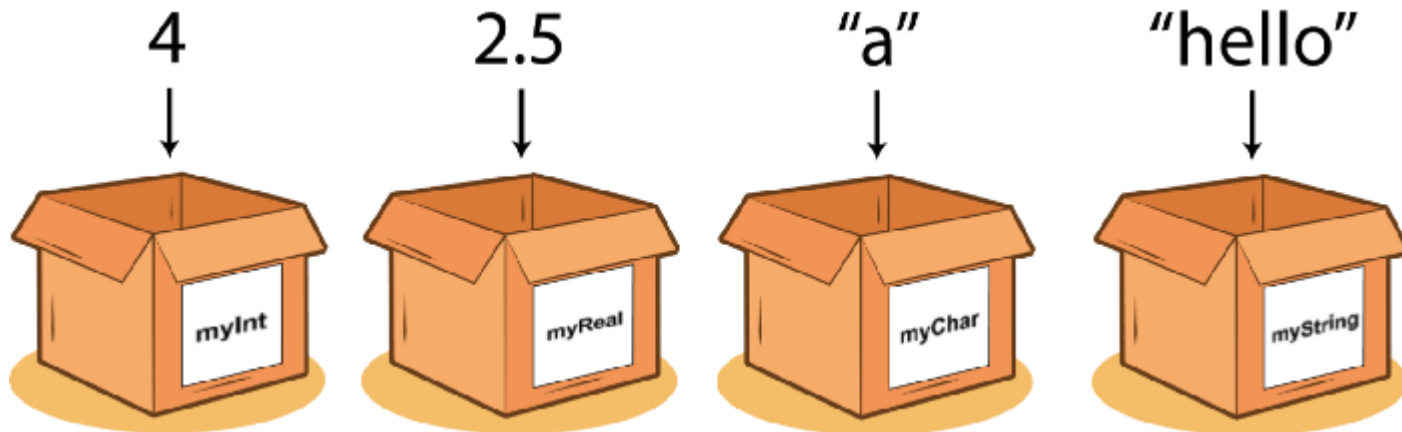


Değişkenler



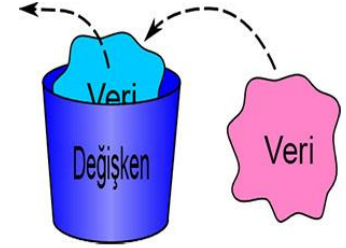
Tanımı:

- *Değişkenler, programların üzerinde işlem yaptığı verilerin saklandığı bellek alanlarıdır.*
- *Bu veriler tam sayı, ondalıklı sayı, metin gibi farklı türlerde olabilir.*
- *Bu yüzden programlama dillerinde de farklı türlerde değişkenler vardır.*

- *Değişken (variable) bir değeri tutan depolama konumudur.*
- *Değişkeni, bilgisayarınızın belleğinde geçici bilgi tutan bir kutu gibi düşünebilirsiniz.*
- *Programdaki her bir değişkene, kullanıldıkları ortamda değişkeni benzersiz olarak tanımlayan **tek ve benzersiz bir ad** vermek zorundasınız.*
- *Değişkenin değerine bu ad ile erişiriz.*
- *Değişkenleri aklımızdan bir sayı tutmaya benzetebiliriz.*
- *Bir şekilde program süresince (İlgili kod blokları arasında veya program açık kaldığı sürece) bize gerekli olacak değerleri hafızada tutmanın bir yoludur.*

- *C# metotlarda değişkenlerde her zaman bir sınıf içerisinde bulunurlar. Ve yaşam alanları küme parantezleri arasındaki kısımdı. Şimdi değişkenlerin tanımlanması ve yaşam alanlarına bakalım.*
- *Nerelerde Kullanılır;*
 - *Kullanıcıdan bir değer alındığı zaman,*
 - *Veri tabanından çekilen bir değer kullanıldığı veya kullanıcıya gösterileceği zaman*
 - *Programın çalışması ile hesaplama ya da oluşan değeri kullanıcıya göstermek/sunmak ya da kaydetmek için*

DİKKAT



- Her değişkenin bir **türü ve ismi** vardır.
- Tip o hafıza alanında saklanacak **verinin tip** bilgilerini ifade eder.
- İsim o **hafıza bölgesine** ulaşmamız için bir etikettir.
- Yalnızca harf (büyük ve küçük), rakam ve alt çizgi karakterini kullanabilirsiniz.
- Tanımlayıcı, bir harf ile (alt çizgi harf olarak kabul edilir) başlamak zorundadır

- *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.*

Değişken Tanımlama Kuralları:

- *C#'ta değişken tanımlanırken önce değişkenin türü sonra değişkenin adı yazılır.*
 - *(int i;)*
- *C# ta değişken isimleri büyük küçük harfe duyarlıdır.*
 - *(İnt a ile int A birbirinden farklıdır.)*
- *Değişkenler tanımlandıkları küme parantezleri içerisinde geçerlidir*
 - *{*
 - *int a*
 - *}*
 - *Burada a değişkenine ulaşamaz.*
 - *{*
 - *int a*
 - *}*

- *C# Programlama diline ait anahtar kelimelerden oluşamaz.*
 - *İnt for yanlış.(C#'ta bir döngü çeşitidir.)*
 - *TextBox yanlış.(C#'ta bir sınıf adıdır.)*
 - *İnt bool(Değişken)*
- *Değişken ismi maksimum 255 karakterli olmalıdır.*
- *Boşluk ve özel karakterler(?,!,],&,...) içeremez. Alt çizgi kullanılabilir.*
 - *Ad Soyad yanlış.*
 - *AdSoyad doğru.*
 - *ad&soyad yanlış.*
 - *ad_soyad doğru.*

- *Rakam ile başlayamaz, sadece rakamdan oluşmaz.*
 - *1ogrenci yanlış.*
 - *ogrenci1 doğru.*
- *Türkçe karakter (ö,ü,ç,ş,ğ,...) kullanılabilir fakat kullanımı tavsiye edilmez. Kullanılırsa hata oluşmaz.*
 - *Sınıf1 yerine Sinif1*
 - *yaş yerine yas*
- *Değişkene değer “=” operatörü ile yapılır.*
 - *int a=23;*

- Her komutun sonunda noktalı virgül olmak zorun da.
- Tek satırda aynı türden değişkenler tanımlanabilir
 - (int a,b,c gibi)
- Değişken isimlerini anlamlı yazmalıyız. Bir değişkene verilen isim ileriki zamanlarda tekrar bakıldığında ya da farklı bir yazılımcı tarafından bakıldığında anlaşılabilmelidir. Örneğin kullanıcının Adı ve Soyadını tuttuğumuz değişkenin adına as yerine adSoyad yazarak daha anlamlı hale getirebiliriz

Değişken Tanımlamada Yapılan Hatalar

- *Aynı satırda farklı türden değişkenler tanımlamaya çalışma*

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

- *Değişkene uygunsuz değer vermeye çalışma*

```
int a;
```

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

- *Değişkeni tanımlamadan ve/veya değişkene ilk değer vermeden değişkeni kullanmaya çalışma.*

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

```
using System;

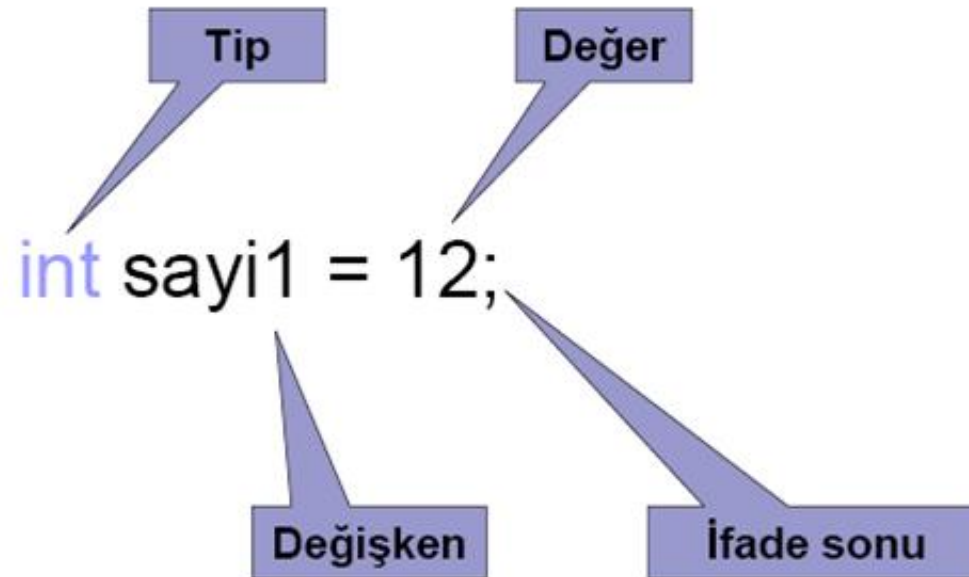
class degiskenler
{
    int a=5;

    static void Main()
    {
        Console.WriteLine(a) ; // farklı parantezlerin arasında olduğu için tanımlı değil
    }
}
```

Değişken Tanımlama

- *Değişkenlerin tanımlama kalıbı;*

[Veri Tipi] **Değişken Adı** = **Değer** (Metin ise “ ” içerisine) ;



Hafıza



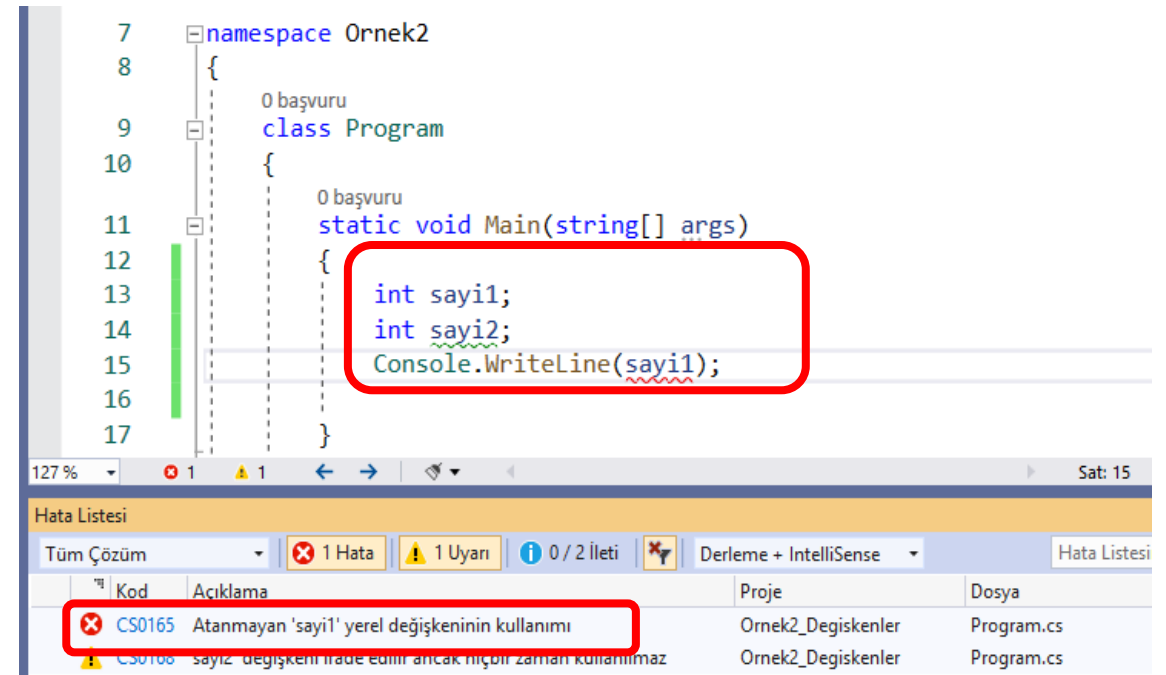
Tam sayı türünden iki adet değişken tanımlayalım;

```
int sayi1;
```

```
int sayi2;
```

- *Değişken isminin altında yeşil ile çizilde .netFramework ün hatırlatma yardımcı olma özelliği ile kullanmadığın bir değişkeni neden tanımlıyorsun diye bize hatırlatmada bulunuyor.*
- *Tanımlanan değişken eğer bir yerde kullanılacak ise başlangıç değeri atanmalıdır.*
- *Başlangıç değeri atamayan değişken derleme aşamasında hata verir. Bu koşul, Definite Assignment Rule (Kesin Atama Kuralı) olarak adlandırılır.*

- Mesela `sayi1` değişkenini ekrana yazdırmak istediğimizde.
- `Console.WriteLine(sayi1);` //değişken tanımlanmadan kullanılıncaya hata verir.
- Hata verir çünkü `colsole` sınıfının `WriteLine` metodunun parametre değeri parantez içinde yazılır ve bu değeri ekrana yazdırır. Burada tanımlı bir `sayi1` değişkeninde değer olmadığından hata verir. Değişkenin altını çizerek.
- Error; Use of unassigned local variable 'sayi1'



Hataları düzeltelim;

- *C# bir değişkene değer atanmadan kullanırsak hata verir. Değer atanmayan bir değişkeni kullanmaya kalktığımız için hata verdi.*
- *Sayı1 değişkenine değer atayalım;*

```
sayi1 = 23; //sayi 1 değişkenine değer atandı
```

- *Bir değişkene başka değişkendeki değeri atayıp eşitleyebiliriz.*

```
sayi2 = sayi1; //sayi 1 ile sayi 2 yi eşitliyoruz
```

- *Değişkene tanımlama sırasında da değer atanabilir;*

```
int sayi3=100;
```


- *Değişkene atanan değer artık kullanılabilir çalıştırıp görelim;*

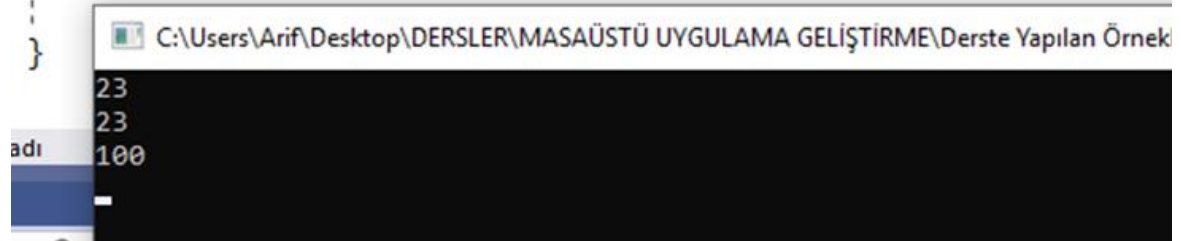
`Console.WriteLine(sayi1);`

`Console.WriteLine(sayi2);`

`Console.WriteLine(sayi3);`

`Console.ReadLine();` *yazalım ekranın beklemesi için.*

```
static void Main(string[] args)
{
    int sayi1;
    int sayi2;
    int sayi3 = 100; // tanımlanırken değer atanabilir.
    sayi1 = 23; // sayi 1 değişkenine değer atandı
    sayi2 = sayi1; // sayi 1 ile sayi 2 yi eşitliyoruz
    Console.WriteLine(sayi1);
    Console.WriteLine(sayi2);
    Console.WriteLine(sayi3);
    Console.ReadLine();
}
```



Değişkenlerin Faaliyet/Yaşam Alanı

- *C# dilinde programlar açılan ve kapanan parantezler içerisinde yazılır.*
- *C# dilinde bloklar süslü parantez “{ }” işaretleri ile gösterilir.*
- *Parantezlerin arasındaki bu bölgeye **blok** denir.*
- *Tanımlanan değişkene ancak bu blok içerisinde ulaşılabilir.*
- *Bu blok aralığına **Değişkenin faaliyet alanı** denir.*

- *Bir sınıfın üye elemanı olarak tanımlanmış değişken her zaman sınıfın faaliyet alanı içindedir.*
- *Yerel değişken tanımlandığı blok arasında kaldığı sürece faaliyet alanındadır.*
- *For, while do-while gibi döngü bloklarında tanımlanan değişkenler, faaliyet alanları döngünün içerisidir.*
- *Üst işlem bloğunda tanımlı olan değişkenler alt işlem bloğunda tanımlanıp kullanılabilirken, alt işlem bloğunda oluşturulan bir değişken üst işlem bloğunda(yani faaliyet alanı dışında-parantezlerin dışında) kullanılamaz tanımlı değildir.Bir örnek ile anlatalım*

Örnek yapalım; Değişken Faaliyet Alanı

```
static void Main(string[] args)
```

```
{//üst işlem bloğu(scope)
```

```
    int sayi1 = 12;
```

```
    {//alt işlem bloğu(scope)
```

```
        int sayi2 = 25;
```

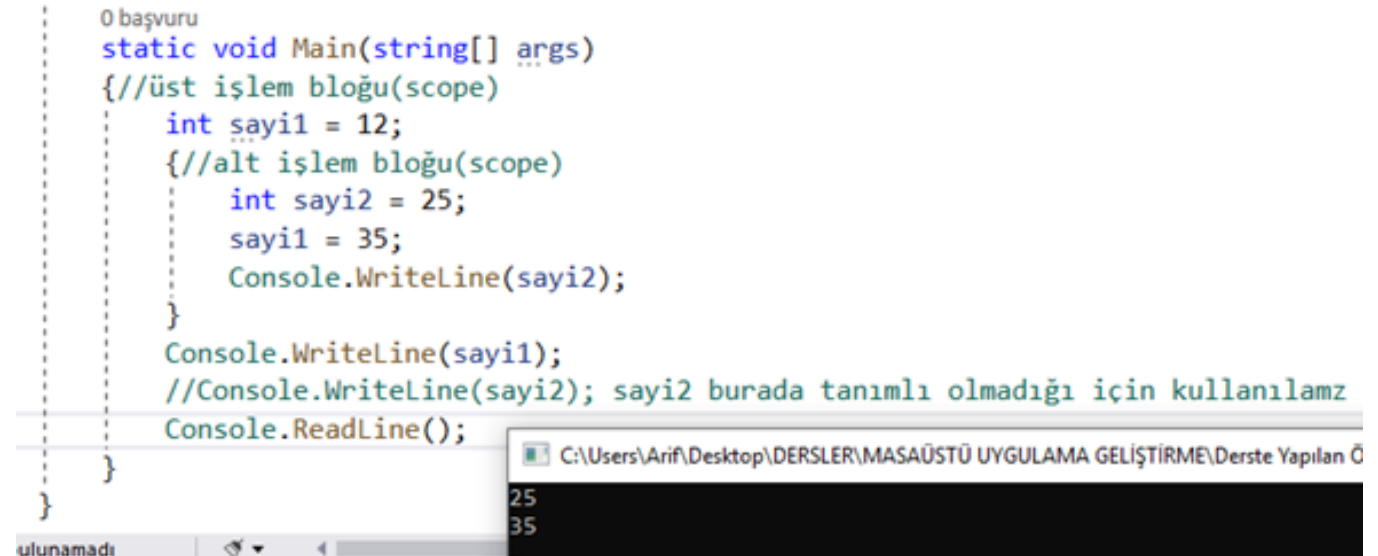
```
        sayi1 = 35;
```

```
        Console.WriteLine(sayi2);
```

```
    }
```

```
    Console.WriteLine(sayi1);//Console.WriteLine(sayi2); sayi2 burada tanımlı olmadığı için kullanılamaz
```

```
    Console.ReadLine();
```



```
0 başvuru
static void Main(string[] args)
{
    //üst işlem bloğu(scope)
    int sayi1 = 12;
    {
        //alt işlem bloğu(scope)
        int sayi2 = 25;
        sayi1 = 35;
        Console.WriteLine(sayi2);
    }
    Console.WriteLine(sayi1);
    //Console.WriteLine(sayi2); sayi2 burada tanımlı olmadığı için kullanılamaz
    Console.ReadLine();
}

C:\Users\Arif\Desktop\DESLER\MASAÜSTÜ UYGULAMA GELİŞTİRME\Derste Yapılan Ö
25
35
```

<pre>class Bloklar { static void Main() { int a = 3; int a = 5; Console.ReadKey(); } }</pre>	<p>Yandaki kod bloğunda aynı blok içerisinde a değişkeni <u>iki kez tanımlandığından</u> program derlemede hata verecektir.</p>
--	---

Derleyicide hata vermemesi için Main() ana kod bloğunun içerisine iki farklı kod bloğu eklenerek değişkenlerin tanımlandığı alanlar birbirinden ayrılır ve hatanın önüne geçilir.

<pre>class Bloklar { static void Main() { { int a = 3; } { int a = 5; } Console.ReadKey(); } }</pre>
--

Kaynaklar

- Öğr. Gör. Hakan Can ALTUNAY ÇARŞAMBA TİCARET BORSASI MESLEK YÜKSEKOKULU Nesne Tabanlı Programlama I Ders Notları
- <https://www.sahinsezgin.com/veri-tipleri-ve-degiskenler/>