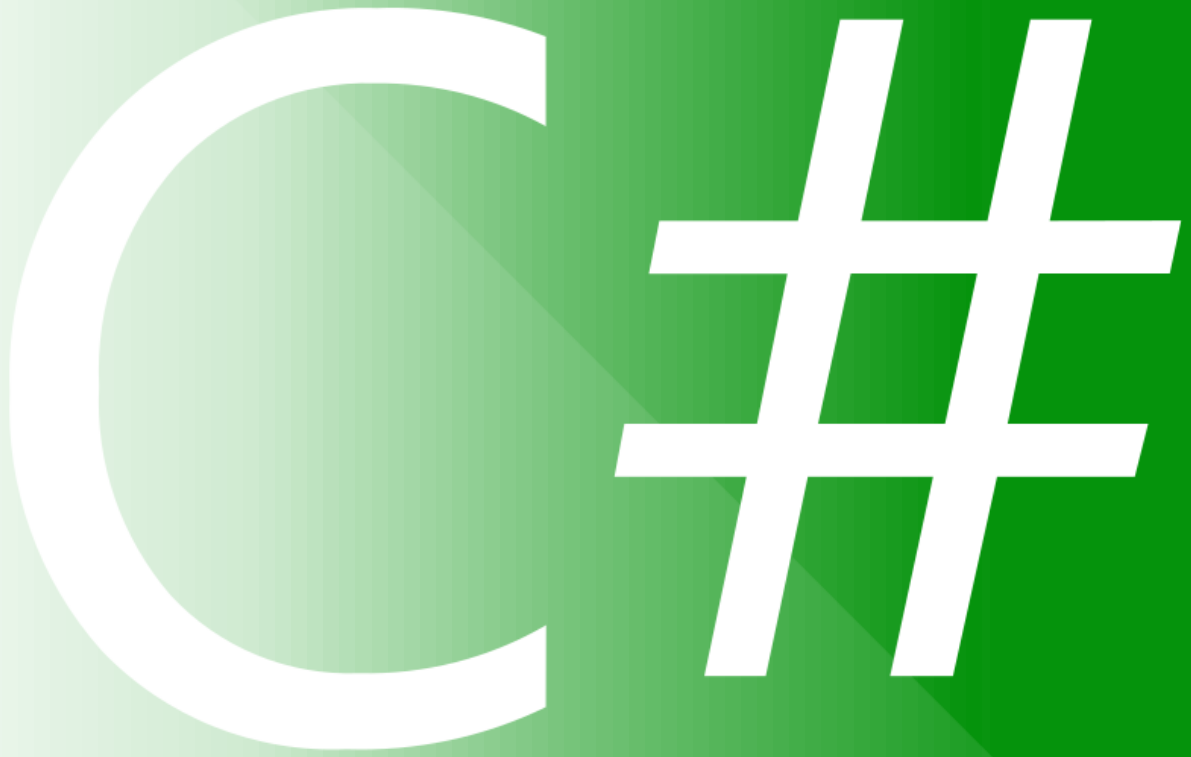
ISE 102- Nesneye Dayalı Programlama

Hafta 2 –

C# Temelleri

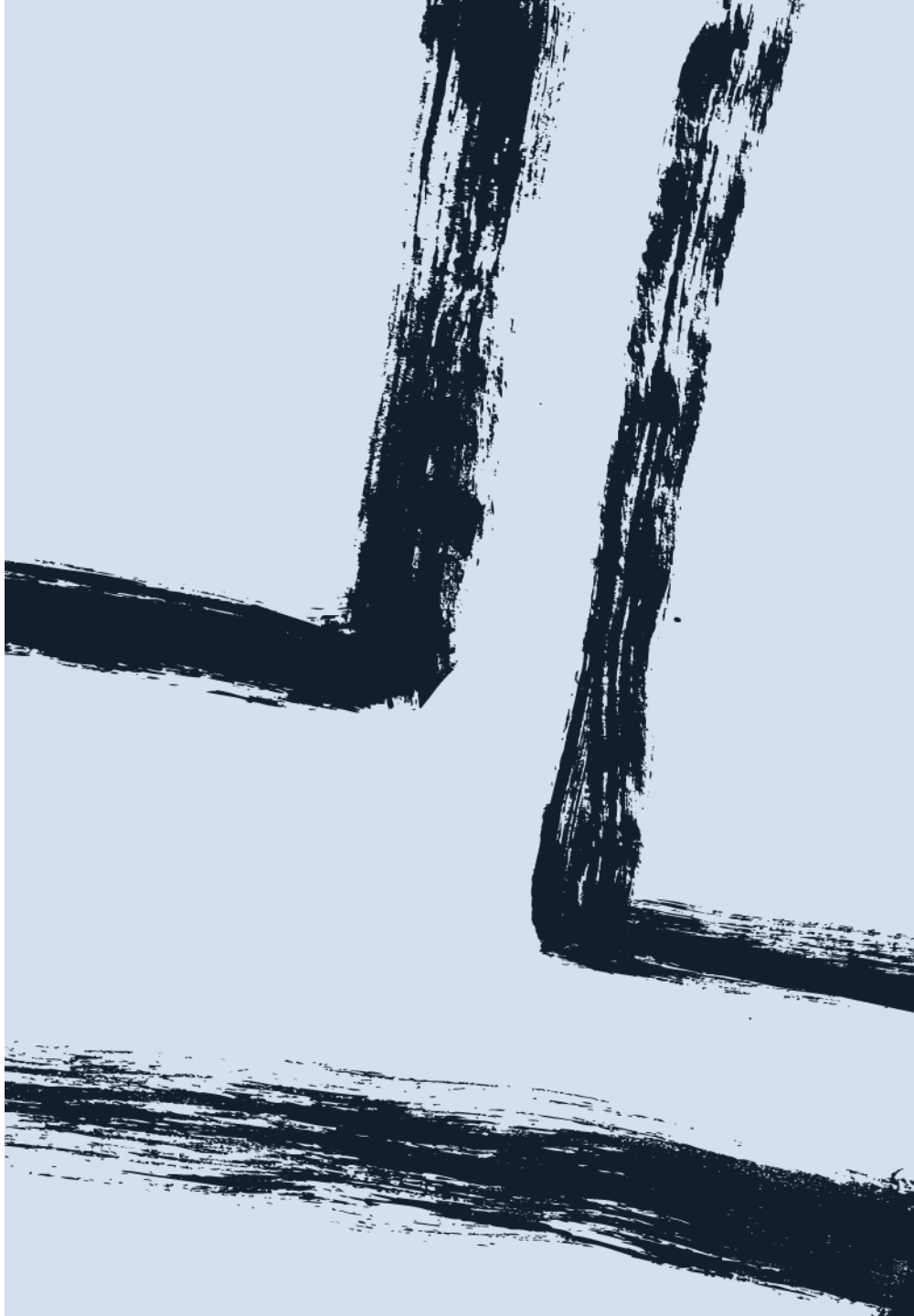


Arş. Gör. Dr. Muhammed KOTAN

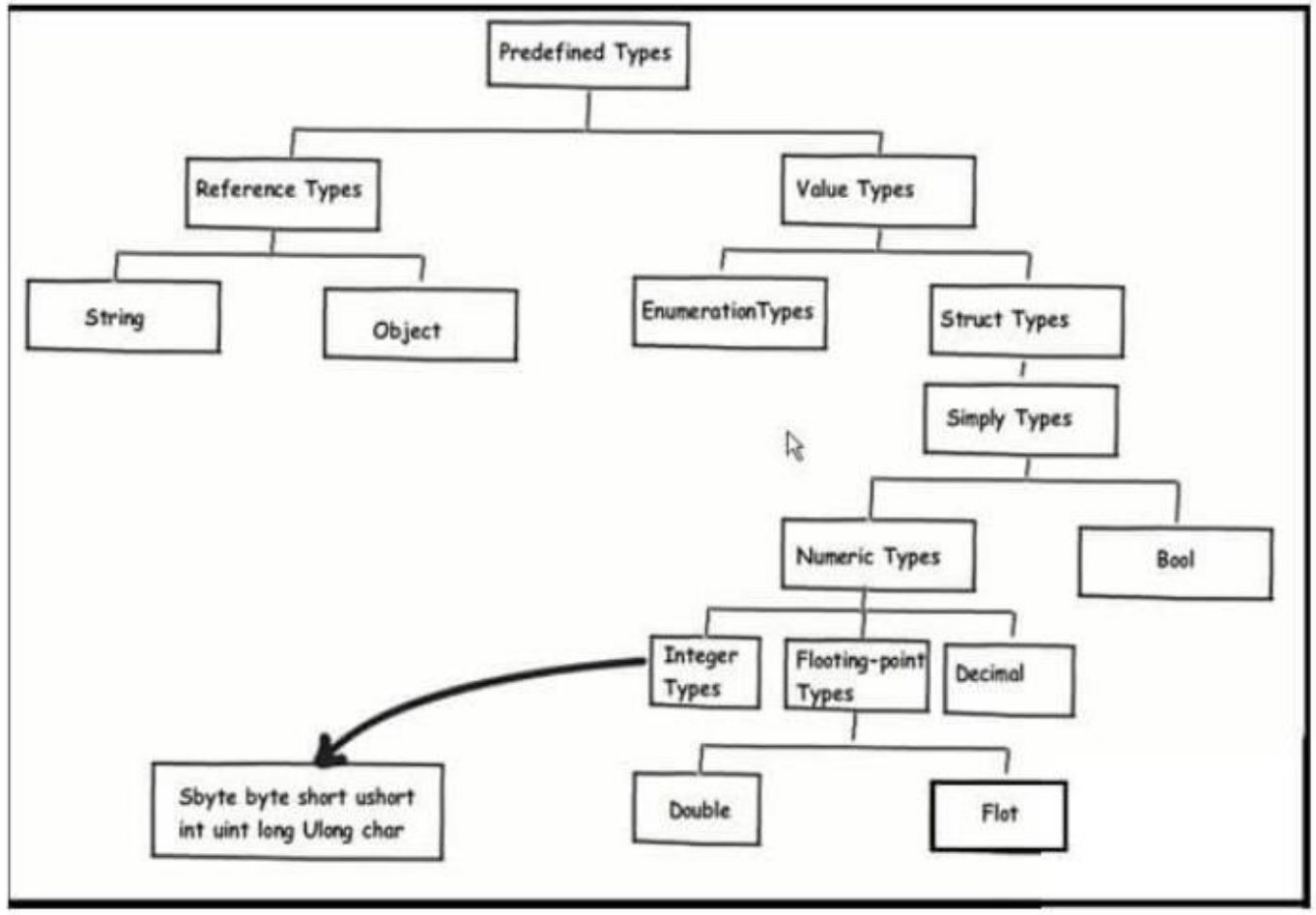
Temel Veri Türleri

C# dilinde temel olarak veri tipleri ikiye ayrılır.

- Önceden tanımlanmış veri türleri
  - Değer tipi
  - Referans tipi
- Kullanıcı tarafından tanımlanmış veri türleri



Temel Veri Türleri



DEĞİŞKENLER

Değişkenler bir bilginin bellekteki konumunu temsil eden sembolik isimlerdir.

- Program çalıştırıldığında değişkene ve bu değişkenin türüne göre bellekte yer ayrılır.
- Programlar belleğe yüklendikten sonra mikroişlemciler vasıtası ile çalıştırılırlar.
- Veri türleri hangi tür hafızalarda ve bellek bölgelerinde tutulurlar?

DEĞİŞKENLERİN ADLANDIRILMASI

1. Değişken isimleri mutlaka bir harf veya (_) ile başlamalıdır.
2. Rakam ve alt çizgi (_) karakterleri de değişken isminde kullanılabilir.
3. C# komutları ve fonksiyonları değişken adı olarak kullanılamaz.
4. Değişken adı arasında boşluk bulundurmamalıdır.
5. C#'ta değişken adlandırmada küçük-büyük harf ayırımı vardır. **AD**, **ad** ve **Ad** farklı değişkenleri ifade etmektedir.
6. C#'ta Türkçe karakterler isimlendirmede kullanılabilir, ancak tavsiye edilmez.

Geçerli değişken isimleri : baslamaZamanı, ad_soyad, x5

Geçersiz değişken isimleri : 3x, while, data?in, data in

DEĞİŞKEN TANIMLAMA

- ***Genel olarak;***
- <veri türü> ismi;

- ***Tip adı ile değişken tanımlama***

```
int sayi1;
int sayi2 = 10;
```

- ***var anahtar kelimesi ile değişken tanımlama***

```
var sayi1=10;  //int
var sayi2 = 10.0; //double
var sayi3 = 10.0f; //float
```

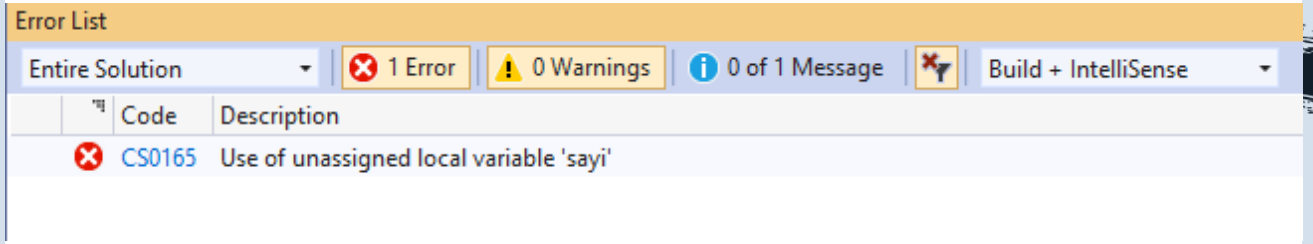
```
var sayi4; // Hata derleyici tipi bilemez;
```

CS0818 Implicitly-typed variables must be initialized

- C# dilinde bir değişkene değer atamadan önce kullanılması yasaktır.
- Derleme işlemi gerçekleşmez.

- Örn:

```
int sayi;
Console.WriteLine(sayi);
```



SABİT TANIMLAMA

- Program boyunca değerinin değişmeyeceğini düşündüğümüz veriler sabit veriler olarak tanımlanır.
- const anahtar sözcüğü kullanılır.

```
const int yukseklik = 10;
const string dil = "Visual C#";
const double pi = 3.14;
const int genislik; // HATA: değer belirtilmemiş
```

CS0145 A const field requires a value to be provided

- ***Değiştirmeye çalışıp ne hata verdiğini inceleyelim!***

SABİT TANIMLAMA

```
int a = 3, b;
b = a;
const int c = a + b;  // Hata
```

CS0133 The expression being assigned to 'c' must be constant

```
const int a = 3;
const int b = a + 3; //Geçerli
```

3 önemli kural:

- Sabitler tanımlandıklarında ilk değer atanmalıdır.
- Sabit ifadelere ancak sabit ifadelerle ilk değer atanabilir.
- İçsel tasarım olarak zaten static oldukları için ayrıca static olarak belirtmek hatalıdır ve kullanılamaz.

VERİ TİPLERİ

STANDART VERİ TİPLERİ

	Veri Tipi	Byte	Aralık
Tamsayı	byte	1	0 ... 255
	short	2	-32,768 ... 32,767
	int	4	-2,147,483,648 ... 2,147,483,647
	long	8	-9,223,372,036,854,775,808 ... 9,223,372,036,854,775,807
Reel Sayı	float	4	-3.4028235E38 ... 3.4028235E38
	double	8	-1.79769313486231E308 ... 1.79769313486231E308
	decimal	16	$-79,228 \times 10^{24}$ ... $79,228 \times 10^{24}$
Karakter	char	2	0 ... 65,535
	string	2	0 ... 2 milyar karakter
Lojik	bool	2	True veya False (False durumunda 0 değeri döndürülür)
Tarih/zaman	DateTime	8	Tarih ve zaman
	TimeSpan	8	Tarih aritmetik işlemlerinde (mesela 2 tarih arasındaki farkı bulmak için) kullanılır.
Genel	object	4	Harhangi bir tip. C# nesnelerini (buton veya form gibi) tanımlamak için kullanılı.

VERİ TİPLERİ

Short Name	.NET Class	Type	Width	Range (bits)
byte	Byte	Unsigned integer	8	0 to 255
sbyte	SByte	Signed integer	8	-128 to 127
int	Int32	Signed integer	32	-2,147,483,648 to 2,147,483,647
uint	UInt32	Unsigned integer	32	0 to 4294967295
short	Int16	Signed integer	16	-32,768 to 32,767
ushort	UInt16	Unsigned integer	16	0 to 65535
long	Int64	Signed integer	64	-9223372036854775808 to 9223372036854775807
ulong	UInt64	Unsigned integer	64	0 to 18446744073709551615
float	Single	Single-precision floating point type	32	-3.402823e38 to 3.402823e38
double	Double	Double-precision floating point type	64	-1.79769313486232e308 to 1.79769313486232e308
char	Char	A single Unicode character	16	Unicode symbols used in text
bool	Boolean	Logical Boolean type	8	True or false
object	Object	Base type of all other types		
string	String	A sequence of characters		
decimal	Decimal	Precise fractional or integral type that can represent decimal numbers with 29 significant digits	128	±1.0 × 10e−28 to ±7.9 × 10e28

byte VERİ TİPİ

```
byte a = 1;
byte b = 2;
byte c = a + b; // Derleme hatası
```

CS0266 Cannot implicitly convert type 'int' to 'byte'. An explicit conversion exists (are you missing a cast?)

Yukarıdaki kod derleme hatası verecektir çünkü byte tipi sayı olarak kabul edilmez, 8 bitlik grup olarak kabul edilir. Bu yüzden de toplama işlemi yapılmadan önce otomatik olarak int tipine dönüştürülür. Bu sebeple de a+b işleminin sonucu int olur. Sonucu byte tipindeki bir değişkene atayabilmek için tip dönüşümü yapılmalıdır.

```
byte a = 1;
byte b = 2;
byte c =(byte) (a + b);
```

Referans Veri Tipleri

- String ve object veri türü önceden tanımlanmış temel referans tipleridir.
- Object türü tüm türlerin türetildiği bir sınıf yapısıdır.
- C# dilinde bütün nesneler birer object tir.
- Boxing-Unboxing kavramına değineceğiz.

```
string s1 = "Merhaba";
string s2 = ".NET";
string s3 = s1 + s2;
string s4 = "Escape karakterleri \' , \\"
string s5 = @"Escape karakterleri ',\";
```

```
Object a;
a = 5;
Console.WriteLine(a.GetType());
a = 'A';
Console.WriteLine(a.GetType());
a = 12.5F;
Console.WriteLine(a.GetType());
a = true;
Console.WriteLine(a.GetType());
a = 3.14M;
Console.WriteLine(a.GetType());
```

VALUE VE REFERENCE TİPLER

- Değişkenler bellekte tutulan verilerdir.
- Aslında bir değişkeni kullanırken o değişkenin bellekte tutulduğu adresteki veriye ulaşıyoruz.
- Değer tipleri değişkenin değerini direkt bellek bölgesinden alırlar.
- Referans tipleri ise başka bir nesneye referans olarak kullanırlar. Yani heap alanında tutulan veri türlerinin adreslerini saklarlar.
- Değer tipleri stack bölgesinde oluşturulurken referans tipleri ise heap bölgesinde saklanırlar.
- C, C++ dillerine aşina olanlar pointer kavramından yola çıkarak referans tiplerinin mantığını kavrayacaklardır.
- İki değer tipi nesnesini birbirine eşitlerken değişkenlerde saklanan değerler kopyalanarak eşitlenir ve bu durumda iki yeni bağımsız nesne elde edilmiş olur. Yani birinin değerini değiştirmek diğerini etkilemez.
- İki referans tipini birbirine eşitlediğimizde buralarda tutulan veriler kopyalanmaz. İşlem yapılan heap bölgesindeki adreslerdir. Aynı adres bölgesine işaret ettikleri için birinde yapılan değişiklik diğerini de etkileyecektir.

- Referans tipi tanımlarken hiçbir adres bölgesine işaret etmediğini belirtmek için null değere atanabilirler.

```
object o1 = null;
```

- Null değer bellekte herhangi bir adrese karşılık gelmez.

VALUE VE REFERENCE TİPLER

Value tipler	bool, byte, sbyte, char, decimal, double, float, int, uint, long, ulong, short, ushort, struct ve enum
Reference tipler	class, interface, delegate, object ve string

int x=5;

Stack	Heap
x=5	

object y=1;

Stack	Heap
x=5	
&y	y=1

```
int sayi1 = 5;

int sayi2 = 10;

string[] ogrenciler1 = new string[] { "Ahmet", "Yasin", "Özgür" };
```

Stack
İnt sayi1=5 İnt sayi2=10 Ogrenciler[]={heapteki adres = örn:100}

Heap	
100	«Ahmet», «Yasin»,….

Verinin Bellekte Tutulması

- **Verinin bellekte tutulması genel bölgeler;**

- **Stack Bölgesi:**
  - Genel anlamda Stack denildiği zaman RAM belleği anlarız.
  - Örneğin programımızda basit bir tamsayı tip tanımladığımız zaman çalışma zamanında yüklendiği yer RAM'in Stack bölgesidir.
  - Mikroişlemcide bulunan Stack Pointer ile RAM'in stack bölgelerine doğrudan erişilebilir.
  - SP o an bellekte çalışılan bölgenin adresini tutan yapıdır.
  - SP bellekteki alan tahsisatına göre arttırılıp azaltılır bu nedenle Stack bölgesine tutulacak verilerin çalışma zamanı öncesi ne kadar yer kapladıklarının bilinmesi gerekir.
  - JIT derleyicilerinin de program yüklendiğinde SP doğru konumlandırmak için verinin tam boyutunu bilmesi gerekir.

Verinin Bellekte Tutulması

- **Verinin bellekte tutulması genel bölgeler;**

- **Heap Bölgesi:**
  - Heap alanları da RAM'de bulunan hafıza alanlarıdır.
  - C# nesneleri bu alanda oluşturulur.
  - Stack'ten farklı olarak heap bölgesinde tahsisat yapılacak nesnenin derleyici tarafından bilinmesi zorunlu değildir. Programlarımıza büyük esneklik katmaktadır.
  - Heap bölgesinde bir nesneye alan tahsisatı yapmak için genellikle new anahtar sözcüğü kullanılır.
  - New kullanılarak tahsisatı yapılmış veriler çalışma zamanında dinamik olarak oluşturulurlar; yani derleme zamanında veriler için tahsisat yapılmaz.
  - Esneklik avantajının yanı sıra hızı ise Stack bölgesine göre daha yavaştır.

Verinin Bellekte Tutulması

- **Verinin bellekte tutulması genel bölgeler;**

- **Register Bölgesi:**
  - Stack ve Heap allocation mekanizmalarına göre çok hızlıdır, çünkü ikincil bellekte değildir.
  - Mikroişlemcinin içinde bulunan sınırlı sayıdaki yapılardır.
  - Derleyici çok sık işlem yaptığı verileri hız kazanmak için registerlarda tutar.
  - Kullanıcının doğrudan erişim hakkı yoktur, derleyicinin insiyatifindedir.

- **Static Bölge:**
  - Bellekte herhangi bir sabit bölgeyi ifade eder. Static alanlarda tutulan veriler programın bütün çalışma süresince saklanırlar.
  - C# da nesneye statik özelliği vermek için static anahtar sözüğü kullanılır.

- **Sabit Bölge:**
  - Sabit değerler genellikle program kodlarının içine gömülü şekildedir. Değişmesi mümkün değildir.
  - Sadece okuma amaçlı oldukları için hızlılık açısından bazen ROM(Read Only Memory) de tutululurlar.

- **Diğer Bölgeler:**
  - Bellek bölgesini temsil etmeyen disk alanlarıdır.
  - Bazı veri türlerinin kalıcı olması istenir, bu nedenle program sonlandığında disklere kaydedilebilir.
  - Program çalışmadığında da verinin bulunmasını istiyorsak bu bölgeleri kullanabiliriz.

TÜR DÖNÜŞÜMLERİ

- Program yazarken en sık yaptığımız hatalardan biri tür dönüşümleri ile ilgilidir.
  - Bazı durumlarda değişkenin farklı bir tür gibi davranması istenebilir.
    - Genel olarak ikiye ayırabiliriz:
    - **Bilinçli (explicit) ve bilinçsiz (implicit) tür dönüşümleri**

TÜR DÖNÜŞÜMLERİ

Bilinçsiz Tür Dönüşümü

- Bir değişkenin derleyici tarafından tanımladığımız türün dışında geçici olarak başka bir türe dönüştürülmesidir.

```
int deger1 = 10;
float deger2;
deger2 = deger1;
```

- Bu işlemi sağlayan derleyicinin gizlice sağladığı tür dönüşümdür.
- deger1 değişkeni geçici olarak float türüne dönüştürülür ve deger2 değişkenine atanır.
- Eğer geçici olmasaydı değişken programın farklı noktalarında farklı türden davranacak ve program karmaşası artacaktı.

TÜR DÖNÜŞÜMLERİ

Bilinçsiz Tür Dönüşümü

- **Küçükten büyüğe dönüştürme**
  - Yüksek anlamlı bitlerin sıfırla beslenmesi değeri değiştirmeyeceği için veri kaybı olmaz


```
float f = 20f;
Double d;
d = f;
```

```
int a = 10;
byte b = 20;
short s = 30;
double d;
d = a + b + s;
```

TÜR DÖNÜŞÜMLERİ

- **Bilinçsiz Tür Dönüşümü**

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

- Dönüşüm yapılamaz.

	Dönüşüm yapılabilir türler
Sbyte	Short,int,float,long,double,decimal
Byte	Short, ushort, int, uint,long,ulong,float,double,decimal
Short	İnt,long,float,double,decimal
Ushort	İnt,uint,long,ulong,float,double,decimal
İnt	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

TÜR DÖNÜŞÜMLERİ

Bilinçsiz Tür Dönüşümü

- Büyük türden küçük türe otomatik dönüşüm C# dilinde yasaklanmıştır.
- Çeşitli veri kayıpları önlenir.
- Tür dönüştürme operatörlerinin kullanılması gerekir.

```
decimal sayi = 10;
byte sayi2 = sayi;
```

CS0266 Cannot implicitly convert type 'decimal' to 'byte'. An explicit conversion exists (are you missing a cast?)

TÜR DÖNÜŞÜMLERİ

- **Bilinçli Tür Dönüşümü**

- Genellikle derleyicinin izin vermediği dönüşümlerde yapılır.
- Veri kayıplarına sebep olacağı için dikkatli olunmalıdır.

Tür dönüştürme operatörü

- (dönüştürelecek tür) değişken yada sabit ifade

```
byte b = 10;
int i = (byte)b;
```

- Yukarıdaki dönüşüm operatörsüzde yapılabilirdi ama burda kod okunabilirliğini arttırmaktadır. Mümkün olduğunca kullanılması tavsiye edilir.

TÜR DÖNÜŞÜMLERİ

- **Bilinçli Tür Dönüşümü**

```
int i = 10;
byte b = (byte)i;
```

- i nin bellekteki durumu
  - 00000000 00000000 00000000 00001010

- byte dönüşümünde son byte kalır. Yüksek anlamlılar gider.

- i=256 yapılırsa veri kaybı yaşarız.

- Tür Dönüşüm Operatörü sabitlere de uygulanabilir.

```
double d = (double)25.35;
```

IMPLICIT VE EXPLICIT TİP DÖNÜŞÜMLERİ

```
int sayi1 = 10;
float sayi2 = sayi1;

float sayi3 = 20.0f;
int sayi4 = sayi3; // HATA: float int'ten büyüktür

float sayi5 = 30.0f;
int sayi6 = (int)sayi5;
```

İŞLEM SONUÇ TİPLERİ

```
int sayi1 = 5, sayi2 = 2;
float sonuc = sayi1 / sayi2; // sonuc=2.0f
```

- İşleme giren tüm elemanlar int olduğu için sonucun da int olacağı kabul edilir ve işlem sonucundaki virgülden sonraki kısım atılır.

```
int sayi1 = 5, sayi2 = 2;
float sonuc = (float)sayi1 / sayi2; // sonuc=2.5f
```

- İşleme giren elemanlardan biri float biri int olduğu için sonucun float (büyük olan) olacağı kabul edilir.

Checked unchecked

- Veri kayıplarına neden olabilecek tür dönüşümlerinde derleyici hata üretmemektedir.
- Checked anahtar sözcüğü ile bu gibi durumlarda hata vermesini sağlarız.

```
int i = 256;
Byte b;

checked
{
    b = (byte)i;
}

Console.ReadLine(
```

Exception Unhandled

System.OverflowException: 'Aritmetik işlem taşmayla sonuçlandı.'

View Details | Copy Details | Start Live Share session...

▷ Exception Settings

- İstisnai durum yakalama (Exception Handling) konusuna değinmeye çalışacağız.

- Uynchecked checked tam tersidir ve varsayılan olandır.
- Bazen checked içinde bazı yerleri unchecked yapmak isteyebiliriz.

Referans ve Değer Türleri Arası Dönüşüm

- Object sınıfına ait olan ToString() metodu bütün temel veri türleri ve referans türlerinde kullanılabilir.

```
3.ToString();
```

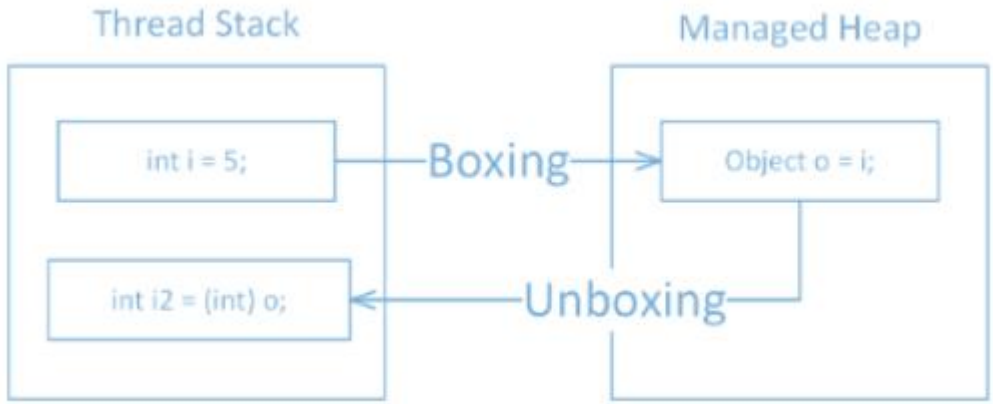
- . (Nokta) Operatörü Üye elemanlara ulaşmak için kullanılır. İleride değineceğiz.

```
int a = 5;
int b = 7;

Console.WriteLine(a + b);

string a1 = a.ToString();
string a2 = b.ToString();

Console.WriteLine(a1 + a2);
```

BOXING VE UNBOXING

- **Boxing: Stack alanından Heap alanına taşıma**
- **Bir nesnenin object türüne dönüştürülmesi.**

  - object o;
  - int i=5;
  - o=i;



- **Unboxing: Heap alanından Stack alanına taşıma**

  - object o;
  - int i=5, i2=10;
  - o=i;
  - i2=(int)o;

  - Runtime hata almamak için
  - 1- unboxing yapılalcak nesnenin daha önceden boxing işlemine tabi tutulmuş olması
  - 2- Boxing işlemine tabi tutulmuş bu nesnenin unboxing işlemi sırasında doğru türe dönüştürülmesi.

TİP DÖNÜŞÜMÜ: Parse

```
string sayi1 = "1234";
int x = int.Parse(sayi1);

string sayi2 ="12.345";
int y = int.Parse(sayi2);   // çalışma zamanı hatası

long s = long.Parse("123456");
```

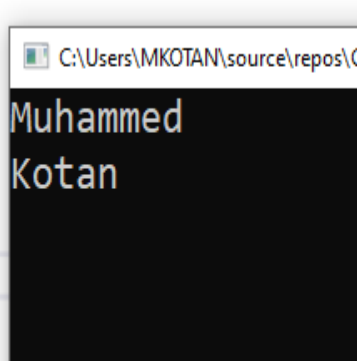
TİP DÖNÜŞÜMÜ: Convert

- int sayi = Convert.ToInt32(“123”);

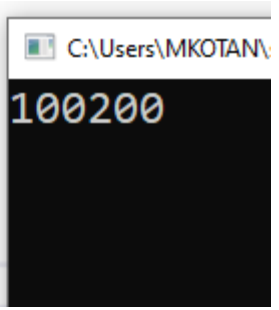
Dönüşüm metodu	Açıklama
Convert.ToBoolean (ifade)	Sayısal ifadeyi **bool** tipine dönüştürür.
Convert.ToChar (ifade)	Tek karakteri **char** tipine dönüştürür.
Convert.DateTime (ifade)	Geçeri bir tarih veya zamanı **DateTime** tipine dönüştürür.
Convert.ToSingle (ifade)	Bir ifadeyi **float** tipine dönüştürür.
Convert.ToDouble (ifade)	İfadeyi **double** tipine dönüştürür.
Convert.ToInt16 (ifade)	Bir ifadeyi **short** tipine dönüştürür.
Convert.ToInt32 (ifade)	Bir ifadeyi **int** tipine dönüştürür.
Convert.ToInt64 (ifade)	Bir ifadeyi **long** tipine dönüştürür.

KONSOLA YAZDIRMA

```
var ad = "Muhammed";
var soyad = "Kotan";

Console.WriteLine(ad);
Console.WriteLine(soyad);
```




```
var sayi1 = 100;
var sayi2 = 200;

Console.Write(sayi1);
Console.Write(sayi2);
```



Çalışma Sorusu

- Klavyeden girilen 2 sayı üzerinde matematiksel işlemler gerçekleştirerek ekrana yazdıralım.

Operatörler

- Önceden tanımlanmış belirli görevleri gerçekleştiren özel karakter yada karakter topluluğudur.

- a+b işleminde + sembolü operatör, a ve b ise operand dır.

- İşlevlerine göre farklı sınıflara ayrılabilir.
  - Aritmetik Operatörler
  - Karşılaştırma Operatörleri
  - Mantıksal Operatörleri
  - Bitsel Operatörler
  - Atama ve işlemli atama
  - Özel amaçlı operatörler

ARİTMETİK OPERATÖRLER

İŞLEM	OPERATÖR	ÖRNEK
Toplama	+	x + 3
Çıkarma	-	x - 3
Çarpma	*	x * 3
Bölme	/	x / 3
Mod Alma	%	x % 3
Değeri 1 arttırma	++	Sayi++ veya ++Sayi
Değeri 1 azaltma	--	Sayi-- veya --Sayi

```
int i = 50 / 40;
float j = 50f / 40f;

int a = 10;
int b;
int c;

b = a++;
c = ++a;

Console.WriteLine("i:{0} \nj:{1} \na:{2} \nb:{3} \nc:{4}", i, j, a, b, c);
```

```
i:1
j:1,25
a:12
b:10
c:12
```

KARŞILAŞTIRMA OPERATÖRLERİ

İŞLEM	OPERATÖR	ÖRNEK
Büyüktür	>	x > 3
Küçüktür	<	x < 3
Büyük veya eşittir	>=	x >= 3
Küçük veya eşittir	<=	x <= 3
Eşittir	==	x == 3
Farklıdır	!=	x != 3
Uygun dönüşüm	as Kullanımı pek yaygın değildir.	Object i=«50»; String s=i as string
Tür uyumu kontrolü	is	İnt i=50 Bool b=i is int; Bool b2=i is double; Bool b3=i as object;

MANTIKSAL OPERATÖRLER

İŞLEM	OPERATÖR	ÖRNEK
Ve	&&	(x > 5) && (y < 4)
Veya	||	(x > 5) || (y < 4)
Değil	!	!(x > 5)

BİT İŞLEM OPERATÖRLERİ

- Bitsel operatörler sayıları bir bütün olarak ele almak yerine sayıları oluşturan bitler üzerinde işlem yaparlar.
- Tamsayılar üzerinde uygulanır. Gerçel sayılarla kullanılmazlar.

- Not: bool türünden operandlar ile kullanılırlarsa mantıksal operatör görevi görürler.

BİT İŞLEM OPERATÖRLERİ

- ~5 ifadesinin sonucunun -6 çıkmasının sebebi işaretli tamsayıların ikiye tümleyen formunda saklanmasıdır.

İŞLEM	OPERATÖR	ÖRNEK
Ve	&	5 & 3 (sonuç = 1)
Veya	|	5 | 3 (sonuç = 7)
xor	^	5 ^ 3 (sonuç = 6)
Değil	~	~255 (sonuç = 0) ~5 (sonuç = -6)
Bitsel sola kaydırma	<< İlk bit sıfırla beslenir son bit ise ötelemeden dolayı atılır.	x<<2 Sola bir defa öteleme sizce ne sonuç üretir?
Bitsel sağa kaydırma	>> son bit sıfırla beslenir ilk bit ötelemeden dolayı atılır.	X>>2 Sağa bir defa öteleme ne sonuç üretir?

ATAMA OPERATÖRLERİ

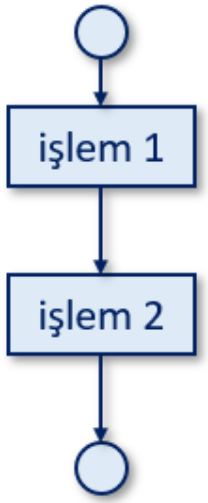
İŞLEM	OPERATÖR	ÖRNEK
Değer atama	=	Sayi = 5
Toplayarak atama	+=	Sayi += 5
Çıkararak atama	-=	Sayi -= 5
Çarparak atama	*=	Sayi *= 5
Bölerek atama	/=	Sayi /= 5

ÖZEL AMAÇLI OPERATÖRLER

İŞLEM	OPERATÖR	ÖRNEK
İf-else yapısı gibi	?:	Sayi==1? «tekil»:»çoğul»
Tür dönüştürme	()	(int)a
İndeks	[]	a[0]
İşaret operatörü	+,-	(int)-a
İşaretçi operatörleri	&,*,sizeof	Unsade kod yazarken. C,C++ dillerindeki işaretçi işlemleri
Sınıf veya yapının elemanlarına ulaşım	.	Console.WriteLine

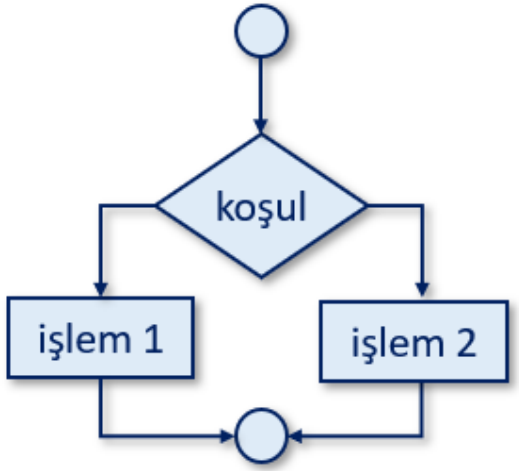
- Programlar 3 bloktan oluşur
  - **Sıralı**

    Bir dizi işlem birbiri ardından sırayla gerçekleştirilir.

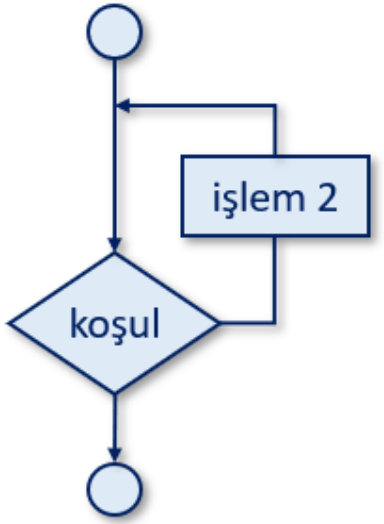
  - **Seçme / Kontrol**

    İki seçenekten hangisinin izleneceği koşula bağlıdır.

  - **Döngü**

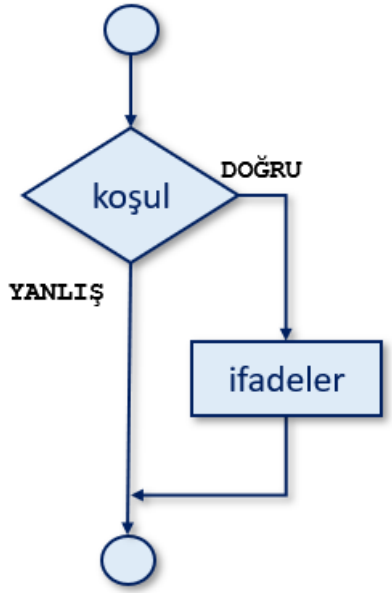
    İki seçenekten hangisinin izleneceği koşula bağlıdır.

Program Blokları

- **if Kontrol Yapısı**
  - Koşul boolean bir ifadedir
    - 1(true) / 0(false)
  - Eğer ((not>70) && (not<80)) ise C yaz

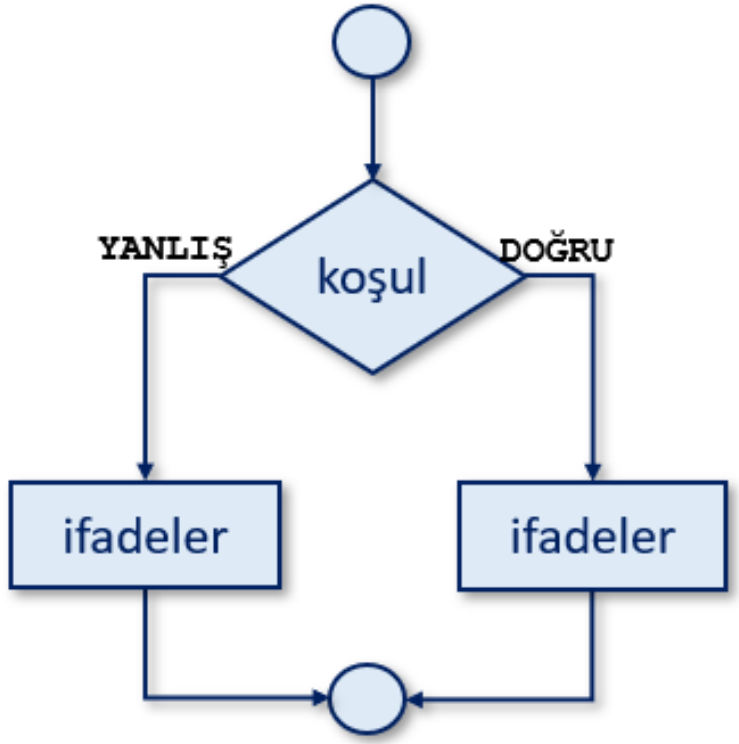
```
if ( koşul )
{
   ifadeler...
}
```



Karar Verme/Kontrol Yapıları

if/else Kontrol Yapısı

```
if ( koşul )
{
   ifadeler...
}
else
{
   ifadeler...
}
```



Karar Verme/Kontrol Yapıları

Karar Verme/Kontrol Yapıları

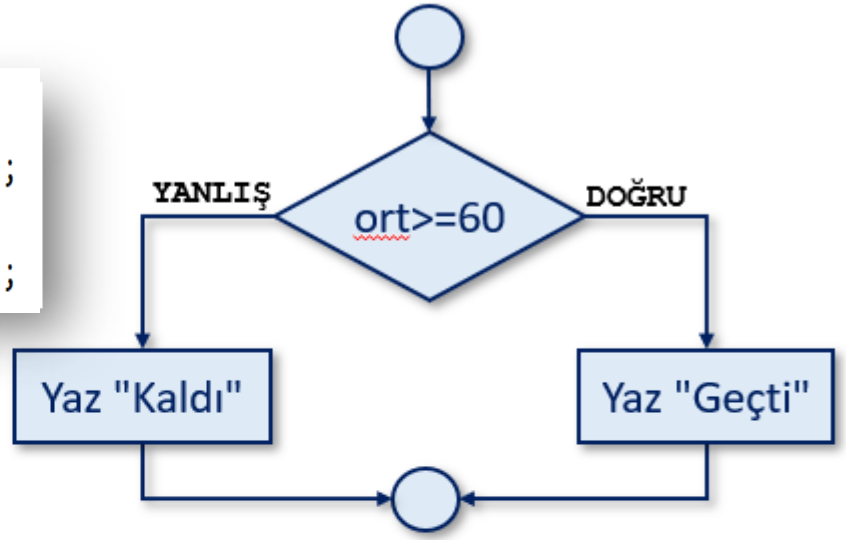
- **if/else Kontrol Yapısı**
  - if (ort>=60)

    "Geçti ☺" yaz

    else

    "Kaldı ☹" yaz

```
if (ort>=60)
    Console.WriteLine("Geçti");
else
    Console.WriteLine("Kaldı");
```



- Kısa if/else operatörü (Ternary conditional operatör) **(?:)**

```
sonuc = (ort < 50 ? "Kaldi" : "Gecti");
```

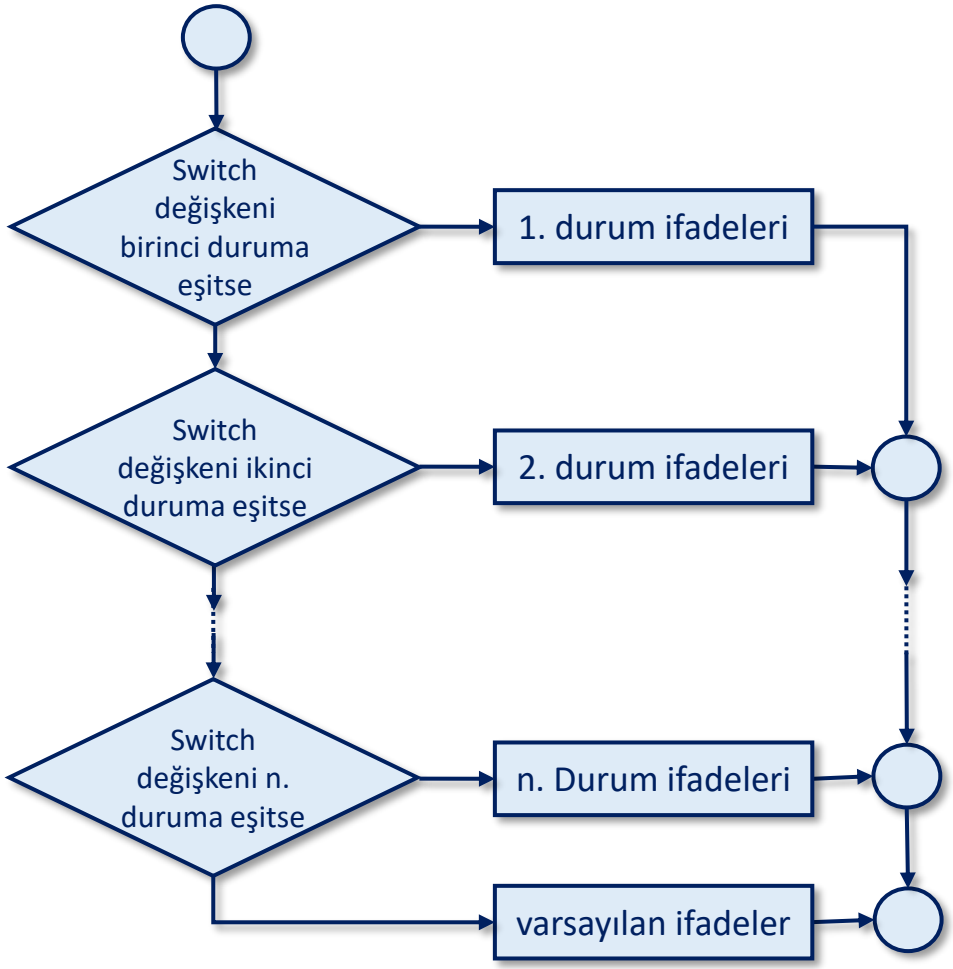
Koşul — Doğru (true) durum işlemi (if) — Yanlış(false) durum işlemi (else)

Çalışma Sorusu

Girilen Sınav Notuna göre öğrencinin Harf Notunu hesaplayan programı yazınız.

- **Çoklu Dallanma Yapısı –> switch-case**

```
switch ( değişken) {
   case sabit1:
      ifadeler
      break;
   case sabit2:
      ifadeler
      break;
.
.
.
default:
   varsayılan ifadeler…
}
```

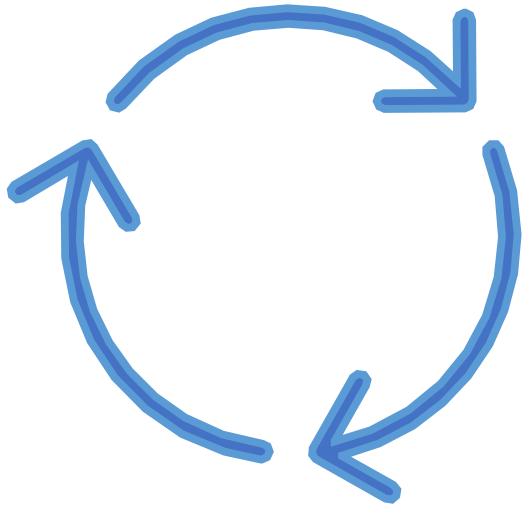


Karar Verme/Kontrol Yapıları

Çalışma Sorusu

Girilen sayıya göre haftanın gününü gösteren programı yazınız.

Menüden yapılan seçime göre işlem yapan hesap makinası programını yazınız.

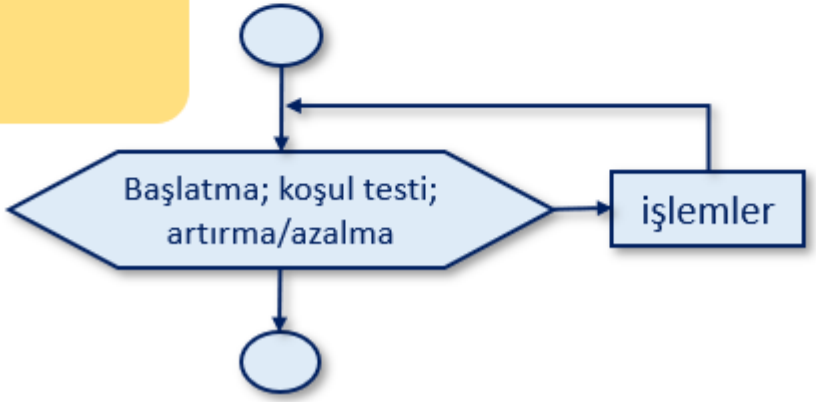


Döngüler

- Döngüler, program içerisinde belirli işleri defalarca yapmamızı sağlayan komut bloklarıdır.
- Temel 4 tip döngü vardır:
  - For
  - While
  - Do while
  - foreach

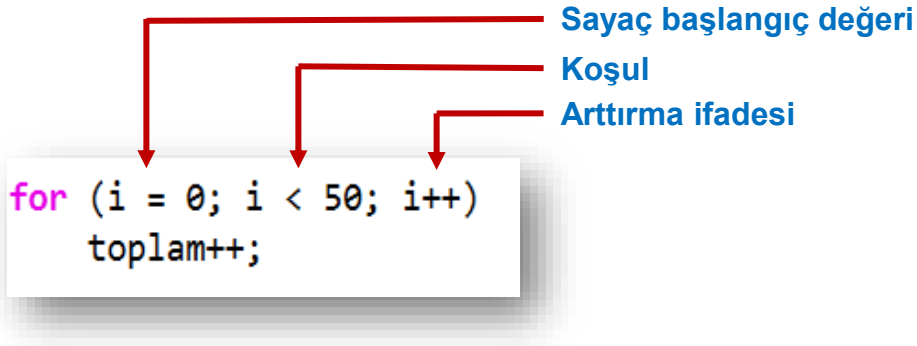
- **For**
  - Programın bir parçasını sabit sayıda çalıştırır.
  - Koşul sınaması çevrime girmeden yapılır.
  - Döngüye girmeden önce sayaç başlangıç değeri alır ve daha sonra koşula bakılır.
  - Döngü içerisindeki işlemler yapıldıktan sonra sayaç üçüncü parametrenin durumuna göre değiştirilir (artırılır/eksiltilir).

```
for ( başlatma; koşul testi;artırma/azaltma ) {
      ifade(ler)
}
```



Döngü / For

- **For…**
  - Tek ifadeli **for** döngüsü



  - Birden çok ifadeden oluşan **for** döngüsü

```
for (i = 0; i < 50; i++)
{
    sayi++;
    toplam = toplam + i * 5;
}
```

Döngü / For

DÖNGÜLER: for

```
var sayi = 0;

for (var i = 0; i < 5; i++)
{
    sayi++;
}


Console.WriteLine(sayi);
```

Kod parçası çalıştırıldığında sonuç ne olur?

Çalışma Sorusu

```
char ch;
for (ch = Convert.ToChar(Console.ReadLine()); ch != 'q'; ch = Convert.ToChar(Console.ReadLine()))

    Console.WriteLine(ch);
```

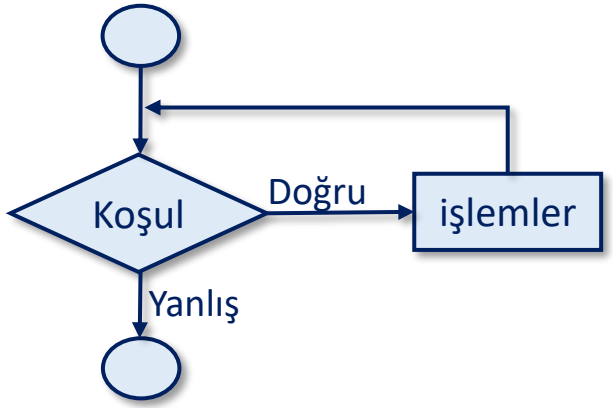
Yukarıdaki for döngüsünün görevi sizce nedir?

Çalışma Sorusu

1-100 arasındaki tüm sayıları, tek sayıları ve çift sayıları ekrana yazdıran programı for döngüsü kullanarak yazınız.

- **While**
  - For döngüsü bir işi belli bir sayıda tekrarlamaya yararken while döngüsünde ise döngüye girmeden ne kadar tekrarlamanın yapılacağı bilinmez.
  - Bu döngüde de koşul sınaması çevrime girmeden yapılır.
  - Koşul tek bir karşılaştırmadan oluşabileceği gibi birden çok koşulun mantıksal operatörler ile birleştirilmesi ile de oluşturulabilir.

```
while ( koşul ) {
    ifade(ler)
}
```




```
while (i <= 10)
{
    toplam += i;
    i++;
}
```
Koşul

Döngü / While

DÖNGÜLER: while

```
var sayi = 0;

while (sayi < 5)
{
    sayi++;
}

Console.WriteLine(sayi);
```

Kod parçası çalıştırıldığında sonuç ne olur?

DÖNGÜLER: while

```
var sayi = 100;

while (sayi < 5)
{
    sayi++;
}


Console.WriteLine(sayi);
```

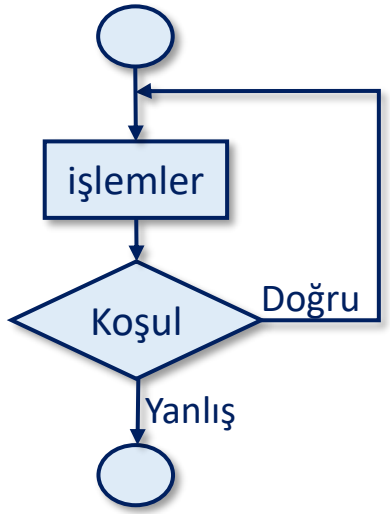
Kod parçası çalıştırıldığında sonuç ne olur?

Çalışma Sorusu

1-100 arasındaki tüm sayıları büyükten küçüğe doğru while döngüsü kullanarak yazdırınız.

- **Do-while**
  - Diğer döngüler gibi aynı işlemleri birçok kez tekrarlamak için kullanılır.
  - Farklı olarak, bu döngüde koşul sınaması yapılmadan çevrime girilir ve işlem kümesi en az bir kere işletilir. Bu deyim yapısında da koşul sağlandığı sürece çevrim tekrarlanır.
  - Koşul tek bir karşılaştırmadan oluşabileceği gibi birden çok koşulun mantıksal operatörler ile birleştirilmesi ile de oluşturulabilir.

```
do {
    ifade(ler)
} while ( koşul );
```




```
do
{
    toplam += i;
    i++;
} while (i <= 10);
```

Koşul

Döngü / Do While

DÖNGÜLER: do while

```
var sayi = 0;

do
{
    sayi++;
} while (sayi < 5);

Console.WriteLine(sayi);
```

Kod parçası çalıştırıldığında sonuç ne olur?

DÖNGÜLER: do while

```
var sayi = 100;

do
{
    sayi++;
} while (sayi < 5);


Console.WriteLine(sayi);
```

Kod parçası çalıştırıldığında sonuç ne olur?

- **foreach**

- Koleksiyon tabanlı nesneler içerisinde adım adım dolaşmamızı sağlar.
- Dizi elemanlarını gezerken, veri tabanından tablo çekerken datayı gezerken…
- Ulaştığımız eleman sadece okunabilir(readonly) özelliktedir.

Döngü / foreach

```
var dizi = new int[] { 1, 2, 3, 4 };

foreach (var sayi in dizi)
{
    Console.WriteLine(sayi);
}
```

```
1
2
3
4
```

DÖNGÜLER: foreach

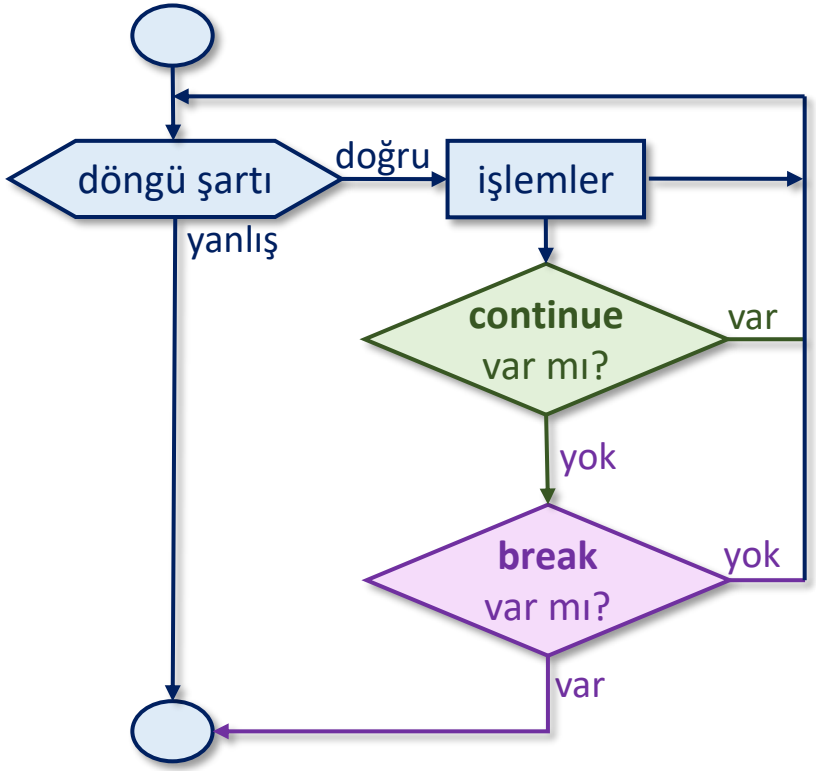
```
string[] yazAylari = new string[3] { "Haziran", "Temmuz", "Ağustos" };

foreach(var ay in yazAylari)
{
    ay = "Eylül";
    Console.WriteLine(ay);
}
```

CS1656 Cannot assign to 'ay' because it is a 'foreach iteration variable'

foreach ile gezdiğimiz elemanlar readonly durumdadır.

Çalışma Sorusu

Klavyeden girilen sayının Asal Sayı olup olmadığını kontrol eden programı yazınız.



Continue-Break

- **continue** ifadesi
  - **while**, **for**, **do/while**
    - Döngünün kalanı atlanır
    - Bir sonraki iterasyona geçilir
  - **for**
    - **continue** ifadesinden sonra artırım ifadesi çalıştırılır.
  - **while, do/while**
    - **continue** ifadesinden sonra koşul testine gidilir.
- **break** ifadesi
  - Döngüden çıkılır

DÖNGÜLER: break

```
var sayi = 0;

for (var i = 0; i < 5; i++)
{
    if (i == 2) break;
    sayi++;
}

Console.WriteLine(sayi);
```

Kod parçası çalıştırıldığında sonuç ne olur?

DÖNGÜLER: continue

```
var sayi = 0;

for (var i = 0; i < 5; i++)
{
    if (i == 2) continue;
    sayi++;
}

Console.WriteLine(sayi);
```

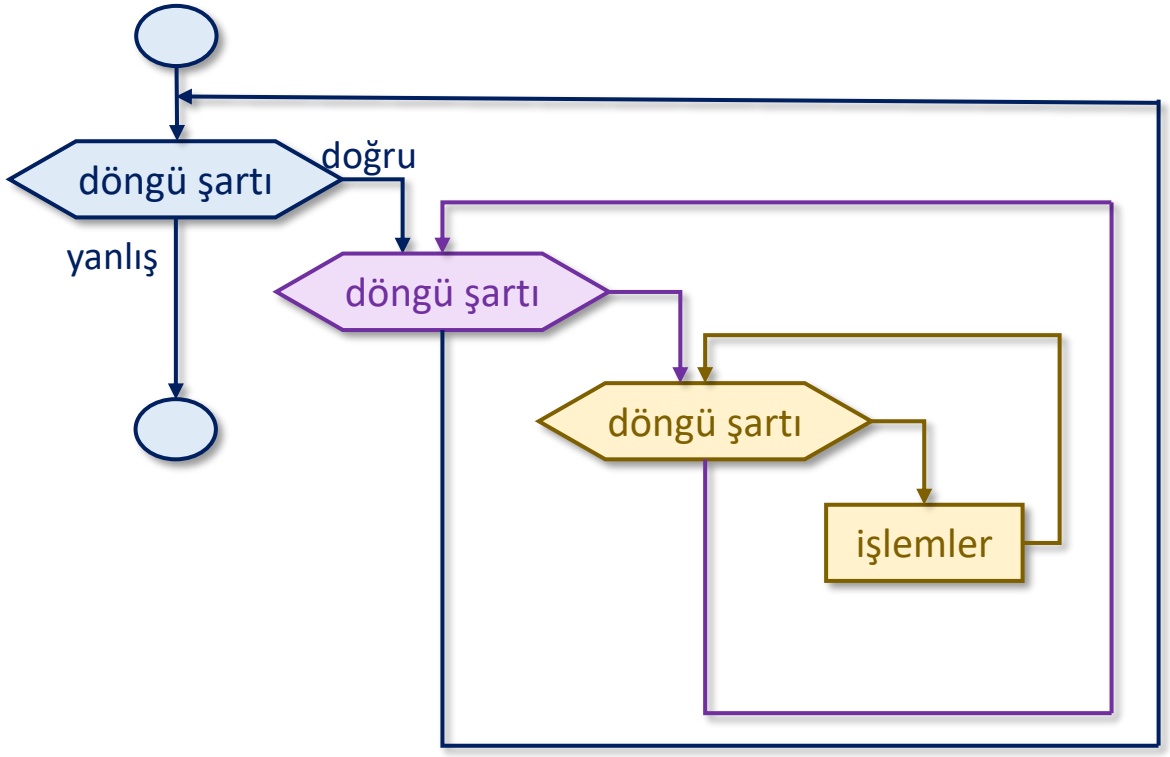
Kod parçası çalıştırıldığında sonuç ne olur?

Çalışma Sorusu

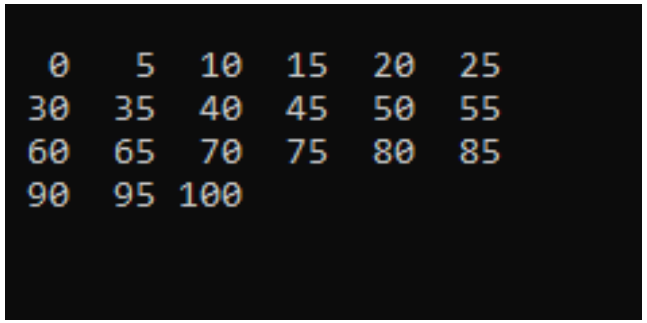
Kullanıcı bir önceki sayıdan farklı sayı girdiğinde sayıların toplamını ekrana gösteren 0 girdiğinde ise programı sonlandıran kodu yazınız.

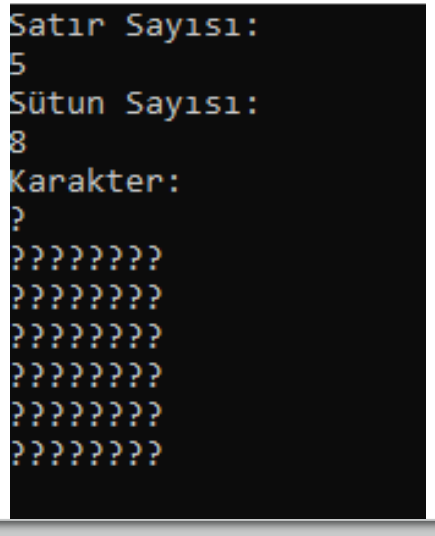
DÖNGÜLER:
İç İçe Döngüler

- Tüm döngüler iç-içe yapılandırılabilir
- Örnek kullanım alanları
  - Çok boyutlu diziler
  - Seri hesaplamaları
  - İlişkili döngüler
  - …



Çalışma Soruları





1. 1-1000 arasındaki sayılardan 5 ile tam bölünen ama 7 ile tam bölünemeyen sayıları ekrana yazdırınız.
2. Yandaki görüntüyü ekrana yazdıran for döngüsünü yazınız.
3. Satır sayısı, sütun sayısı ve karakter dışarıdan girilecek şekilde yandaki ekran çıktısını üreten kodu yazınız
4. Kullanıcının gireceği byte türünden sayının bitlerini ekrana yazdıran programı yazınız.
5. 0-100 arası girilen 10 notun en büyük en küçük ve ortalamalarını bulan programı yazınız

Diziler

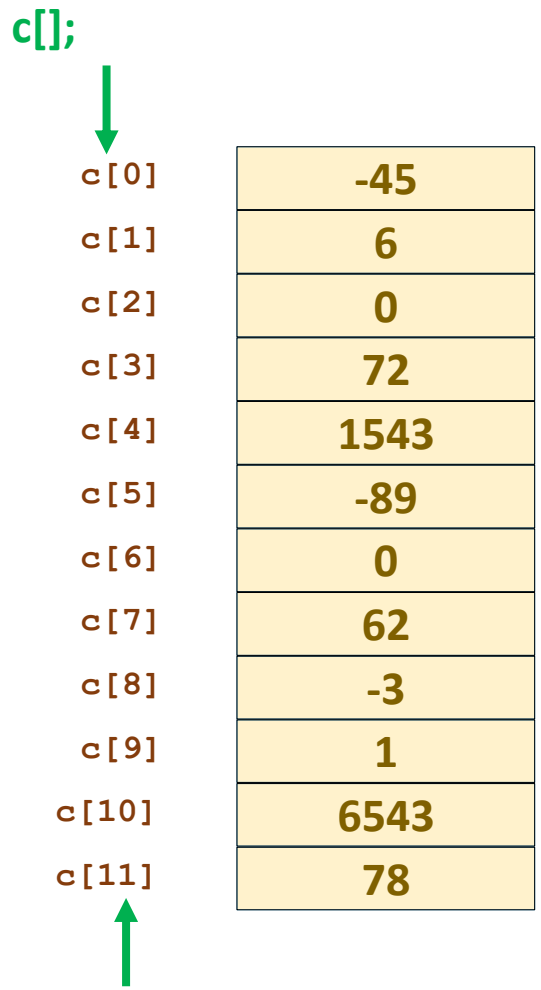
- Bellekte ard arda yer alan aynı türden nesneler kümesine dizi denilir.
- Dizi içindeki bütün elemanlara aynı isimle ulaşılır. Ayırt edici özellik bellekteki yerleridir.
- Dizi elemanlarına [] indeks operatörü ile ulaşlılır. İndeks numarası 0 dan başlar.
- C# dilinde diziler System.Array sınıfından türemiş ayrı bir tür olarak tasarlanmıştır. (C,C++ daki pointer mantığına dikkat).

```
int[] dizi = new int[25];
int i = dizi[0];
int j = dizi[24];

string[] dizi2 = { "Bir", "İki", "Üç" };
```

Dizi boyutu C ve C++ dilinde derleme sırasında bilinmesi gerekli. Dinamik bellek tahsisi için çeşlitli mekanizmalar bulunmaktadır.
C# dilinde diziler referans tipi olduğu için boyutu çalışma zamanında belirlenebilir.

• Dizi Tanımlama…

```
int[] a = new int[] { 1, 2, 3, 4, 5 };
int[] b = { 7, 21, 35, 14, 5 };
int[] c = new int[12];
int[] d;
d = new int[10];
```



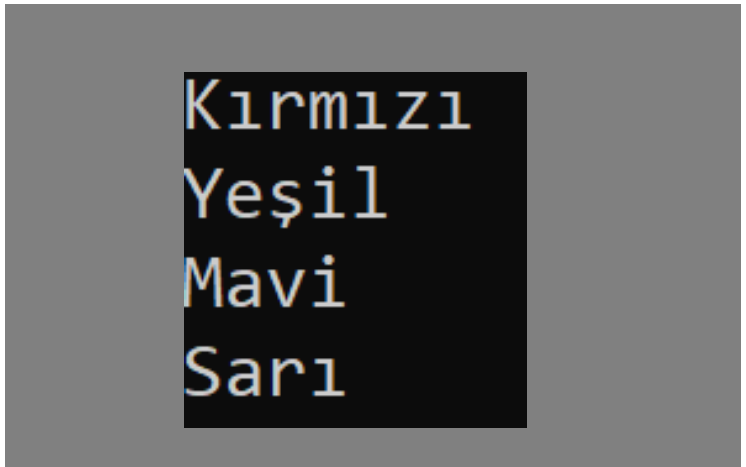
- **Diziler…**

```
string[] kelimeler = { "Sakarya", "Üniversitesi" };

char[] s = {'M','e','r','h','a','b','a'};
// string s = "Merhaba"; ile aynı değil

int sayi = Convert.ToInt32(Console.ReadLine());
//int sayi = Convert.ToInt32(elemanSayisiTextBox.Text);
int[] dizi = new int[sayi];

// Diziler dinamik olarak başlatılabilir
```

Diziler

```
string[] renkler = { "Kırmızı", "Yeşil", "Mavi", "Sarı" };

for (var i = 0; i < 4; i++)
{
    var renk = renkler[i];
    System.Console.WriteLine(renk);
}
```

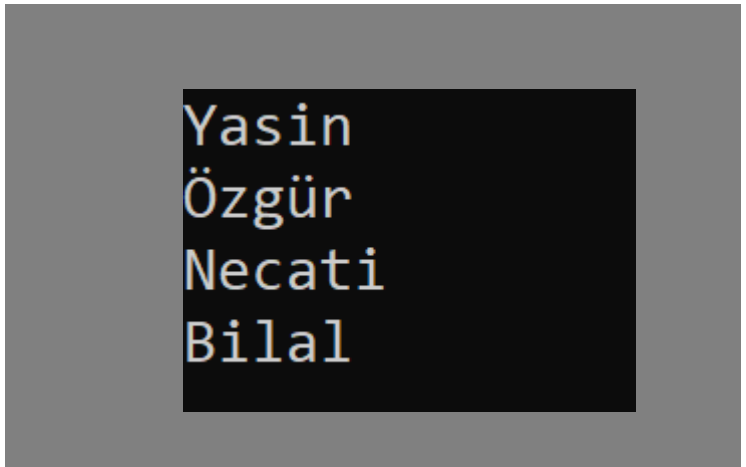
Ekran



Diziler

```
string[] oyuncular = new string[4];
oyuncular[0] = "Yasin";
oyuncular[1] = "Özgür";
oyuncular[2] = "Necati";
oyuncular[3] = "Bilal";

//yada

string[] oyuncular2 = { "Yasin", "Özgür", "Necati", "Bilal" };

foreach(var oyuncu in oyuncular2)
{
    Console.WriteLine(oyuncu);
}
```

Ekran



Diziler-Örnek: Dizi Elemanlarını Grafiksel Olarak Çizdirme

Ekran

```
Dizi Boyutunu Giriniz
5
Dizi elemanlarını giriniz
4
7
9
1
2
 İndis   Değer   Grafik
 0       4       ****
 1       7       *******
 2       9       *********
 3       1       *
 4       2       **
```

```
//Dizi boyutunu kullanıcıdan alma
Console.WriteLine("Dizi Boyutunu Giriniz");
int boyut = Convert.ToInt32(Console.ReadLine());

//Diziyi tanımla
int[] sayilar = new int[boyut];

//Dizi elemanlarını kullanıcıdan isteme
Console.WriteLine("Dizi elemanlarını giriniz");
for (int i = 0; i < boyut; i++)
    sayilar[i] = Convert.ToInt32(Console.ReadLine());

//Başlık Yazdırma
Console.WriteLine(" {0} \t {1} \t {2}","İndis","Değer","Grafik");

//Grafik çizdirme
for(int i=0;i<boyut;i++)
{
    Console.Write(" {0} \t {1} \t ",i,sayilar[i]);
    for (int j = 0; j < sayilar[i]; j++)
        Console.Write("*");
    Console.WriteLine();
}
```

Rastgele Değer Üretme

- .Net Framework kütüphanesinden System.Random sınıfını kullanacağız.

```
Random rnd = new Random();
int rs1 = rnd.Next(10, 20); //10-20 arası bir sayı üret. 20 dahil değil.
int rs2 = rnd.Next(50);   // 0-50 arası bir sayı üret. 50 dahil değil.
int rs3 = rnd.Next();     //pozitif türden bir sayı üret.
double rs4 = rnd.NextDouble(); // 0.0 ile 1 arası

Console.WriteLine(rs1);
Console.WriteLine(rs2);
Console.WriteLine(rs3);
Console.WriteLine(rs4);
```

```
11
8
1647542710
0,487912128906656
```

Diziler-Örnek: Rastgele Sayı Üretimi

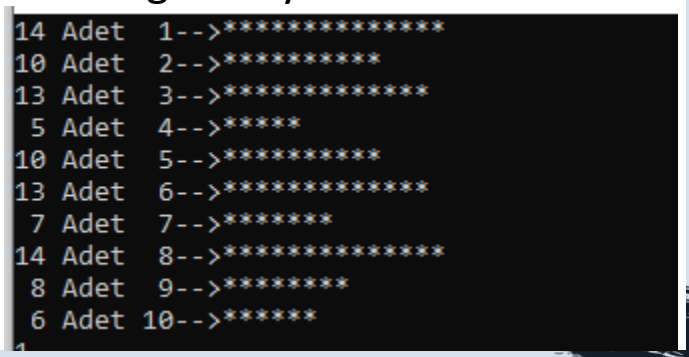
```
Random rnd = new Random();
int[] rastgeleDizi = new int[10];

Console.WriteLine("İndis \t Değer");

for (int i = 0; i < 10; i++)
{
    rastgeleDizi[i] = rnd.Next(100);
    Console.WriteLine("{0} \t {1}",i,rastgeleDizi[i]);
}
```

Ekran

```
İNDİS     DEĞER
  0         41
  1         67
  2         34
  3          0
  4         69
  5         24
  6         78
  7         58
  8         62
  9         64
Press any key to continue . . .
```

Çalışma Sorusu

100 elemanlı bir dizinin bütün elemanlarına 1-10 arası rastgele değerler vererek bu rastgele sayıların her birinden kaç adet üretildiğini grafiksel olarak gösterelim.



ÇOK BOYUTLU DİZİLER

```
int[,] dizi1 = new int[3, 3];

int[,] dizi2 = { { 1, 2 }, { 3, 4 }, { 10, 11 } };

int[,,] dizi3 = new int[5, 5, 5];
```

```
var dizi1 = new int[3, 3];
var dizi2 = new int[5, 5, 5];
```

ÇOK BOYUTLU DİZİLER

```
int[,] dizi={{1,2}, {3,4},{5,6}};

dizi[0, 0] = 1;
dizi[0, 1] = 2;
dizi[1, 0] = 3;
dizi[1, 1] = 4;
dizi[2, 0] = 5;
dizi[2, 1] = 6;
```

- Bellekle bu şekilde tutulmazlar. Gösterim kolaylığı için böyle gösterilmiştir.

Dizi[0,0]	Dizi[0,1]	Dizi[0,2]	Dizi[0,3]
Dizi[1,0]	Dizi[1,1]	Dizi[1,2]	Dizi[1,3]
Dizi[2,0]	Dizi[2,1]	Dizi[2,2]	Dizi[2,3]

ÇOK BOYUTLU DİZİLER – Örnek

```
string[,] bolgeler = new string[7, 3]
{
    {"Sakarya","Kocaeli","Bursa"},
    {"İzmir","Manisa","Denizli"},
    {"Antalya","Mersin","Isparta"},
    {"Ankara","Konya","Kayseri"},
    {"Muş","Van","Malatya"},
    {"Gaziantep","Şanlıurfa","Diyarbakır"},
    {"Trabzon","Rize","Zonguldak"}
};

for (int i = 0; i <= bolgeler.GetUpperBound(0); i++)
{
    for (int j = 0; j <= bolgeler.GetUpperBound(1); j++)
        Console.WriteLine(bolgeler[i, j]);
    Console.WriteLine("----------------");
}
```

```
Sakarya
Kocaeli
Bursa
----------------
İzmir
Manisa
Denizli
----------------
Antalya
Mersin
Isparta
----------------
Ankara
Konya
Kayseri
----------------
Muş
Van
Malatya
----------------
Gaziantep
Şanlıurfa
Diyarbakır
----------------
Trabzon
Rize
Zonguldak
----------------
```

ÇOK BOYUTLU DİZİLER

```
var matris =new[,]
    {
{10, 12, 20, 22},
{17, 22, 19, 13},
{10, 12, 20, 22},
{17, 22, 19, 13}
    };

Console.WriteLine("Matris" );

for (var i = 0; i < 4; i++)
{
    for (var j = 0; j < 4; j++)
    {
        Console.Write(" " + matris[i, j] + " ");
    }

    Console.WriteLine();
}
```

Ekran Çıktısı

```
Matris
 10  12  20  22
 17  22  19  13
 10  12  20  22
 17  22  19  13
```

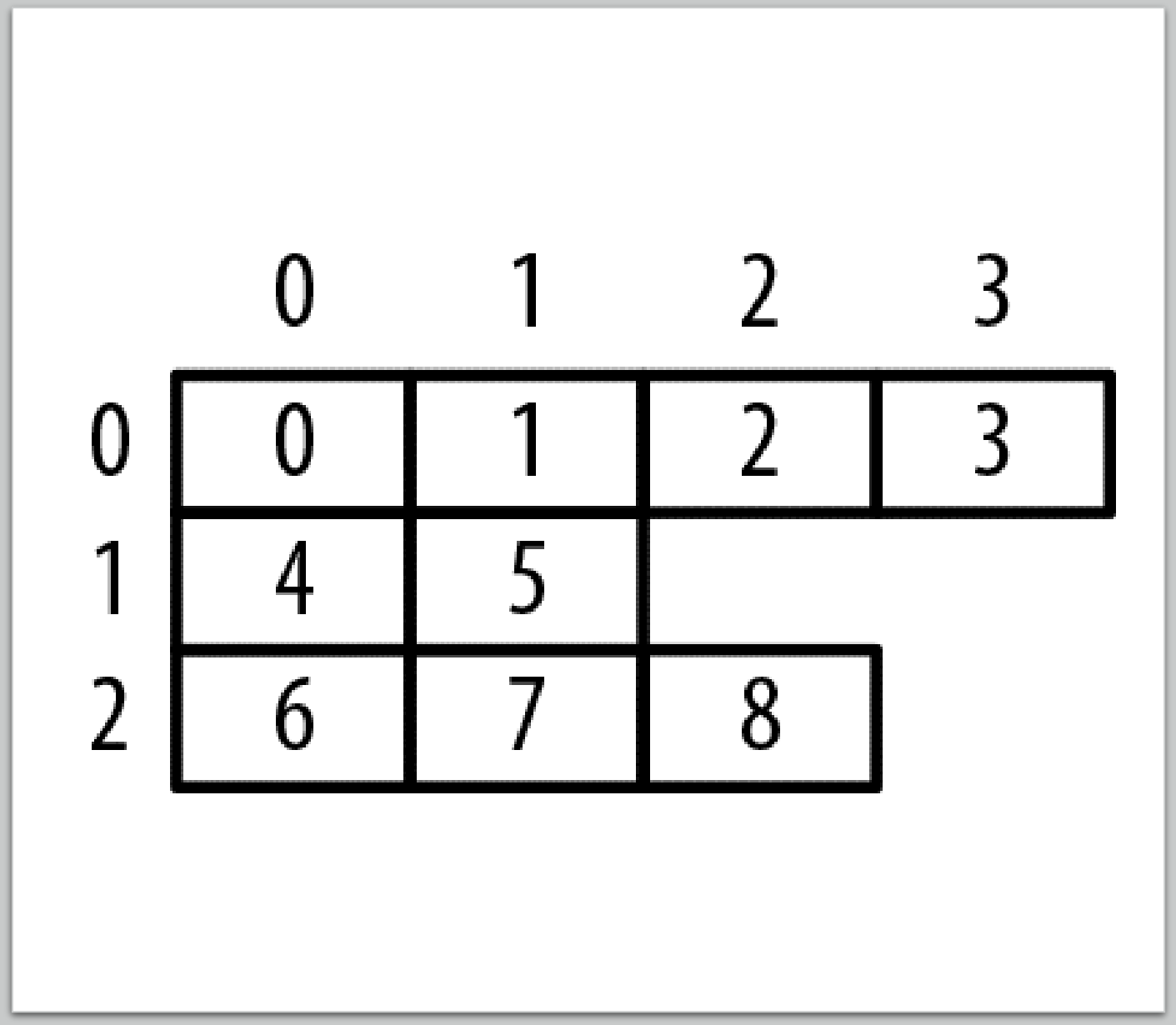
Çalışma Sorusu

15X15 boyutlu bir matris düşünerek yandaki rüzgargülünü oluşturan programı yazalım.

DÜZENSİZ DİZİLER (JAGGED ARRAYS)

- Elemanları dizi olan dizilere düzensiz dizi adı verilir.
- Düzensiz dizilerin her bir elemanı birbirinden bağımsız başka bir dizidir.
- Her bir eleman birbirinden bağımsız olduğu için, eleman sayısı da birbirinden farklı olabilir.

```
int[][] jagged = new int[3][];
jagged[0] = new int[4] { 0, 1, 2, 3 };
jagged[1] = new int[2] { 4, 5 };
jagged[2] = new int[3] { 6, 7, 8 };
```



System.Array Sınıfı

- Metotlar ve sınıflar konusuna ileride değineceğiz.

- Dizileri Kopyalama:
- CopyTo metodu ile bir dizinin tamamı başka bir dizinin istenilen yerine kopyalanabilir.
- Yandaki dizi1'in tüm elemanları dizi2 ye 3. indeksten itibaren kopyalanır.

```
int[] dizi1 = { 1, 2, 3, 4 };
int[] dizi2 = new int[10];
dizi1.CopyTo(dizi2, 3);

foreach (var eleman in dizi2)
    Console.WriteLine(eleman);
```

```
0
0
0
1
2
3
4
0
0
0
```

- Array sınıfının Copy metodu da kullanılabilir:
- Copy(Array dizi1,Array dizi2, int uzunluk)
- Uzunluk kadar eleman dizi1'den dizi2 ye kopyalanır. Kopyalama 0. indeksten başlar

```
int[] dizi1 = { 1, 2, 3, 4 };
int[] dizi2 = new int[10];

Array.Copy(dizi1,dizi2,2);

foreach (var eleman in dizi2)
    Console.WriteLine(eleman);
```

```
1
2
0
0
0
0
0
0
0
0
```

- ConstrainedCopy(Array dizi1, int x, Array dizi2,int y, int uzunluk)
- Dizi1 in x indeksinden sonraki uzunluk kadar eleman dizi2'nin y indeksinden sonrasına kopyalanır.

```
int[] dizi1 = { 1, 2, 3, 4 };
int[] dizi2 = new int[10];

Array.ConstrainedCopy( dizi1,1,dizi2,2,3);

foreach (var eleman in dizi2)
    Console.WriteLine(eleman);
```

```
0
0
2
3
4
0
0
0
0
0
```

Dizileri Sıralama

- Kendi algoritmalarımızı geliştirebileceğimiz gibi System.Array sınıfının statik Sort metodunu da kullanabiliriz.

- Sıralama yaparken Arayüz(interface) dediğimiz kavramlardan faydalanılır. Interface kavramına henüz değinmediğimiz için Sort metodunun basit sıralama örneğini yapacağız.

- **Örnek:** «Öğrencilerin isimlerinden oluşan bir diziyi alfabetik sıraya göre Sort metodunu kullanarak sıralayalım».

Dizilerde Arama

- Çeşitli algoritmalar kullanılabilir.

- **Array.BinarySearch(Array dizi,object nesne)**
  - Nesneyi arar eğer nesne bulunursa indeksini yoksa negatif bir sayı döndürür.
- **BinarySearch(Array dizi,int başlangıç,int uzunluk,object nesne)**
  - Baslangıc indeksinden itibaren uzunluk kadar eleman içerisinde arar. Bulunursa indeks bulunmazsa negatif sayı döndürür.

- *Binary Search sıralanmış diziler üzerinde arama yapar.*
- **Örnek:** «Sıralanmış bir dizide arama yapalım»

Diziyi Temizleme ve Ters Çevirme

- **Array.Clear()**
  - Belirli sayıdaki eleman sıfırlanır(varsayılan değere atanır: bool türü-> false, nümerik değer ->0, referans tür -> null).

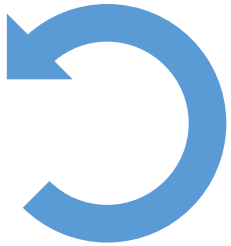
```
Array.Clear(dizi, 1, 3);
// 1. elemandan itibaren ilk 3 eleman sıfırlanır.
```

- **Array.Reverse()**
  - Dizinin tamamını ters çevirmek için

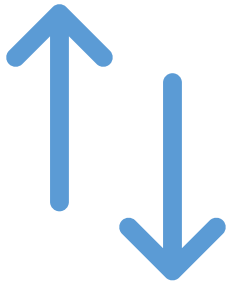
```
Array.Reverse(dizi);
Array.Reverse(dizi, 1, 3); //1.elemandan itibaren ilk 3 eleman ters çevrilir.
```

Örnek Sorular

Örnek 1: int türünden bir dizinin elemanlarını Sort() metodunu kullanmadan sıralayalım



Örnek2:Herhangi bir tür dizinin elemanlarını Reverse metodu kullanmadan terse çevirelim

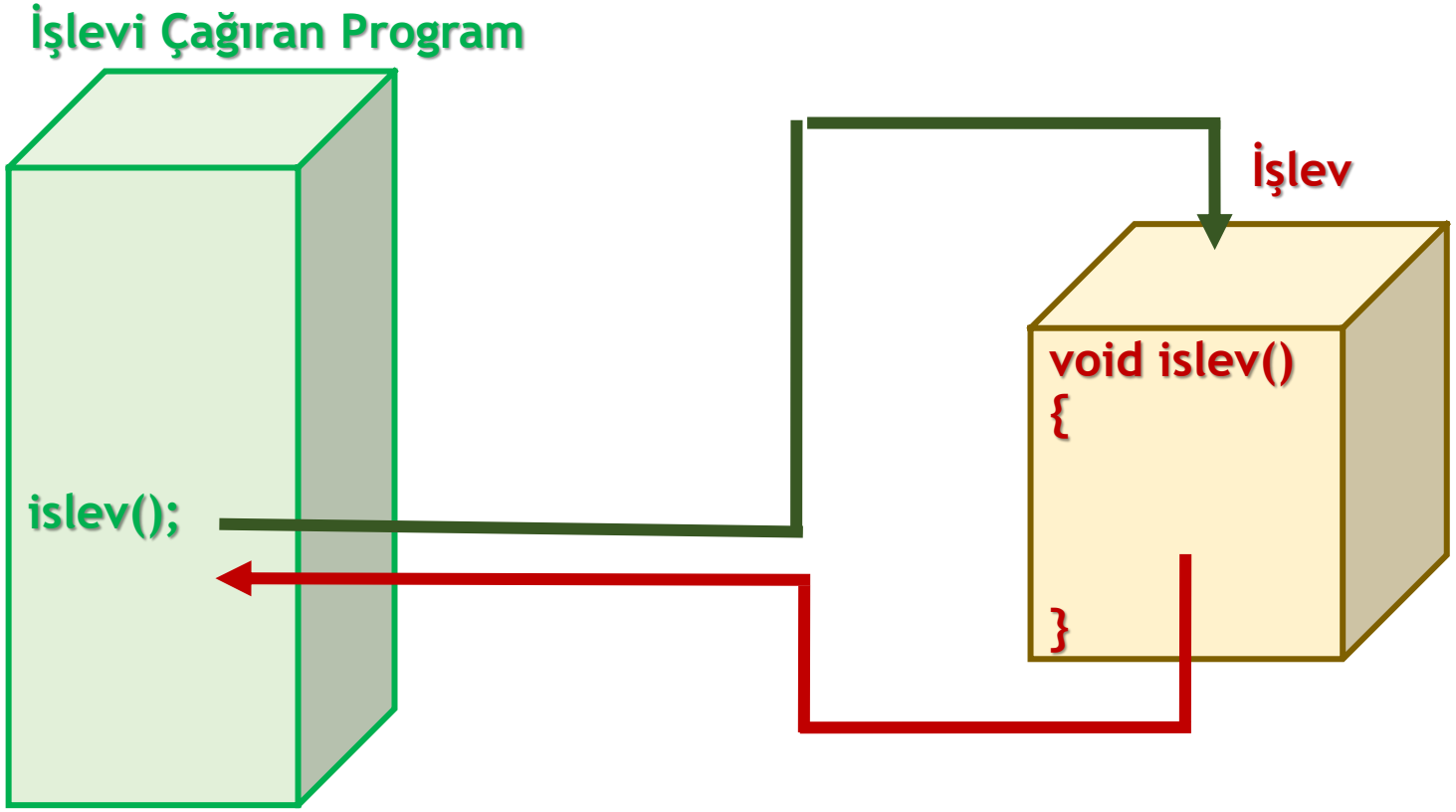


Örnek3: Elemanları 1-1000 arası rastgele değerler olan int türünden 20 elemanlı bir dizideki elemanların ortalamalarını, en büyük ve en küçük değeri veren programı yazalım

Metot ve Fonksiyonlar

- Programın herhangi bir yerinde bir işi yerine getirmek amacıyla tasarlanmış alt programlardır.
- Büyük problemler, küçük parçalara bölünerek kolay çözülür.
- Yazılımlar benzer modüller içerirler. Dolayısıyla program içerisinde tekrar eden işlemlerin her defasında yeniden yazılması engellenerek program geliştirme kolaylaştırılır.
- Hata ayıklama küçük ölçekte daha kolaydır.
- Küçük parçalara yoğunlaşmak kolaydır.
- Daha güvenli ve verimli kod üretimi sağlanır.
- Ayrıca program boyutları da nispeten küçülür.

Metot



Metot Tanımlama

```
[erişim belirteçleri] <Geri dönüş tipi> Metot Adi (parametre_listesi) {
    gerekli değişken tanımlamaları…;
    metot gövdesi…;
    return geriDonusDegeri / Degiskeni;
}
```

- Erişim belirteci yazılmazsa varsayılan olarak bildirilen sınıfa özel <private> olur.
- **Metot adı**
  - Metodun adını belirler. Değişken isimlendirme kuralları aynen geçerlidir.
- **Parametre listesi**
  - Metot içerisinde kullanılacak giriş parametreleri tanımlanır.
- **Geri Dönüş Değeri / Değişkeni**
  - Geriye dönen değer / değişken belirtilir.
- **Dönüş Tipi**
  - Metottan geri dönen değerin tipi yazılır. Geriye değer dönmeyecek ise **void** yazılmalıdır. void kullanımı durumunda fonksiyon return komut satırı içermez. Fonksiyonda geri dönüş değeri/değişkeni **return** komut satırı ile belirtilir.

Örnekler

```
float aylikMaas(float mesaiSaati, float saatUcreti)
```

```
float bol(int x, float f)
```

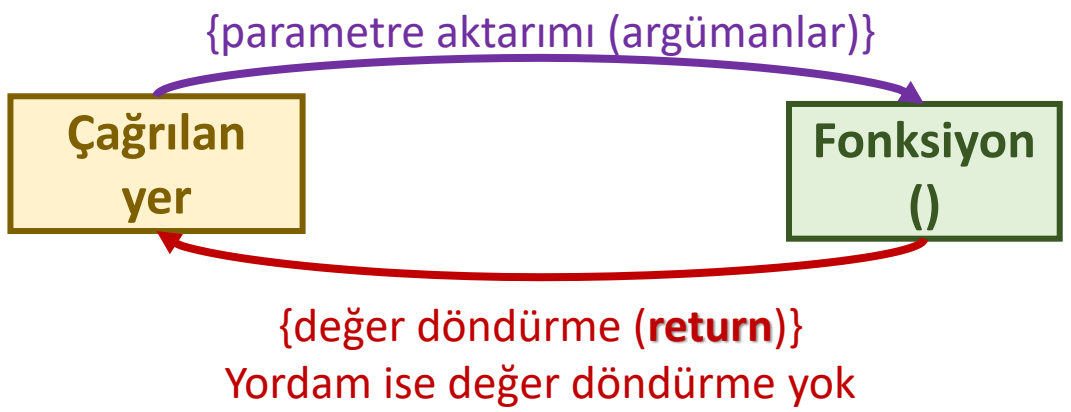
```
int kareAl(int x)
{

    return x*x;
}
```

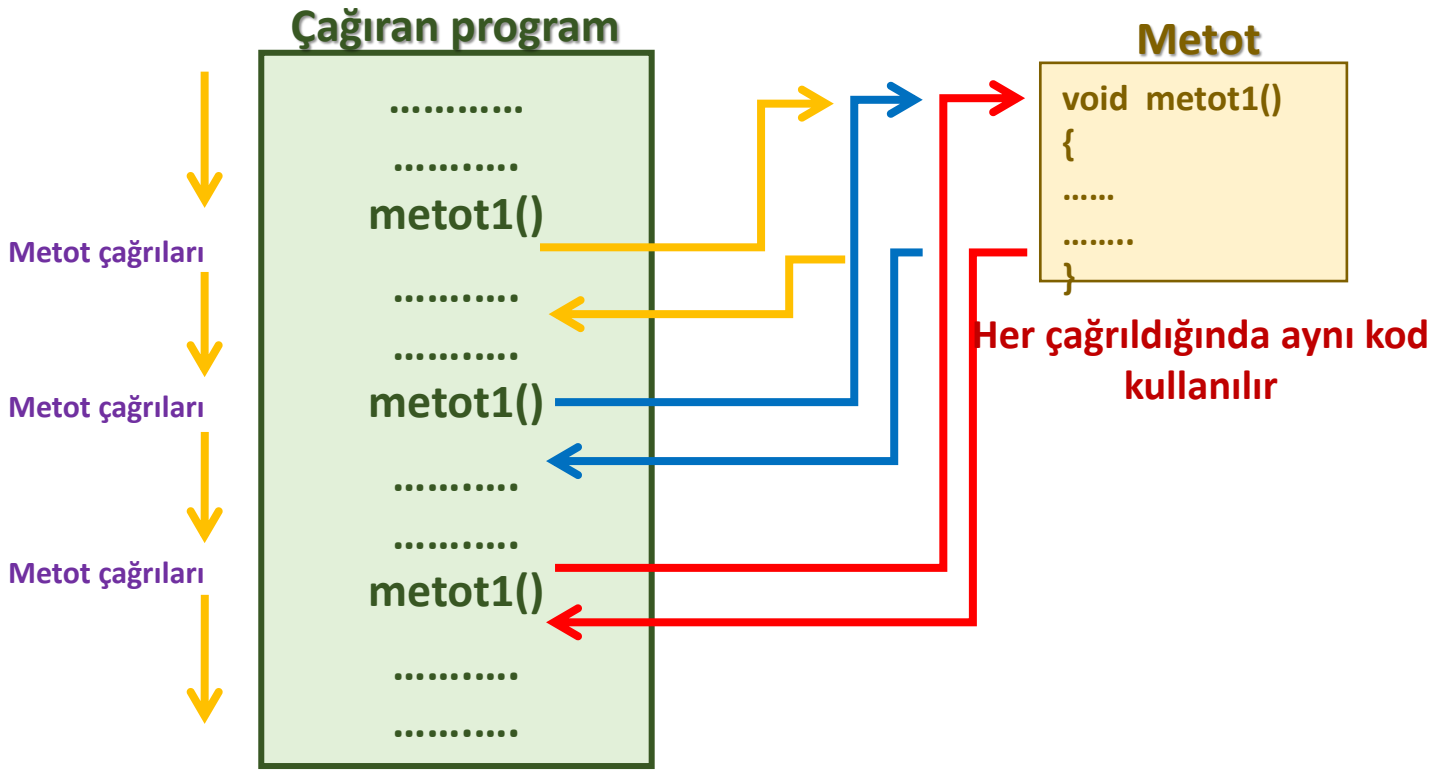
```
void mesaj(int mesajNo)
{
    switch (mesajNo)
    {
    case 0 : cout << "Mesaj 0"; break;
    case 1 : cout << "Mesaj 1"; break;
    default: cout << "Tanımsız..."; break;
    }
}
```

Önemli Özellikler

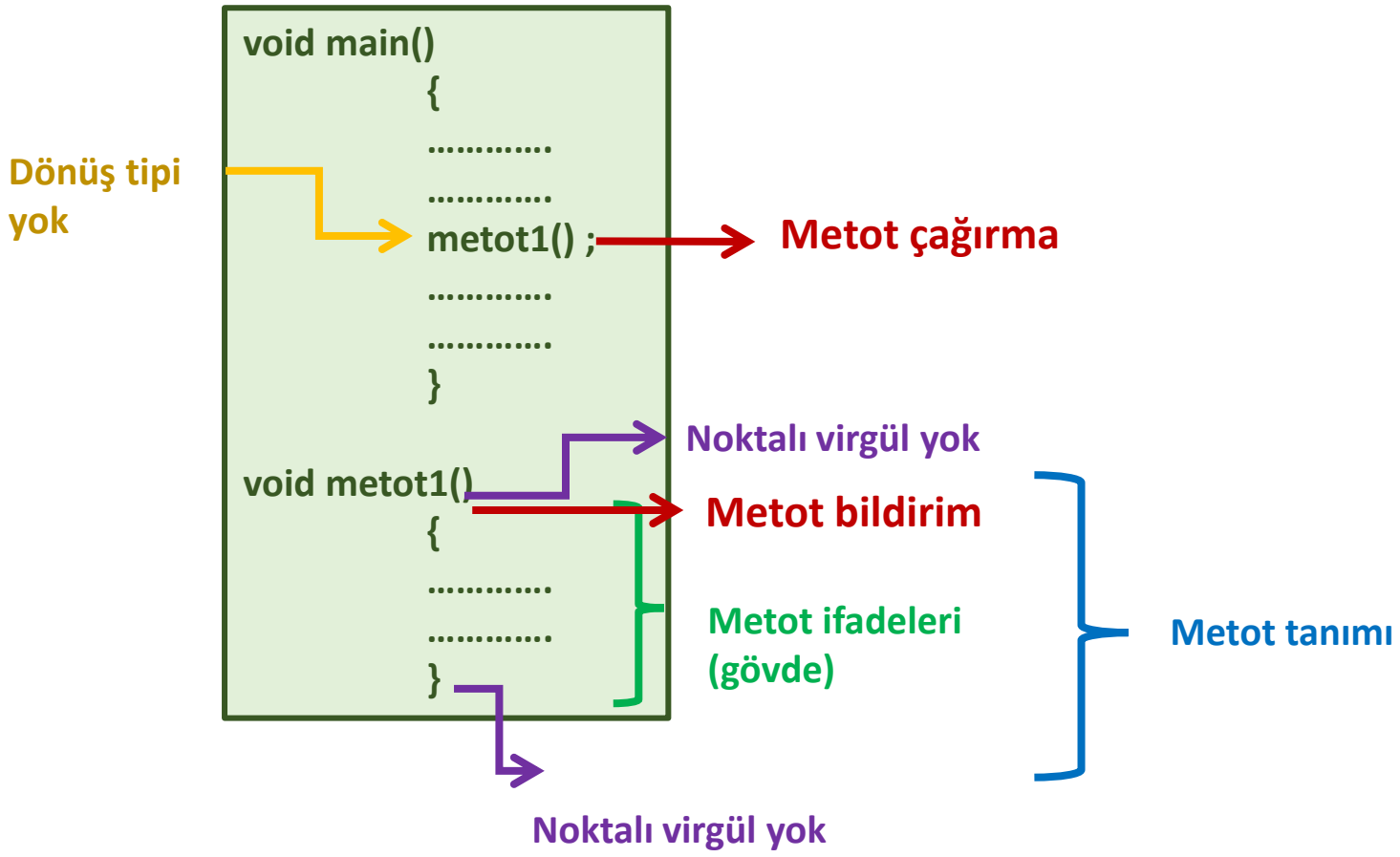
- Metotlara mümkün olduğu kadar az görev verilmelidir.
- Bir metot az ve öz iş yapmalıdır.
  - Örneğin bir metot hem diziyi sıralayıp hem de dizinin elemanlarını ekrana yazdırmamalıdır.
- Değişken isimlendirme kuralları aynen geçerlidir.
- Metot çağrılırken istenen parametreler eksiksiz girilmelidir.
- Geri dönüş değeri herhangi bir türden olabilir. Geri dönüş değeri olmayan metotlar **void** anahtar sözcüğü ile bildirilmelidir.
- Metotların geri dönüş değerini uyumlu bir türden nesneye atamak gerekir.
  - Örn. Geri dönüş değeri ***double*** ise ***int*** türünden bir nesneye dönüşüm yapılmadan atanamaz.
- **Void** metotlarda **return** anahtar sözcüğünü bir ifade ile birlikte kullanılamaz.
- Metot içerisinde başka bir metot bildirilemez.
- Metot içerisinde tanımlanan değişken metot dışında geçersizdir.

Metot kullanımı ve kontrol akışı



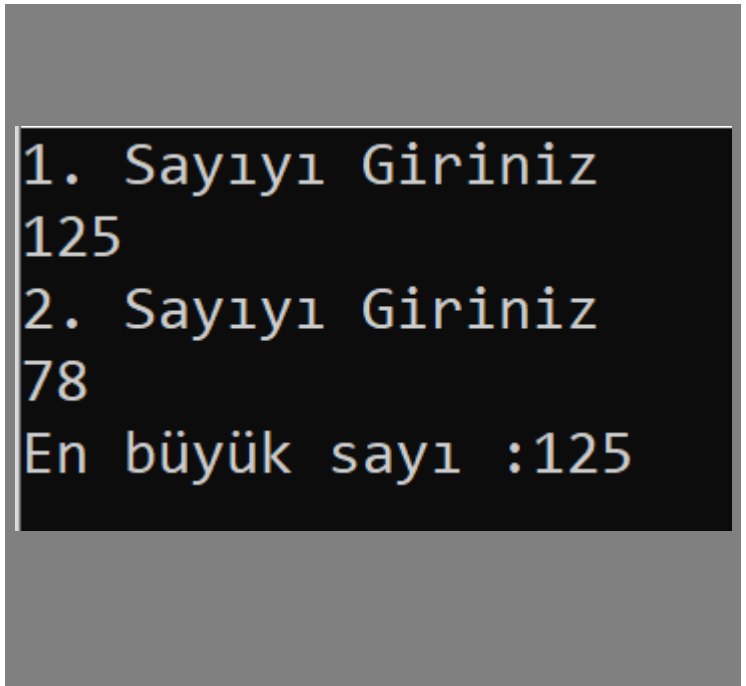


Metot kullanımı ve kontrol akışı



Metot-Örnek: Girilen 2 sayıdan büyük olanı bulma

Ekran




```
1 reference
static int EnBuyuk(int s1,int s2)
{
    if (s1 > s2)
        return s1;
    else
        return s2;
}
0 references
static void Main(string[] args)
{

    int sayi1, sayi2;
    //Sayıları kullanıcıdan alma
    Console.WriteLine("1. Sayıyı Giriniz");
    sayi1 = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("2. Sayıyı Giriniz");
    sayi2 = Convert.ToInt32(Console.ReadLine());

    //Metot çağırma
    int buyukSayi = EnBuyuk(sayi1, sayi2);

    Console.WriteLine("En büyük sayı :{0}", buyukSayi);


    Console.ReadLine();
}
```

Çalışma Sorusu

Girilen 3 sayıdan en büyüğünü bulan programı bir önceki slayttaki EnBuyuk metodunu kullanan yeni bir metot ile yazınız.

Değer ve Referans Parametreleri

- Referans türleri metotlara aktarılırken bit bit kopyalanmazlar, yani metoda aktarılan bir referanstır.
- Metot içerisinde bu referans ile direkt işlem yapmak nesnenin değerini değiştirebilir. Bu nedenle dikkatli olmak gerekir.
- C#'da bütün değer tipleri metotlara bit bit kopyalanarak geçirilir.
  - Örn int türünden bir nesne metoda parametre olarak aktarıldığında metot içerisinde bu değer değiştirilse bile ana değişken değişmez.
- Bir referans türü olan örneğin dizileri metoda parametre olarak aktarır isek dizinin bütün elemanları metoda parametre olarak kopyalanmaz. Parametre olarak kopalanan sadece ilgili dizinin referansıdır. Metot içerisinde referansla ilgili değişiklik yaptığımızda orijinal nesneyi de değiştirmiş oluruz.
- C# varsayılan olarak bütün değer tipleri metotlara değer olarak; referans tipleri ise referans olarak aktarılır.
- Bazı durumlarda değer tipleri de referans olarak aktarılmak istenebilir.(Bknz: **ref** ve **out** anahtar sözcükleri)

Değer ile Aktarım
Referans ile Aktarım

```
1 reference
static void DegerTipiAktarma(int deger)
{
    deger = 54;
}
1 reference
static void ReferansTipiAktarma(int[] deger)
{
    deger[0] = 100;
}
0 references
static void Main(string[] args)
{
    //Değer tipi aktarım
    Console.WriteLine("Değer ile Aktarma");
    Console.WriteLine("---------------");
    int not = 49;
    Console.WriteLine(not);
    DegerTipiAktarma(not);
    Console.WriteLine(not);

    //Referans tipi aktarım
    Console.WriteLine("\nReferans ile Aktarma");
    Console.WriteLine("---------------");
    int[] notlar = { 15, 35, 55, 90 };
    Console.WriteLine(notlar[0]);
    ReferansTipiAktarma(notlar);
    Console.WriteLine(notlar[0]);

    Console.ReadLine();
}
```

```
Değer ile Aktarma
---------------
49
49

Referans ile Aktarma
---------------
15
100
```

Metot Aşırı Yükleme ve İmza Kavramı

- **Metot Aşırı Yükleme**
  - Aynı isme sahip fonksiyonun aldığı parametrelere göre farklı şekilde çalışmasıdır.

```
float aylikMaas(float mesaiSaati, float saatUcreti)
```

Metot imzası

```
float aylikMaas(float mesai, float saatUcreti);
float aylikMaas(float maas);
float aylikMaas();
```

- Metotlar aşırı yüklenerek aynı isimde birden fazla metot oluşturulabilir.
- İsimler aynı olduğuna göre hangi metodu çağırdığımız nasıl anlaşılacaktır?
  - ***Metot İmzası (method signature)***

- **Metot imzası;** bir metodun karakteristik özelliklerini içeren bir bilgidir. Metodun adı, parametrelerinin sayısı ve türleri ile alakalıdır. **Geri dönüş değeri metodun imzasına dahil değildir.**

Metot-Örnek:
Aşırı Yükleme

```
1 reference
static void Metot1(int x, int y)
{
    Console.WriteLine("1. metot çağrıldı.");
}
1 reference
static void Metot1(float x, float y)
{
    Console.WriteLine("2. metot çağrıldı.");
}
1 reference
static void Metot1(string x, string y)
{
    Console.WriteLine("3. metot çağrıldı.");
}
0 references
static void Main()
{
    Metot1("deneme", "deneme");
    Metot1(5, 6);
    Metot1(10f, 56f);

    Console.ReadLine();
}
```

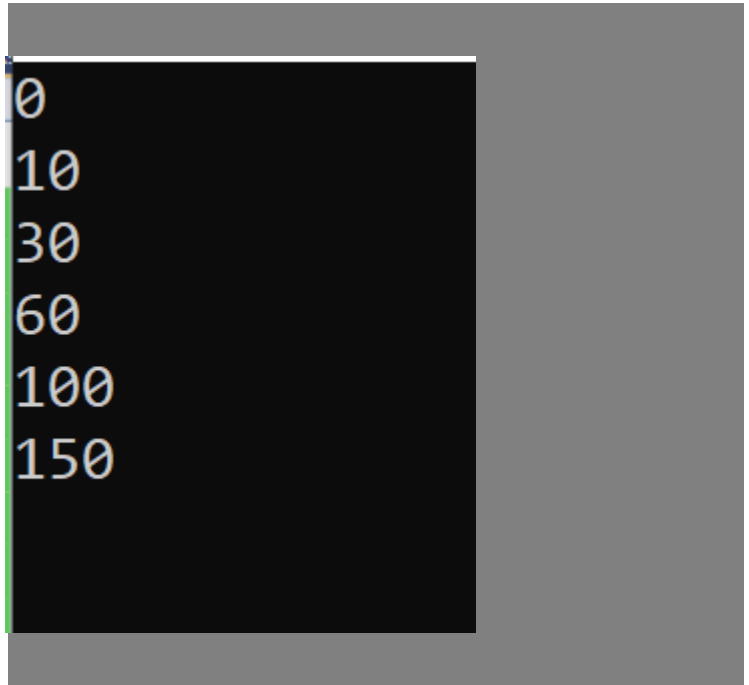
```
4 references
private static int EnBuyuk(int sayi1, int sayi2)
{
    if (sayi1 > sayi2)
        return sayi1;
    else
        return sayi2;
}
2 references
private static int EnBuyuk(int sayi1, int sayi2, int sayi3)
{
    int buyuk = EnBuyuk(sayi1, sayi2);
    return EnBuyuk(buyuk, sayi3);
}
1 reference
private static int EnBuyuk(int sayi1, int sayi2, int sayi3, int sayi4)
{
    int buyuk = EnBuyuk(sayi1, sayi2, sayi3);
    return EnBuyuk(buyuk, sayi4);
}
```

Değişken Sayıda Parametre Alan Metotlar

- Bazı durumlarda metotlara göndereceğimiz parametrelerin sayısı belli olmayabilir.
- ***params*** anahtar sözcüğü kullanılır.
- Params anahtar sözcüğü değişken sayıda eleman içerebilen bir veri yapısı tanımlar. Metoda gelen her bir parametre bu dizinin bir elemanı olacak şekilde ayarlanır.

Metot-Ornek: Değişken sayıda parametre alabilen Topla() metodu

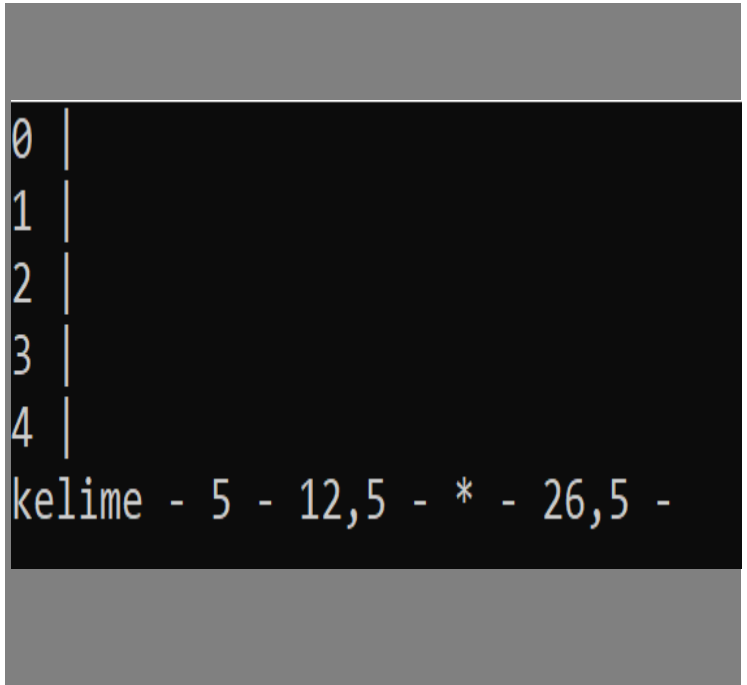
Ekran




```
6 references
static int Toplam(params int[] sayilar)
{
    if (sayilar.Length == 0)
        return 0;
    int toplam = 0;
    foreach (int sayi in sayilar)
        toplam += sayi;
    return toplam;

}
0 references
static void Main()
{
    Console.WriteLine(Toplam());
    Console.WriteLine(Toplam(10));
    Console.WriteLine(Toplam(10,20));
    Console.WriteLine(Toplam(10,20,30));
    Console.WriteLine(Toplam(10,20,30,40));
    Console.WriteLine(Toplam(10, 20, 30, 40,50));


    Console.ReadLine();
}
```

Metot-Ornek: Değişken sayıda parametre alabilen Ciz() metodu.

Ekran




```
2 references
static void Ciz(int yon,params object[] nesne)
{
    if (nesne.Length == 0)
        Console.WriteLine("Sembol girilmedi");
    else
    {
        if (yon == 0)
            foreach (object o in nesne)
                Console.Write(o.ToString() + " - ");

        else
            foreach (object o in nesne)
                Console.WriteLine(o.ToString()+" | ");
    }
}
0 references
static void Main()
{
    Ciz(1, 0, 1, 2, 3, 4);
    Ciz(0, "kelime", 5, 12.5f,'*', 26.5);



    Console.ReadLine();
}
```

!

- Console.WriteLine() metodunun da istenilen sayıda parametre alabildiğine dikkat ediniz.

```
Console.WriteLine("{0} - {1} - {2} - {3}", "C#", "Programlama", 2020, "Bahar");
```

```
C# - Programlama - 2020 - Bahar
```

Opsiyonel(Optional) Parametreler

- Metot bildirimi sırasında istenilen parametrelere varsayılan değerler verilerek ilgili parametrenin gönderilmediği durumlarda metot içinde alacağı değer belirtilmektedir.

```
2 references
static void FiyatYaz(int gerekliFiyat, string opsiyonelBirim= "TL")
{
    Console.WriteLine("Ödenmesi gereken: {0} {1} ", gerekliFiyat, opsiyonelBirim);
}
0 references
static void Main()
{

    FiyatYaz(20);
    FiyatYaz(5, "Dolar");



    Console.ReadLine();
}
```

```
Ödenmesi gereken: 20 TL
Ödenmesi gereken: 5 Dolar
```

İsimlendirilmiş(Named) Parametreler

```
static void Main()
{
    //Metod parametrelerin tanımlandığı sırada normal olarak çağrılabilir.
    SiparisDetay("Kotan Cafe", 25, "Filtre Kahve");

    //Named argümanlar istenilen sırada parametrelere gönderilebilir.
    SiparisDetay(siparisNo:25,urunAdi:"Filtre Kahve",satici:"Kotan Cafe");
    SiparisDetay(urunAdi: "Filtre Kahve", satici: "Kotan Cafe",siparisNo: 25);

    //Doğru sıralamada kullanılırsa named argümanlar named olmayan argümanlar ile
    //birlikte kullanılabilir.
    SiparisDetay("Kotan Cafe", 25, urunAdi:"Filtre Kahve");
    SiparisDetay(satici:"Kotan Cafe", 25, urunAdi: "Filtre Kahve");

    //Sıralama dışında kullanılırsa karmaşık argümanlar geçersiz olur.
    //Aşağıdaki örnekler derleyici hatasına neden olacaktır.
    SiparisDetay(urunAdi:"Filtre Kahve",25,"Kotan Cafe");
    SiparisDetay(25,satici:"Kotan Cafe","Filtre Kahve");


    Console.ReadLine();
}

7 references
static void SiparisDetay(string satici, int siparisNo, string urunAdi)
{
    if (string.IsNullOrWhiteSpace(satici))
    {
        Console.WriteLine("Satıcı adı girilmelidir");
        return;
    }

    Console.WriteLine($"Satıcı: {satici}, Sipariş #: {siparisNo}, Ürün: {urunAdi}");
}
```

- Metot çağrımında parametre sırasından bağımsız olarak parametre adı belirtilerek parametreler metoda gönderilebilir.

```
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
```

Özyineli(Recursive) Metotlar

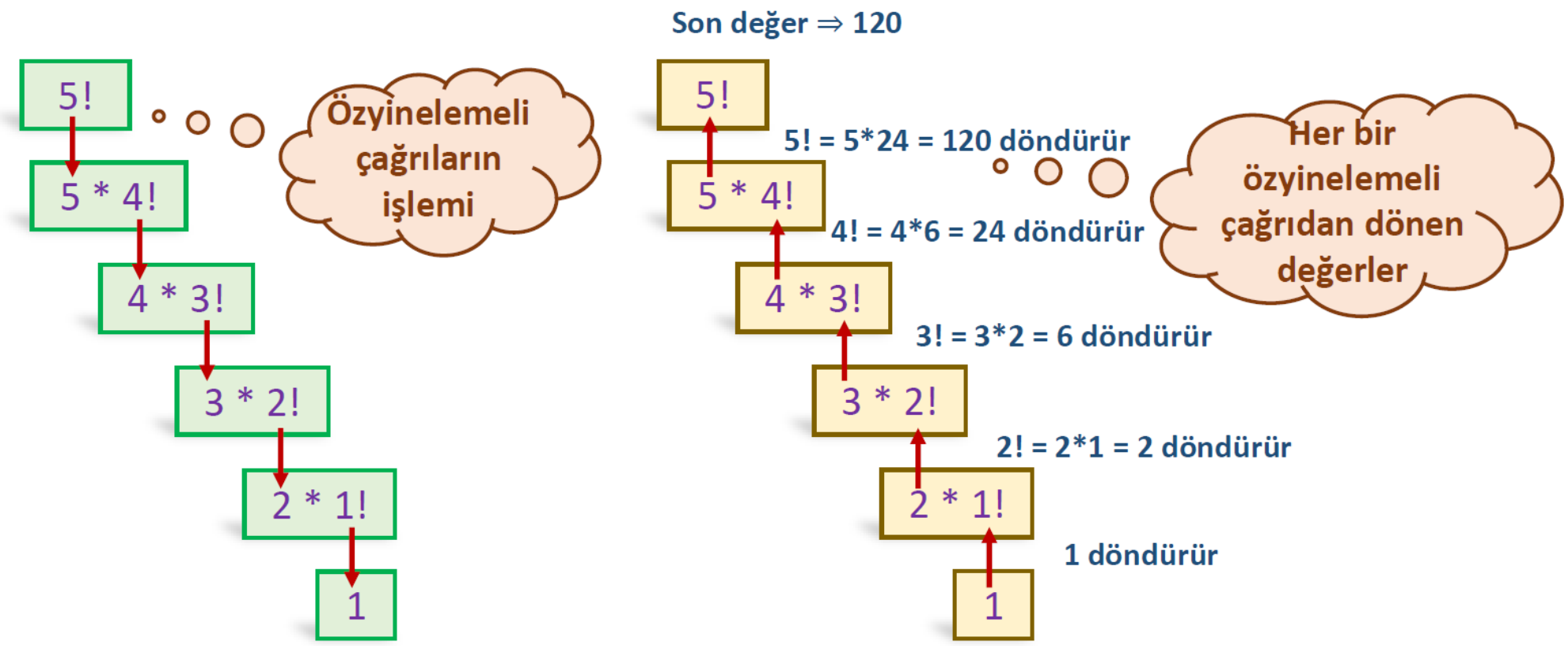
- Metodun kendi kendini çağırmasıdır.
- *Klasik bir örnek: Faktöriyel Hesaplama*

```
5 references
static int faktoriyel(int sayi)
{
    if (sayi == 0)
        return 1;

    return sayi * faktoriyel(sayi - 1);
}

0 references
static void Main()
{

    Console.WriteLine(faktoriyel(0));
    Console.WriteLine(faktoriyel(1));
    Console.WriteLine(faktoriyel(5));
    Console.WriteLine(faktoriyel(8));

    Console.ReadLine();
}
```

```
1
1
120
40320
```

Özyineli Metotlar

- Faktöriyel hesabı ⟹ **n!=n*(n-1)!**



System.Math Sınıfı ve Metotları

- Belirli matematiksel işlemleri yapmak için Sistem isim alanında bulunan Math sınıfı kullanılabilir.
- Math sınıfı metotları statik olarak tanımlandığı için herhangi bir nesne tanımlaması yapmadan kullanabiliriz.

System.Math

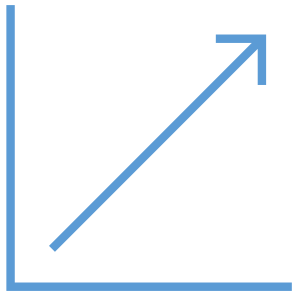
Kullanım	Açıklama
Math.PI	pi=3,14159265358979
Math.E	e=2,71828182845905
Abs(x)	Mutlak değer alma
Cos(x)	Kosinüs alma
Sin(x)	Sinüs alma
Tan(x)	Tanjant alma
Ceiling(x)	X sayısını x'ten büyük en küçük tamsayıya yuvarla
Floor(x)	X sayısını x'ten küçük en büyük tamsayıya yuvarla
Max(x,y)	X ve y sayısından en büyüğü
Min(x,y)	X ve y sayısından en küçüğü
Pow(x,y)	X üzeri y hesapla
Sqrt(x)	Karekök hesapla
Log(x)	X sayısının e tabanında logaritması
Exp(x)	E üzeri x hesapla
Log10(x)	X sayısının 10 tabanında logaritması

• • •

Çalışma Sorusu

Bir önceki slayttaki tüm Math sınıfı metotları için örnek yapalım.

Örnek Sorular



Örnek 1:Değişken sayıda parametre alan ve kendisine gönderilen bütün sayıların karesini hesaplayıp bir diziye aktararak bu dizinin referansını geri döndüren bir metot yazınız.

Örnek 2: Fibonacci sayı dizisini kendisine gönderilen değer için hesaplayan bir programı özyineli(recursive) metot kullanarak yazınız. 0'dan 10'a kadar olan sayıların Fibonacci değerlerini ekrana yazdırınız.

Fibonacci sayı dizisi: 0 1 1 2 3 5 8 13 21 34 …

Referanslar

- Sefer Algan her yönüyle C#
- BTK Akademi C#-Engin Demiroğ
- https://www.c-sharpcorner.com/blogs/data-types-in-c-sharp
- Ü.Kocabıçak,C.Öz,N.Taşbaşı,S.İlyas Ders Notları
- Images:
- https://tr.wikipedia.org/wiki/C_Sharp