons. The main content area is divided into two sections. On the left, under the heading "Giriş Bilgileri", there are two input fields: "Başlangıç" with the value 4 and "Bitiş" with the value 6. Below these fields is a button labeled "Hesapla". On the right, there are three output fields: "Tek Çarpım" with the value 5, "Çift Çarpım" with the value 24, and "Genel Toplam" with the value 29.

Giriş Bilgileri	Çıktılar
Başlangıç: 4	Tek Çarpım: 5
Bitiş: 6	Çift Çarpım: 24
Hesapla	Genel Toplam: 29

Yararlanılan Kaynaklar

- Sefer Algan , HER YÖNÜYLE C# , Pusula Yayıncılık, İstanbul, 2003
- Volkan Aktaş, HER YÖNÜYLE C# 5.0 , Kodlab Yayıncılık, İstanbul, 2013
- Milli Eğitim Bakanlığı "Nesne Tabanlı Programlama", 2012

İyi Çalışmalar...

Doç. Dr. Deniz Kılınç

deniz.kilinc@bakircay.edu.tr

drdenizkilinc@gmail.com

www.denizkilinc.com