

Veri Türleri / Data Types

*Veri Tiplerinin Ram'da Saklandığı Bölgeler
Sabitler*

- *Veri türleri, programımızda kullandığımız verileri saklamak için tutulacak verinin çeşidine göre bellek bölgesinde yer ayrılmasını sağlar.*
- *Mesela bellek bölgelerinin eşya taşıdığımız kutular gibi düşünecek olursak ince uzun boru şeklinde eşya ile sandık şeklindeki eşyanın kutu biçimleri nasıl farklı ise bellekte tutulacak olan bilgilerin türleri, uzunluğu, değişkenliğine ve kullanım yerine göre farklı tiplerde değerleri tutmamızı bu veri tipleri ile yaparız.*

- Uygun veri tiplerini seçmek taşınacak eşyaya uygun koli seçmek gibidir. Küçük kutu eşyayı taşımaz gereğinden büyük kutu seçerseniz gereğinden fazla yer kaplar kamyon ile taşıyacak olduğunuz eşyayı iki tır ile taşıyamazsınız.
- C# ' da 8 adet tam sayı, 3 adet ondalıklı sayı, 1 adet mantıksal ve 1 adet karakter değer tipi (value type), 2 tanesi referans tip (reference type) olmak üzere 15 tane önceden tanımlanmış veri tipi vardır.



Değer Tipli Değişken Türleri

- *Değer Tipli veri türleri belleğin **stack** denilen bölgesinde tutulur. Saklanması istenen değer doğrudan belleğe yazılır gerektiğinde çağrılır ve kullanılır.*
- *Veri türlerini niteleyen sıfatlar;*
 - *long = uzun*
 - *short = kısa*
 - *signed = işaretli*
 - *unsigned = işaretsiz*

- *Alfasayısal(Sayısal olmayan) Türler*
 - *Char Veri Tipi (Karakter)*
- *Sayısal Türler*
 - *Tam sayı*
 - *byte Veri Tipi*
 - *sbyte Veri Tipi*
 - *short Veri Tipi*
 - *ushort Veri Tipi(Pozitif Tamsayı)*
 - *int Veri Tipi(Tamsayı)*
 - *uint Veri Tipi (Pozitif Tamsayı)*
 - *long Veri Tipi(Tamsayı)*
 - *ulong Veri Tipi(Pozitif Tamsayı)*
 - *Ondalıkli sayı*
 - *Float Veri Tipi (Virgüllü-Gerçek-Ondalıkli sayı sayı)*
 - *Double Veri Tipi(Ondalıkli Sayı)*
 - *Decimal Tipi*
- *Mantıksal (Doğru/Yanlış – 0/1)*
 - *Bool Veri Tipi(Mantıksal)*

Char Veri Tipi (Karakter)

- Tek bir karakter tutar. 'x', 'z', '0', ...
- *Char veri türü **16 bit(2 bayt)** uzunluğunda Unicode standartlarında karakterlerin karşılıklarını tutan veri tipidir.*
- *Tek bir karakteri hafızada tutmanız gerektiğinde kullanabilirsiniz.*
- *Her bir karakterin Unicode standartları çerçevesinde bir karşılığı bulunmaktadır ve char tipinde değişken değer atandıktan sonra bir karakteri temsil etmektedir.*

Tür	Boyut	Açıklama	Örnek
char	2 bayt	Tek bir karakteri tutar.	<code>char a='h';</code>

Örnek Uygulama

```
char harf1 = 'C';
```

```
Console.WriteLine(harf1);
```

byte Veri Tipi

- *Byte türündeki değişkenler **8 bitlik** veri depolarlar.*
- *8 bitlik binary sistemdeki veriye byte da denilmektedir.*
- *Byte değişkeni 0 ile 255 arasında pozitif değer depolayabilmektedir*

Örnek Uygulama

```
byte birinci = 1;
byte ikinci = 123;

byte maxByteDegeri = byte.MaxValue;

byte minByteDegeri = byte.MinValue;

Console.WriteLine("Birinci degisken: " + birinci);
Console.WriteLine("Ikinci degisken: " + ikinci);
Console.WriteLine("Maksimum deger: " + maxByteDegeri);
Console.WriteLine("Minimum deger: " + minByteDegeri);
Console.ReadLine();
```

```
static void Main(string[] args)
{
    byte birinci = 1;
    byte ikinci = 123;
    byte maxByteDegeri = byte.MaxValue;
    byte minByteDegeri = byte.MinValue;

    Console.WriteLine("Birinci degisken: " + birinci);
    Console.WriteLine("Ikinci degisken: " + ikinci);
    Console.WriteLine("Maksimum deger: " + maxByteDegeri);
    Console.WriteLine("Minimum deger: " + minByteDegeri);
    Console.ReadLine();
}
```

 C:\Users\Arif\Desktop\DERSLEF

```
Birinci degisken: 1
Ikinci degisken: 123
Maksimum deger: 255
Minimum deger: 0
```


sbyte Veri Tipi

- *byte veri türüyle aynı özellikleri taşır.*
- *Sadece +- tanımlandığı için 128 ile 127 arasında değer depolayabilmektedir.*
- *Orijinal adı signed(işaretli) byte olarak adlandırılır.*
- *Değer aralığı:*

sbyte değişkeni -128 ile 127 arasında değer depolayabilmektedir.

Örnek Uygulama

```
sbyte birinci = -11;
sbyte ikinci = 26;

sbyte maxSbyteDegeri = sbyte.MaxValue;
sbyte minSbyteDegeri = sbyte.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);
Console.WriteLine("Ikinci degisken: " + ikinci);
Console.WriteLine("Maksimum deger: " + maxSbyteDegeri);
Console.WriteLine("Minimum deger: " + minSbyteDegeri);
```

```
/**sbyte***/
sbyte birinci = -11;
sbyte ikinci = 26;
sbyte maxSbyteDegeri = sbyte.MaxValue;
sbyte minSbyteDegeri = sbyte.MinValue;

Console.WriteLine("Birinci degisken: " + birinci);
Console.WriteLine("Ikinci degisken: " + ikinci);
Console.WriteLine("Maksimum deger: " + maxSbyteDegeri);
Console.WriteLine("Minimum deger: " + minSbyteDegeri);
```

C:\Users\Arif\Desktop\DERSLER\MASAÜSTÜ UYGULAMA GELİŞTİRME\Derste Yapılan Örnekler\MasaU:

```
Birinci degisken: -11
Ikinci degisken: 26
Maksimum deger: 127
Minimum deger: -128
```

short Veri Tipi

- *short türündeki değişkenler **16 bitlik (2 byte)** TAM SAYI veri depolarlar.*
- *Değer aralığı ;*
 - *short değişkeni -32768 ile 32767 arasında değer depolayabilmektedir.*
- *Değer aralığının dışına çıkma ihtimalinin olmadığı basit hesaplamalarda kullanılabilir.*

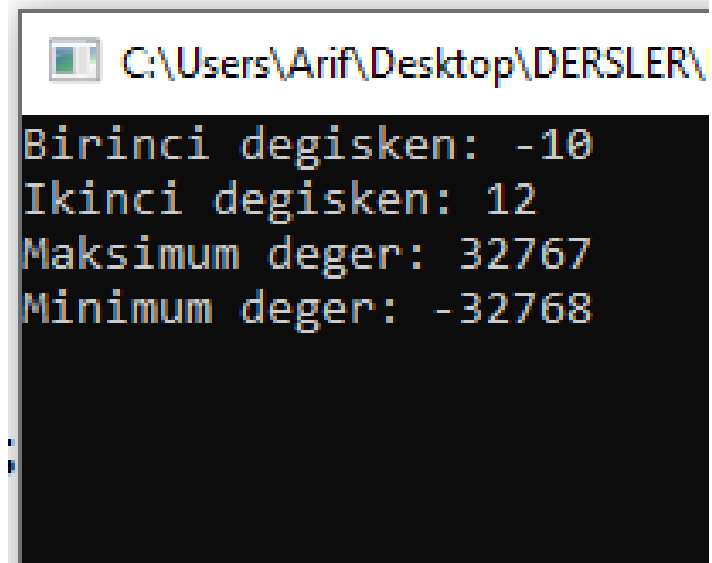
Örnek Uygulama

```
short birinci = -10;  
short ikinci = 12;  
short maxShortDegeri = short.MaxValue;  
short minShortDegeri = short.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxShortDegeri);  
Console.WriteLine("Minimum deger: " + minShortDegeri);
```

```
/******Short******/
```

```
short birinci = -10;  
short ikinci = 12;  
short maxShortDegeri = short.MaxValue;  
short minShortDegeri = short.MinValue;  
  
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxShortDegeri);  
Console.WriteLine("Minimum deger: " + minShortDegeri);
```



A screenshot of a Windows console window. The title bar shows the file path "C:\Users\Arif\Desktop\DESLER\". The console output displays four lines of text: "Birinci degisken: -10", "Ikinci degisken: 12", "Maksimum deger: 32767", and "Minimum deger: -32768". The text is rendered in a monospaced font with color coding: "Birinci" and "Ikinci" are red, "Maksimum" and "Minimum" are green, and the values are black.

```
C:\Users\Arif\Desktop\DESLER\  
Birinci degisken: -10  
Ikinci degisken: 12  
Maksimum deger: 32767  
Minimum deger: -32768
```

ushort Veri Tipi(Pozitif Tamsayı)

- *ushort türündeki değişkenler **16 bitlik (2 byte)** veri depolarlar.*
- *Short değişkeninden farkı **işaretsiz(sadece pozitif tam sayı)** değişken türü olmasıdır.*
- *U harfi un yani işaretsiz anlamı taşıyor.*
- *Değer aralığı:*
*ushort değişkeni **0 ile 65535** arasında pozitif değer depolayabilmektedir.*

Örnek Uygulama

```
ushort birinci = 111;
```

```
ushort ikinci = 35023;
```

```
ushort maxUshortDegeri = ushort.MaxValue;
```

```
ushort minUshortDegeri = ushort.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);
```

```
Console.WriteLine("Ikinci degisken: " + ikinci);
```

```
Console.WriteLine("Maksimum deger: " + maxUshortDegeri);
```

```
Console.WriteLine("Minimum deger: " + minUshortDegeri);
```

```
/******ushort Veri Tipi*****
```

```
ushort birinci = 111;
```

```
ushort ikinci = 35023;
```

```
ushort maxUshortDegeri = ushort.MaxValue;
```

```
ushort minUshortDegeri = ushort.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);
```

```
Console.WriteLine("Ikinci degisken: " + ikinci);
```

```
Console.WriteLine("Maksimum deger: " + maxUshortDegeri);
```

```
Console.WriteLine("Minimum deger: " + minUshortDegeri);
```

C:\Users\Arif\Desktop\DESLER\MA

```
Birinci degisken: 111
Ikinci degisken: 35023
Maksimum deger: 65535
Minimum deger: 0
```

int Veri Tipi(Tamsayı)

- *En çok kullanılan veri tipidir.*
- *Bunda en önemli etken tam sayı değer tutabilmesi ve veri aralığının geniş olmasıdır.*
- *int türündeki değişkenler **32 bitlik(4 byte)** işaretli veri depolarlar.*
- *Değer aralığı:*

İnt değişkeni -2.147.483.648 ile 2.147.483.647 arasında değer depolayabilmektedir.

- *int veri tipinin kullanım alanları; Hesaplamalarda, işlemlerde, tanımladığınız enum(sabit değerli) yapılarında, tam sayı değer kullanmanız gereken her yerde kullanabilirsiniz.*

Örnek Uygulama

```
int birinci = -123450;  
int ikinci = 123456789;  
int maxIntDegeri = int.MaxValue;  
int minIntDegeri = int.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxIntDegeri);  
Console.WriteLine("Minimum deger: " + minIntDegeri);
```

```
/** int */
```

```
int birinci = -123450;  
int ikinci = 123456789;  
int maxIntDegeri = int.MaxValue;  
int minIntDegeri = int.MinValue;  
  
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxIntDegeri);  
Console.WriteLine("Minimum deger: " + minIntDegeri);
```

 C:\Users\Arif\Desktop\DERSLER\MASAÜ

```
Birinci degisken: -123450  
Ikinci degisken: 123456789  
Maksimum deger: 2147483647  
Minimum deger: -2147483648  
-
```


uint Veri Tipi (Pozitif Tamsayı)

- *uint türündeki değişkenler 32 bitlik (4 byte) **işaretsiz tam sayı** veri depolarlar.*
- *İnt ile farkı işaretsiz olması sadece pozitif tam sayı tanımlanır.*
- *Değer aralığı :*

uint değişkeni 0 ile 4.294.967.295 arasında pozitif değer depolayabilmektedir.

- *uint veri tipinin kullanım alanları ; Hesaplamalarda, işlemlerde, tanımladığınız enum yapılarında,pozitif tam sayı değer kullanmanız gereken her yerde kullanabilirsiniz.*

Örnek Uygulama

```
uint birinci = 123;

uint ikinci = 123456789;

uint maxUIntDegeri = uint.MaxValue;

uint minUIntDegeri = uint.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);

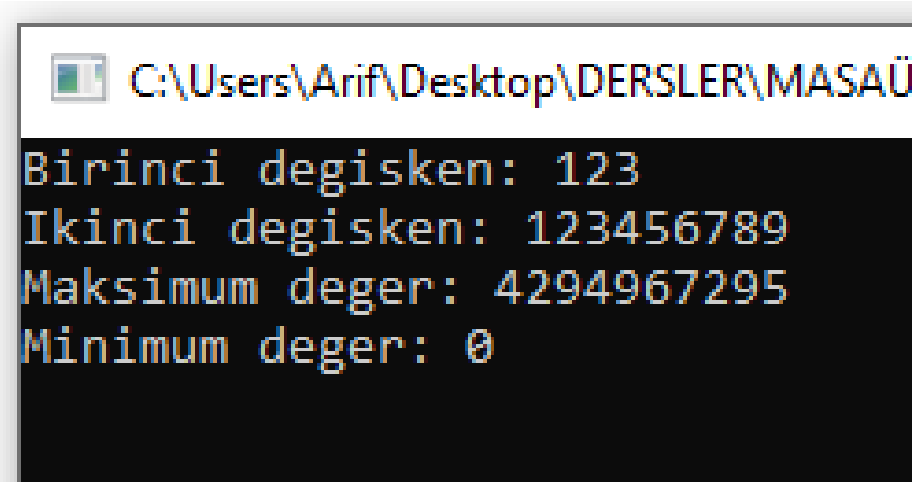
Console.WriteLine("Ikinci degisken: " + ikinci);

Console.WriteLine("Maksimum deger: " + maxUIntDegeri);

Console.WriteLine("Minimum deger: " + minUIntDegeri);
```

```
/****** uint *****/
uint birinci = 123;
uint ikinci = 123456789;
uint maxUIntDegeri = uint.MaxValue;
uint minUIntDegeri = uint.MinValue;

Console.WriteLine("Birinci degisken: " + birinci);
Console.WriteLine("Ikinci degisken: " + ikinci);
Console.WriteLine("Maksimum deger: " + maxUIntDegeri);
Console.WriteLine("Minimum deger: " + minUIntDegeri);
```



```
C:\Users\Arif\Desktop\DESLER\MASAÜ
Birinci degisken: 123
Ikinci degisken: 123456789
Maksimum deger: 4294967295
Minimum deger: 0
```

long Veri Tipi(Tamsayı)

- *long türündeki değişkenler **64 bitlik (8 byte)** işaretli veri depolarlar.*
- *Değer aralığı ne kadar? (Kuintrilyon (18))*
- *long değişkeni $-9,223,372,036,854,775,808$ to $9,223,372,036,854,775,807$ arasında değer depolayabilmektedir.*
- *long veri tipinin kullanım alanları : Hesaplamalarda, karmaşık işlemlerde kullanabilirsiniz.*

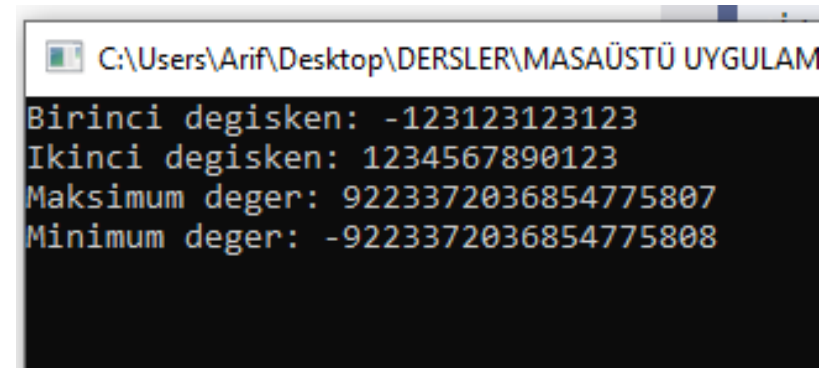
- *Örneğin iş yerinde çalışanların T.C Kimlik numaralarını 'integer' bir veri tipinde saklayamazsınız çünkü değer aralığı buna uygun değildir.*
- *Bu gibi durumlarda 'long' veri tipini rahatlıkla kullanabilirsiniz.*
- *Bununda karşılığı olarak integer veri tipine göre bellekte çok daha fazla yer kaplamaktadır (2 katı daha fazla yer kaplar).*
- *Bundan dolayı gerek duyulmadığı sürece 'long' veri tipini kullanmamalısınız.*

Örnek Uygulama

```
long birinci = -123123123123;  
long ikinci = 1234567890123;  
long maxLongDegeri = long.MaxValue;  
long minLongDegeri = long.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxLongDegeri);  
Console.WriteLine("Minimum deger: " + minLongDegeri);
```

```
/****** long *****/  
  
long birinci = -123123123123;  
long ikinci = 1234567890123;  
long maxLongDegeri = long.MaxValue;  
long minLongDegeri = long.MinValue;  
  
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxLongDegeri);  
Console.WriteLine("Minimum deger: " + minLongDegeri);  
|
```



C:\Users\Arif\Desktop\DESLER\MASAÜSTÜ UYGULAM

```
Birinci degisken: -123123123123  
Ikinci degisken: 1234567890123  
Maksimum deger: 9223372036854775807  
Minimum deger: -9223372036854775808
```

ulong Veri Tipi(Pozitif Tamsayı)

- *ulong türündeki değişkenler 64 bitlik (8 byte) işaretli veri depolarlar.*
- *İşaretsiz olduğu için sadece pozitif tam sayı.*
- *Değer aralığı;*
- *ulong değişkeni 0 ve 18,446,744,073,709,551,615 arasında değer depolayabilmektedir.*
- *ulong veri tipinin kullanım alanları;*
- *Hesaplamalarda, karmaşık işlemlerde kullanabilirsiniz.*

Örnek Uygulama

```
ulong birinci = 123;
```

```
ulong ikinci = 1234567890123;
```

```
ulong maxUlongDegeri = ulong.MaxValue;
```

```
ulong minUlongDegeri = ulong.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);
```

```
Console.WriteLine("Ikinci degisken: " + ikinci);
```

```
Console.WriteLine("Maksimum deger: " + maxUlongDegeri);
```

```
Console.WriteLine("Minimum deger: " + minUlongDegeri);
```

```
/****** ulong *****/
```

```
ulong birinci = 123;  
ulong ikinci = 1234567890123;  
ulong maxUlongDegeri = ulong.MaxValue;  
ulong minUlongDegeri = ulong.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxUlongDegeri);  
Console.WriteLine("Minimum deger: " + minUlongDegeri);
```

C:\Users\Arif\Desktop\DESLER\MASAÜSTÜ UYGULAM

```
Birinci degisken: 123  
Ikinci degisken: 1234567890123  
Maksimum deger: 18446744073709551615  
Minimum deger: 0
```

Float Veri Tipi (Virgüllü-Gerçek-Ondalıkli sayı sayı)

- Float türündeki değişkenler 32 bitlik (4 byte) gerçek sayı depolarlar.
- float değişkeni tanımladığımızda eğer içerisine tam sayı olmayan bir değer atamak istediğimizde sayısının sonuna F veya f koyarak türünün float olduğunu belirtmemiz gerekmektedir.
- Bunun sebebi C# ortamında noktalı sayıların **varsayılan değeri double'dır**.
- Biz virgüllü bir sayıyı float olarak tanımladığımızı sonuna gerekli eklemeyi yaptıktan sonra belirtiriz.

C# Tipi	.NET Tipi	Uzunluk (Byte)	Değer Aralığı	Duyarlı Basamak Sayısı
float	Single	4 byte	$1.5 * 10^{-45}$ - $3.4 * 10^{38}$	6 - 7 basamak


Not: e/E 10 ÜZERİ DEMEK 3.4E +/- 38 anlamı +/- 3.4*10³⁸ demektir.

Örnek Uygulama

```
float birinci = 123.04f;  
float ikinci = -1234123;  
float maxFloatDegeri = float.MaxValue;  
float minFloatDegeri = float.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxFloatDegeri);  
Console.WriteLine("Minimum deger: " + minFloatDegeri);
```

```
***** float *****  
  
float birinci = 123.04f;  
float ikinci = -1234123;  
float maxFloatDegeri = float.MaxValue;  
float minFloatDegeri = float.MinValue;  
  
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxFloatDegeri);  
Console.WriteLine("Minimum deger: " + minFloatDegeri);
```

 C:\Users\Arif\Desktop\DESLER\MASAÜS

```
Birinci degisken: 123,04  
Ikinci degisken: -1234123  
Maksimum deger: 3,402823E+38  
Minimum deger: -3,402823E+38
```

Double Veri Tipi(Ondalıkli Sayı)

- *double türündeki değişkenler 64 bitlik (8 byte) gerçek sayı depolarlar.*
- *double değişkeni $1.7E \pm 308$ (15 basamak) arasında değer depolayabilmektedir*
- *Double ondalıklı sayı (kayan noktalı sayı ifadesini de görebilirsiniz) grubunun **Default** veri tipidir.*
- *Yani C# üzerinde hiç bir tanım gözetmeden yazılan ondalıklı bir sayı double sınıfına aitmiş gibi davranır*

double	Double	8 byte	$5.0 * 10^{-324} - 1.7 * 10^{308}$	15 - 16 basamak
--------	--------	--------	------------------------------------	-----------------

Not: e/E 10 ÜZERİ DEMEK $3.4E \pm 38$ anlamı $\pm 3.4 * 10^{38}$ demektir.

Örnek Uygulama

```
double birinci = 123.04;  
double ikinci = -1233;  
double maxDoubleDegeri = double.MaxValue;  
double minDoubleDegeri = double.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxDoubleDegeri);  
Console.WriteLine("Minimum deger: " + minDoubleDegeri);
```

```
***** double *****/
```

```
double birinci = 123.04;  
double ikinci = -1233;  
double maxDoubleDegeri = double.MaxValue;  
double minDoubleDegeri = double.MinValue;  
  
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxDoubleDegeri);  
Console.WriteLine("Minimum deger: " + minDoubleDegeri);
```

 C:\Users\Arif\Desktop\DESLER\MASAÜSTÜ UYGUL

```
Birinci degisken: 123,04  
Ikinci degisken: -1233  
Maksimum deger: 1,79769313486232E+308  
Minimum deger: -1,79769313486232E+308
```

Decimal Tipi

- *Decimal türündeki değişkenler 128 bitlik (16 byte) gerçek sayı depolarlar.*
- *C# da decimal veri tipi ondalık sayılar için kullanılan veri tipleri içinde en fazla hassasiyete sahip olan veri tipidir.*
- *Daha çok, yüksek hassasiyete sahip muhasebe ve para işlerinde yada bilimsel araştırmalarda virgülden sonraki hassasiyetin çok önemli olduğu durumlarda kullanılan bir değişken türüdür.*
- *Bunun sebebi ise 'decimal' veri tipi **kesin sonuçlar** elde etmek istediğiniz kullanılır.*
- *Float ve Double veri tiplerinde ondalıklı sayılar ile işlem yaparken yazdığının veri tipinin son hanesi "5" den küçükse bunu ekrana yazdırmaz ama decimal'de bu durum söz konusu değildir.*
- *Veriyi net ve tam olarak hesaplar herhangi bir yuvarlama işlemi yapmaz. Bu nedenle parasal hesaplamalar yaparken her daim decimal kullanılır! Bellekte 16 Bayt yer kaplar ve 29 haneye kadar destekler.*

- *Aynı şekilde float da bahsettiğimiz gibi bu değişken türünde işlem yapacağımız zaman girilen(tam sayı olmadığı zaman) ondalıklı değer sonuna onun decimal türüne ait olduğunu bildiren m yada M işareti koyulur.*
- *Bunun sebebi C# ortamında noktalı sayıların varsayılan değeri double'dır. decimal değişkenine yaptığımız tam sayı atamalarında sonuna ekleme yapmamıza gerek yoktur*

Değer aralığı;

decimal değişkeni (-7.9×10^{28} to 7.9×10^{28}) / (100 to 28) (28 basamak) arasında değer depolayabilmektedir.

C# Tipi	.NET Tipi	Uzunluk (Byte)	Değer Aralığı	Duyarlı Basamak Sayısı
decimal	Decimal	12 byte	$\pm 1.0 \times 10^{-28}$ - $\pm 7.9 \times 10^{28}$	28 - 29 basamak

C# dilinde decimal sayı veri tipi

Double, Float, Decimal Arasında ki Temel Farklar:

- *Üç veri tipinin arasında ki en temel fark double ve float ile %100 kesin sonuçlar alamazsınız ondalıklı sayının son hanesine göre **yuvarlama** işlemi yapabilir fakat Decimal her zaman **kesin ve tam sonuçları** size gösterir.*

- Bunun yanı sıra 3 veri tipinin de **bellekte kapladığı alanlar** birbirinden çok farklı ve aynı şekilde alabileceği maksimum basamak sayısı da (kaç haneli) farklıdır.
- Yapacağınız ve tutacağınız veri tipine göre iyi bir seçim yapmanız gerekmektedir.
- Örneğin doğa olaylarını hesaplarken kesin sonuçlara ihtiyacınız yoktur ve zaten doğa olaylarında (deprem tahmini, hava durumu vs) kesin sonuç alamazsınız bundan dolayı Float veya Double kullanabilirsiniz. Ama Finansal veya kesin sonuca dayalı bir işlem olacaksa kesinlikle seçmeniz gereken veri tipi Decimal olmalıdır.

Örnek Uygulama

```
decimal birinci = 123.04m;
```

```
decimal ikinci = -12332;
```

```
decimal maxDecimalDegeri = decimal.MaxValue;
```

```
decimal minDecimalDegeri = decimal.MinValue;
```

```
Console.WriteLine("Birinci degisken: " + birinci);
```

```
Console.WriteLine("Ikinci degisken: " + ikinci);
```

```
Console.WriteLine("Maksimum deger: " + maxDecimalDegeri);
```

```
Console.WriteLine("Minimum deger: " + minDecimalDegeri);
```

```
/****** decimal *****/  
decimal birinci = 123.04m;  
decimal ikinci = -12332;  
decimal maxDecimalDegeri = decimal.MaxValue;  
decimal minDecimalDegeri = decimal.MinValue;  
  
Console.WriteLine("Birinci degisken: " + birinci);  
Console.WriteLine("Ikinci degisken: " + ikinci);  
Console.WriteLine("Maksimum deger: " + maxDecimalDegeri);  
Console.WriteLine("Minimum deger: " + minDecimalDegeri);
```

C:\Users\Arif\Desktop\DESLER\MASAÜSTÜ UYGULAMA GELİŞTİRME

```
Birinci degisken: 123,04  
Ikinci degisken: -12332  
Maksimum deger: 79228162514264337593543950335  
Minimum deger: -79228162514264337593543950335
```

Bool Veri Tipi(Mantıksal)

- *Mantıksal veri tipidir. **Doğru** ve **Yanlış** olmak üzere iki değer alan veri tipidir.*
- *Uygulama içerisinde true(doğru) ve false(yanlış) değer ataması yapılmaktadır.*
- ***Kontrol işlemlerinde** sıklıkla kullanılmaktadır.*
- *İlk değer karşılaştırılması sonucu mantıksal sonucun doğru ya da yanlış olduğunu kontrol edip iki sonuçtan birini verir.*

Örnek Uygulama

```
int sayi = 5;
```

```
bool sonuc = sayi > 10;
```

```
// sayi degiskenine atadigimiz deger 10dan büyük mu kontrol ediliyor.
```

```
//sayi 10dan küçük olduğu için yanlış (false) sonucu donmekte.
```

```
Console.WriteLine(sonuc);
```

```
/****** bool *****/
```

```
int sayi = 5;
```

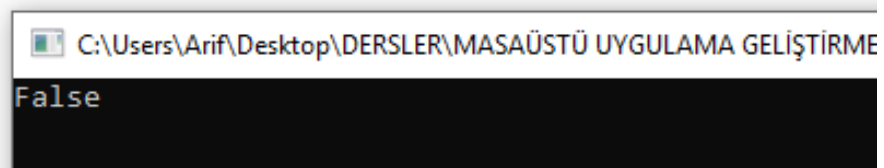
```
// sayi degiskenine atadigimiz deger 10dan büyük mu kontrol ediliyor.
```

```
bool sonuc = sayi > 10;
```

```
//sayi 10dan küçük olduğu için yanlış (false) sonucu donmekte.
```

```
Console.WriteLine(sonuc);
```

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



DateTime(Tarih ve Zaman)

- *Tarih ve zaman veri türündeki değişkenler 64 bitlik (8 byte) lık alan kaplar.*

Örnek Uygulama

- `DateTime zaman = DateTime.Now;`
- `Console.WriteLine(zaman);`

Değer Tipleri(Value Types)		
Veri Türü	Veri Tipi	Min. Max Değer Aralığı
Mantıksal	bool	false - true
Karakter	char	Tek karakter
Tam Sayılar	sbyte	- 127 ile 128 arası
	byte	0 ile 255 arası
	short	-32.768 ile 32.767 arası
	ushort	0 ile 65.535 arası
	int	-2.147.483.648 ile 2.147.483.647 arası
	uint	0 ile 4.294.967.295 arası
	long	-9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807 arası
	ulong	0 ile 18.446.744.073.709.551.615 arası
Ondalık Sayılar	float	$\pm 1.5 \times 10^{-45}$ ile $\pm 3.4 \times 10^{38}$ arası
	double	$\pm 5.0 \times 10^{-324}$ ile $\pm 1.7 \times 10^{308}$ arası
	decimal	$\pm 1.0 \times 10^{-28}$ ile $\pm 7.9 \times 10^{28}$ arası

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	$\pm 1.5 \cdot 10^{-45}$, ..., $\pm 3.4 \cdot 10^{38}$ (reel sayı)	float a=5F; veya float a=5f;
double	8 bayt	$\pm 5.0 \cdot 10^{-324}$, ..., $\pm 1.7 \cdot 10^{308}$ (reel sayı)	double a=5; veya double a=5d; veya double a=5D;
decimal	16 bayt	$\pm 1.5 \cdot 10^{-28}$, ..., $\pm 7.9 \cdot 10^{28}$ (reel sayı)	decimal a=5M; veya decimal a=5m;

Tür	Boyut	Açıklama	Örnek
char	2 bayt	Tek bir karakteri tutar.	char a='h';
string	Sınırsız	Metin tutar.	string a="Buraya Bir Metin Gelecektir.";

Referans Tipleri

Referans Tipler
(References
Types)

string

object

- *Referans kelime anlamı olarak temsil etmek anlamına gelmektedir.*
- ***Alacağı değer belli değildir(Bellekte sabit bir alan ayrılmaz)***
- ***Verinin kendisi yerine adresini tutarlar.***
- *Yani nesnenin kendisiyle değil, adresi üzerinden işlem yapılır. Tanımlandığında bellekte yer ayrılır.*
- *Bu sefer belleğin **Heap Bölgesi**'dir.*
- *Başka bir yerde kullanılmak istenildiğinde değer yerine adresi tutulur. Direk değer değiştirildiğinde buradan referans gösterilen tüm tiplerin değeri değişir.*
- *İki temel referans tipi vardır: **String ve object.***

string Referans Tipi

- *String veri türü Unicode karakterlerden oluşan bir dizi grubudur.*
- *Genellikle uygulama içerisinde metinsel değerleri bir string değişkeni içerisinde tutmaktayız.*
Sınırsız veri tutar.
- *Not: String değişkenin üzerinde sayı değeri tutabiliriz fakat bu sayı değerlerini matematiksel işlemlerde kullanamayız.*

Örnek Uygulama

```
string universite = "Bilecik Şeyh Edebali Üniveristesi";  
string okul = "Pazaryeri MYO";  
Console.WriteLine(universite);  
Console.WriteLine(okul);  
Console.WriteLine(universite + " " + okul);
```

```
/*STRING*/  
string universite = "Bilecik Şeyh Edebali Üniveristesi";  
string okul = "Pazaryeri MYO";  
Console.WriteLine(universite);  
Console.WriteLine(okul);  
Console.WriteLine(universite + " " + okul);
```

C:\Users\Arif\Desktop\DESLER\MASAÜSTÜ UYGULAMA GELİŞTİRME\Derste Yapılan Örnekler\MasaUstuUygu

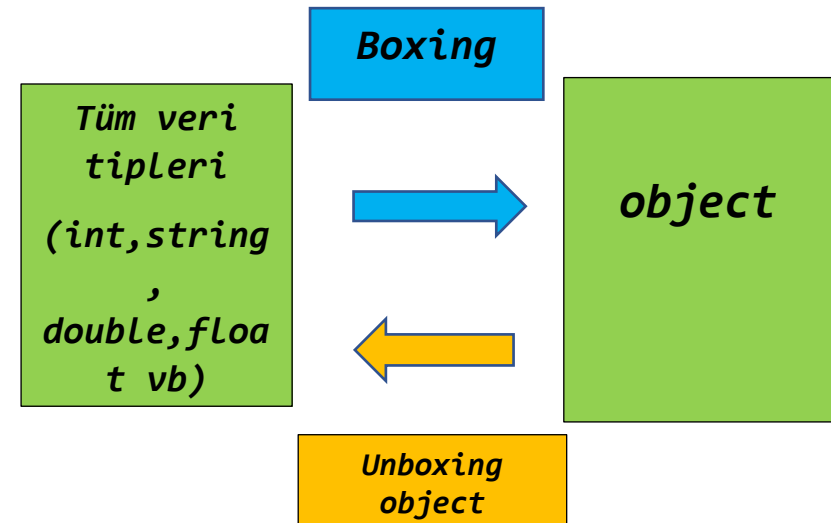
```
Bilecik Şeyh Edebali Üniveristesi  
Pazaryeri MYO  
Bilecik Şeyh Edebali Üniveristesi Pazaryeri MYO
```

Object Referans Tipi ve Boxing(Kutulama)

- *C#' da bütün veri türleri object veri türünden türemiştir.*
- *Bütün veri tiplerinin atası olduğu için **tüm veri tipleri object nesnesini tanır.***
- *Bu yüzden herhangi bir nesne rahatlıkla object türüne aktarılabilir.*
- *Yani tüm veri tipleri Object veri tipine dönüştürülebilir bu object halinde başka bir noktaya alıp hiç değer kaybı yaşamadan eski haline alabiliriz.*
- *Matematiksel işlemler yapılamaz.*

Boxing(kutulama)

- *Herhangi bir veri tipini object veri tipine atamaya Boxing(kutulama) denir.*
- *Tersi işleme de unboxing denir.*



Örnek ile bakalım;

- İlk olarak *int, double, string* türünden 3 değişken tanımlayalım;

```
int sayi1 = 34;  
double sayi2 = 43324;  
string selam = "merhaba";
```

- *Object* veri tipi oluşturalım ve *sayi1* değişkeninin bu nesneye atalım. Bu işleme **boxing** denir.

```
object ata = sayi1; //Boxing türünde değişken tanımladık ve değer atadık  
Console.WriteLine(ata);  
ata = sayi2; //Boxing  
Console.WriteLine(ata);  
ata = selam; //Boxing  
Console.WriteLine(ata);
```

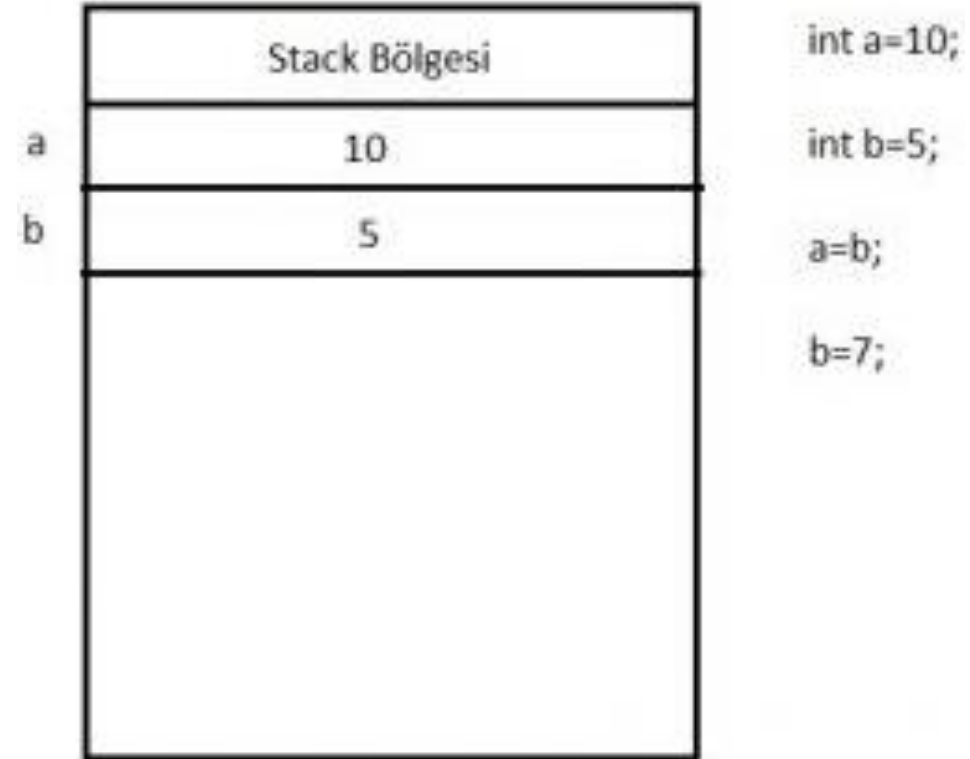
- *Unboxing* tekrar eski haline getirme;

```
ata = sayi1;  
sayi1 = (int)ata; // Unboxing
```

Veri Tiplerinin Ram'da Saklandığı Bölgeler

Stack Bölgesi:

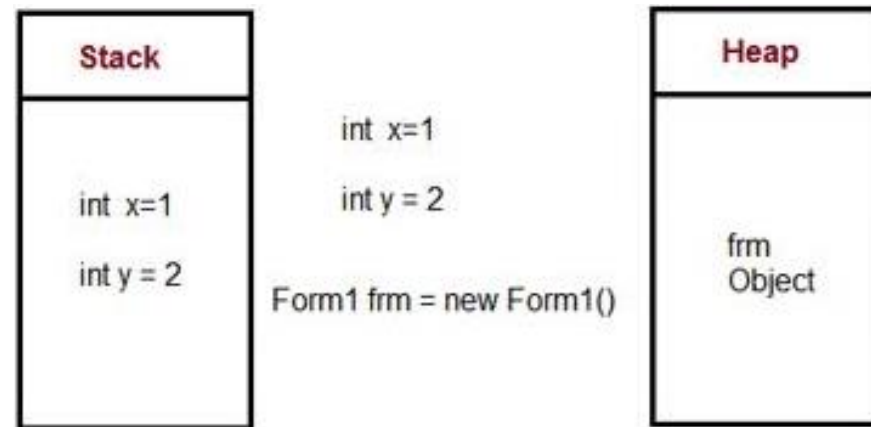
- *Stack bölgesi, RAM'in stack olarak adlandırdığımız bölgeleridir.*
- *Stack bölgesinde, **değer tipi değişkenler tutulur.***
- *Yani int, bool, double ve diğerleri. Ayrıca, stack alanında referans yoluyla belirtilen nesnelerin adres bilgileri tutulur.*

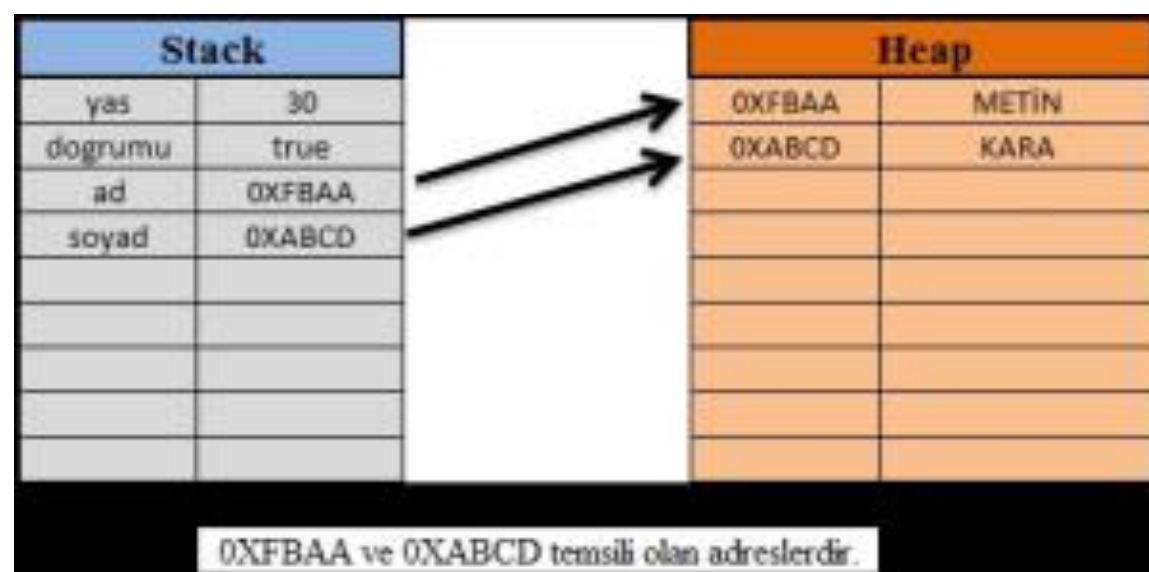
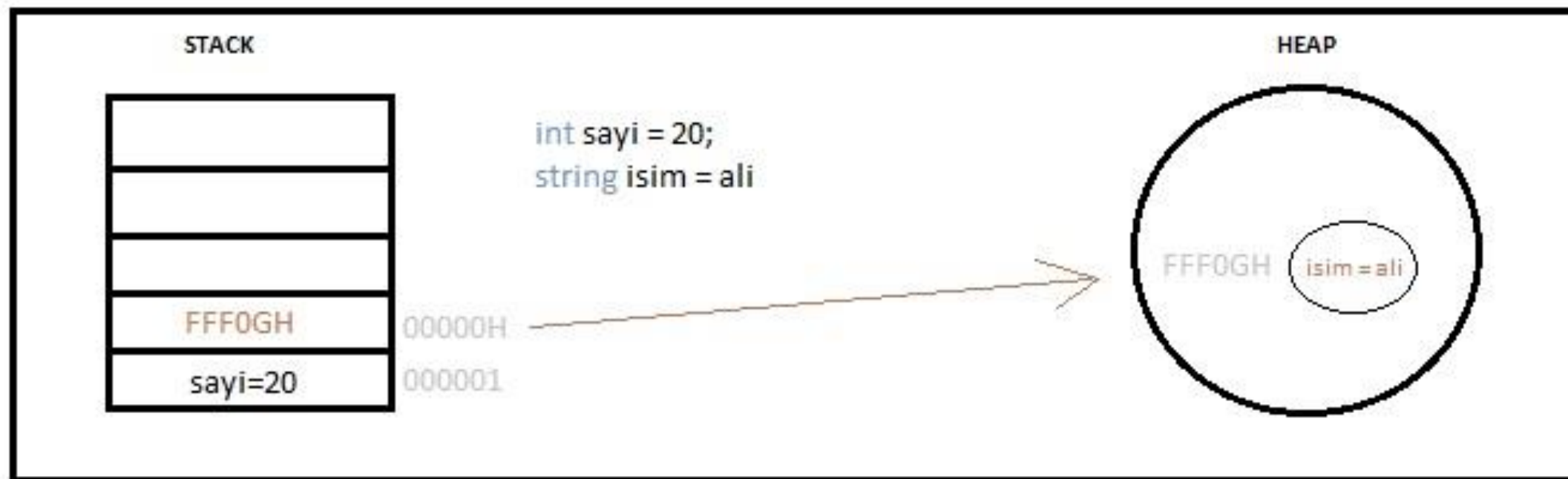


- *Değer tipi (Value Types) tipinden bir değişken tanımladığımızda, daha derleme aşamasında stack bellekten int tipinin bellekte tuttuğu yer kadar (4 byte) yer tahsis edilir ve değişkene değer atanmışsa ilk atanan değer yerleştirilir.*
- *Sonra bu değer değişse bile, bu değişkene 4 byte'lık (2^{32} bit)'lik bir sayı yerleştirebileceğimiz için çok büyük sayıları bile atayabileceğimiz bir yer tahsis etmiş oluruz.*

Heap Bölgesi:

- *Heap bölgesi, RAM'in heap olarak adlandırdığımız bölgeleridir.*
- *Heap bölgesi, **C# nesneleri** tarafından kullanılır.*
- *Nesneler burada oluşturulur.*
- *Heap bölgesinden bir nesneye yer ayırmak için "**new**" kelimesi kullanılır.*
- *Heap bölgesindeki veriye ulaşmak için Stack bellekteki adresi üzerinden ulaşılır.*





Farkları nedir?

- Stack bellekten statik olarak yer tahsisi için kullanılırken, Heap dinamik olarak yer tahsisi içindir.

<u>STACK</u>	<u>HEAP</u>
<i>Statik(Boyut sabit)</i>	<i>Dinamik(Boyut değişken)</i>
<i>Erişimi kolay ve hızlıdır</i>	<i>Erişimi zor ve yavaştır</i>
<i>Derlenme aşamasında belleğe yerleşir</i>	<i>Çalışma zamanında belleğe yerleşir.</i>
<i>Stack bölgesi küçük ve sınırlıdır</i>	<i>Bellek bölgesi büyüktür</i>
<i>Hemen silinir</i>	<i>Hemen silinmez(Belli mekanizmalar ile silinir)</i>
<i>Değer Tipli veri tutar(int, char, double, float vb.)</i>	<i>Referans Tipli verileri tutar(string, object, referans vb.)</i>

Sabitler

- *Değişmeyen değişken.(Umut)*
- *Program boyunca sabit kalacak veriler için kullanılan tanımlamalardır.*
- *Bir sabit tanımlamak için **const** anahtar kelimesini kullanırız. Değişken program içerisinde farklı değerler alabilirken **sabit program süresince tanımlandığı değeri korur.***
- *Sabitler tanımlanırken ilk değer ataması yapılmak zorundadır. Atanan bu değer sabit(2,300 gibi) bir değer olmalıdır.*
- *Program boyunca sabit değeri değiştirilemez. Değiştirilmeye kalkılırsa hata verir.*

Kullanımı:

```
const double pi = 3.14159265;
```

- *Pi sayısı, yer çekimi kuvveti veya ışık hızı gibi değerler değişmeyeceğinden, bir uygulama içerisinde çalıştığı sürece bir sabit ile ifade edilebilir.*
- *Bunu, bir dairenin çevresini ve alanını bulmamızı sağlayan bir uygulama yazarak örneklendirelim*

```
// Pi sayısını sabit olarak tanımlayalım
    const double pi = 3.14159;

    // Değişkenleri tanımlayalım
    double alan, cevre, yaricap;
    string yaricapGir;

    Console.WriteLine("Dairenin yarıçapını girin: ");
    yaricapGir = Console.ReadLine();
    yaricap = Convert.ToDouble(yaricapGir);
```

Dairenin çevresini $\Ç=2\pi r$ ve alanını $A=\pi r^2$ hesaplayalım. Pow üs almaya yarar.

```
    cevre = 2 * pi * yaricap;
    alan = pi * Math.Pow(yaricap, 2);
    Console.WriteLine("Dairenin alanı= {0}", alan);
    Console.WriteLine("Dairenin çevresi= {0}", cevre);
    Console.ReadLine();
```

Özel Karakterler

```
static void Main(string[] args)
{
    Console.WriteLine("Merhaba\tDünya\n\n");
    Console.ReadLine();
}
```

```
static void Main(string[] args)
{
    Console.WriteLine("Merhaba \tDünya\nNasılsın");
    Console.ReadLine();
}
```

C:\Users\Arif\Desktop\DERSLER\MASAÜSTÜ UYGULAMA GELİŞTİRME\Derste Yapılan

Merhaba Dünya
Nasılsın

\b	backspace karakteri
\n	yeni satıra geçmek için kullanılır
\t	tab tuşu kadar boşluk bırakır

Kaynaklar

- <https://www.sahinsezgin.com/veri-tipleri-ve-degiskener/>

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