PROGRAMLAMA DİLLERİ I

Öğr. Gör. M. Mutlu YAPICI

Ankara Üniversitesi
Elmadağ Meslek Yüksekokulu

Öğr. Gör. M. Mutlu YAPICI

Ders İzlencesi

Hafta	Modüller/İçerik/Konular
1. Hafta	
2. Hafta	
3. Hafta	
4. Hafta	
5. Hafta	
6. Hafta	
7. Hafta	
8. Hafta	
9. Hafta	
10. Hafta	
11. Hafta	
12. Hafta	
13. Hafta	
14. Hafta	

Algoritma Örnekleri

Beş sayının toplamını ve ortalamasını veren programa ait algoritma

Çözüm:

Toplam adı için Top
Ortalama adı için Ort
Girilen sayılar için X
Arttırma için Sayac kullanılırsa

A1 :Top = 0, Sayac = 0
A2 :X değerini gir.
A3 :Top= Top+X
A4 : Sayac = Sayac +1
A5 :Eğer Sayac <5 ise A2'ye git.
A6 :Ort= Top/5
A7 :Top ve Ort değerlerini yaz.
A8 :Bitir

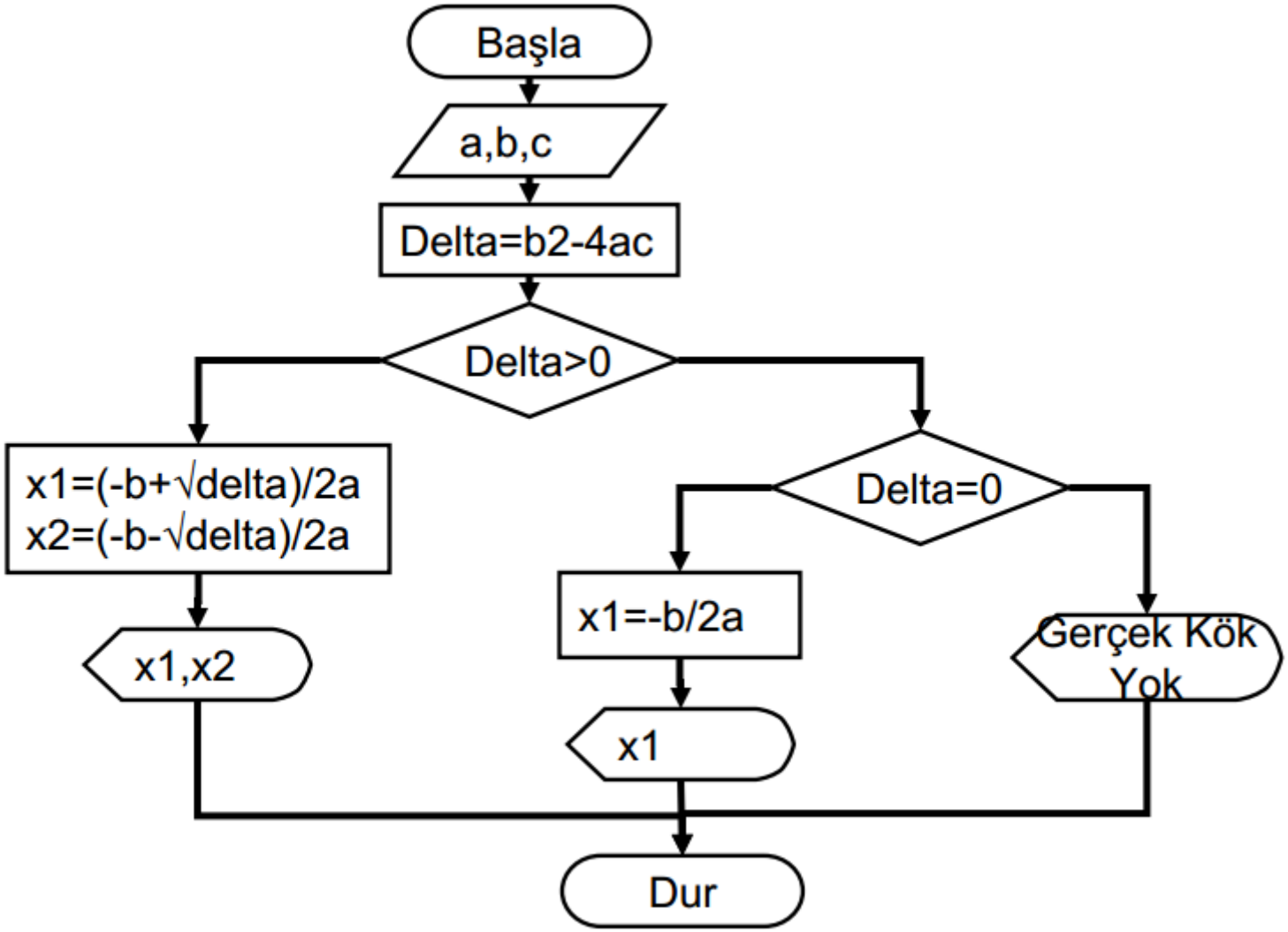
Algoritma Örnekleri

Örnek2: $ax^2+ bx + c = 0$ şeklindeki ikinci dereceden bir denklemin köklerini bulan algoritmayı tasarlayıp akış şeması ile gösteriniz.

Çözüm:

A1 : Başla
A2 : Sayısal a, b, c,delta,x1,x2
A3 : Yaz; “Denklem için a, b ,c değerlerini sırayla giriniz”
A4 : Oku; a
A5 : Oku; b
A6 : Oku; c
A7: delta=b^2-4ac
A8: Eğer delta > 0 ise x1=(-b+kök(delta))/2a ve x1=(-b-kök(delta))/2a
A9: Eğer delta == 0 ise x1=x2=-b/2a
A10: Eğer delta < 0 gerçek kökü yoktur.
A11 : Yaz x1 , x2;
A9 : Bitir

Algoritma Örnekleri



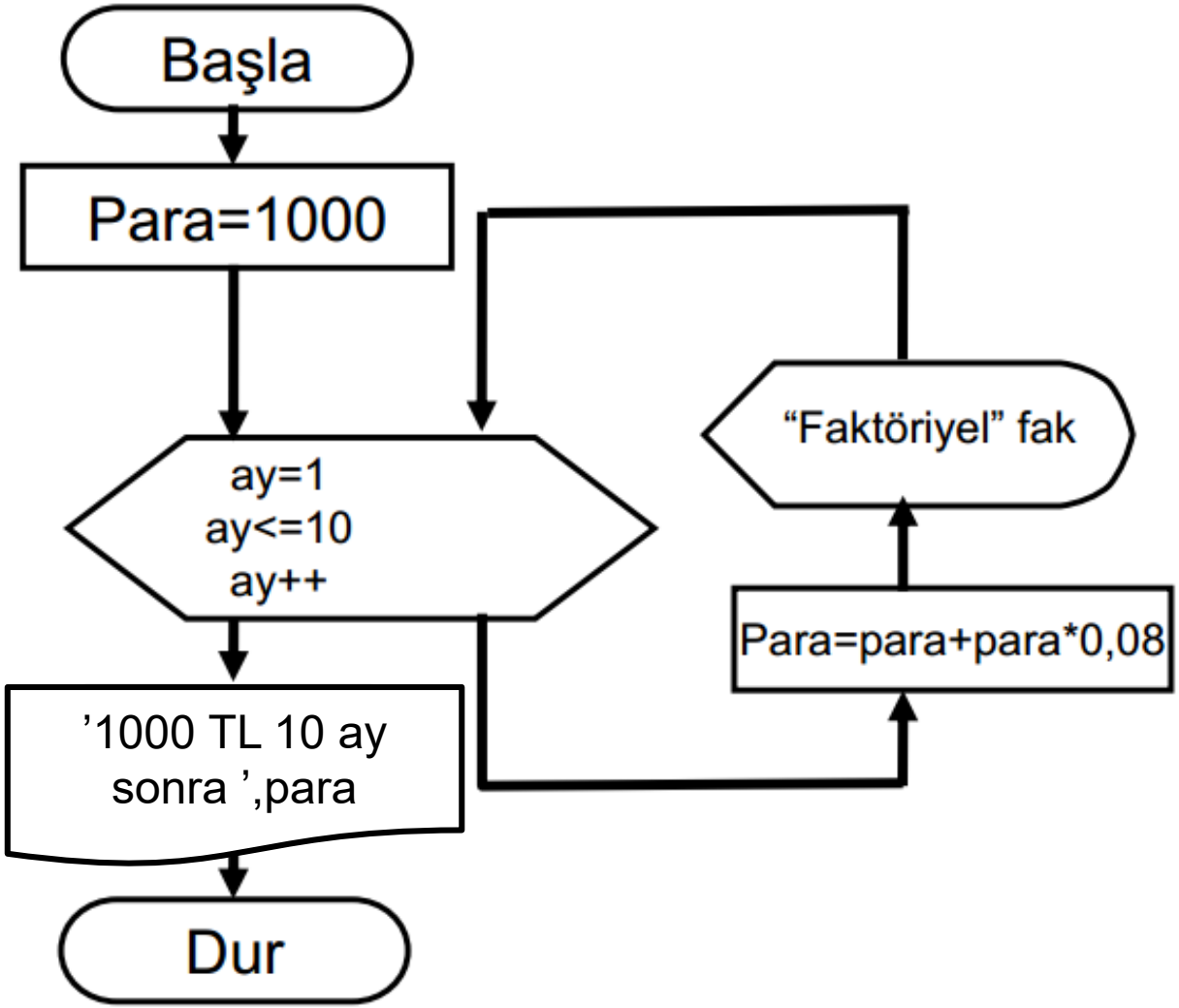
Algoritma Örnekleri

Örnek4: Bankaya aylık getirisi %8 ile 1000 TL para yatırılmıştır. Buna göre; 10 ay sonrası için paranın ne kadar olacağını ve bu süre içerisinde her ay sonunda ne kadar olacağını hesaplayıp ekrana yazan programın akış şemasını nasıldır?

Çözüm:

A1 : Başla
A2 : Sayısal para, aySayac
A3 : Yaz; “Parayı girin”
A4 : Oku; a
A5 : aySayac=1
A6: para=0
A7: para=para+para*0.8
A8: yaz:“Bu ay sonu toplam paray”,&para
A9:aySayac=aySayac+1
A10 : Eğer aySayac<=10 iken A6 ya git
A11 : yaz: :“10. ay sonu toplam paray”,&para
A12 : Bitir

Akış Diyagramı



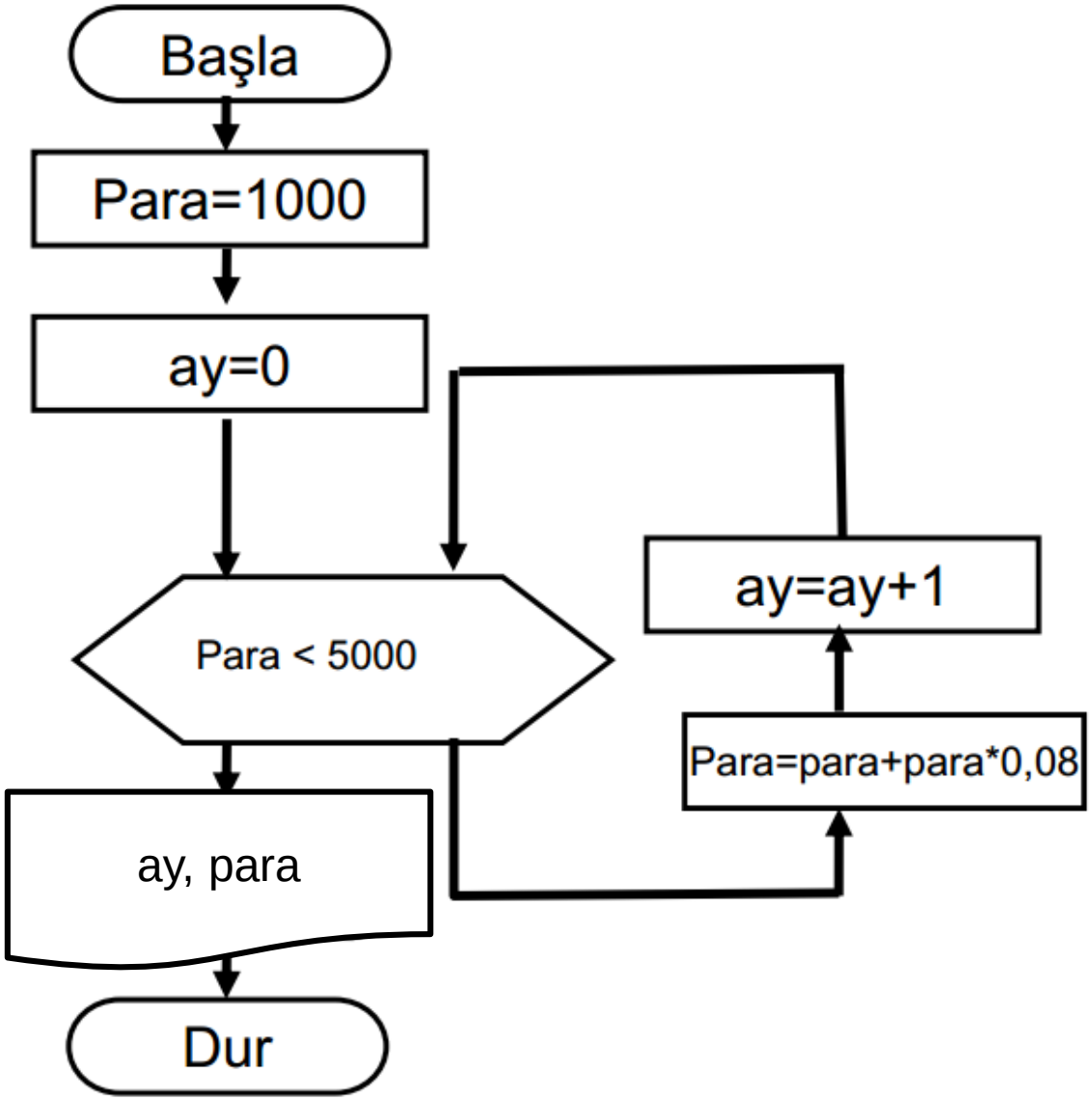
Algoritma Örnekleri

Örnek4: Bankaya aylık getirisi %8 ile 1000 TL para yatırılmıştır. Buna göre; Paranın kaç ay sonra 5000 TL olacağını hesaplayan programın akış şeması nasıldır?

Çözüm:

A1 : Başla
A2 : Sayısal para, aySayac
A3 : Yaz; “Parayı girin”
A4 : Oku; para
A5 : aySayac=1
A6: para=0
A7: para=para+para*0.8
A8:aySayac=aySayac+1
A9 : para<=5000 iken A6 ya git
A10 : yaz: “Paranız ”,&aySayac, “Ay sonra 5000 den fazla olur”
A11 : Bitir

Akış Diyagramı



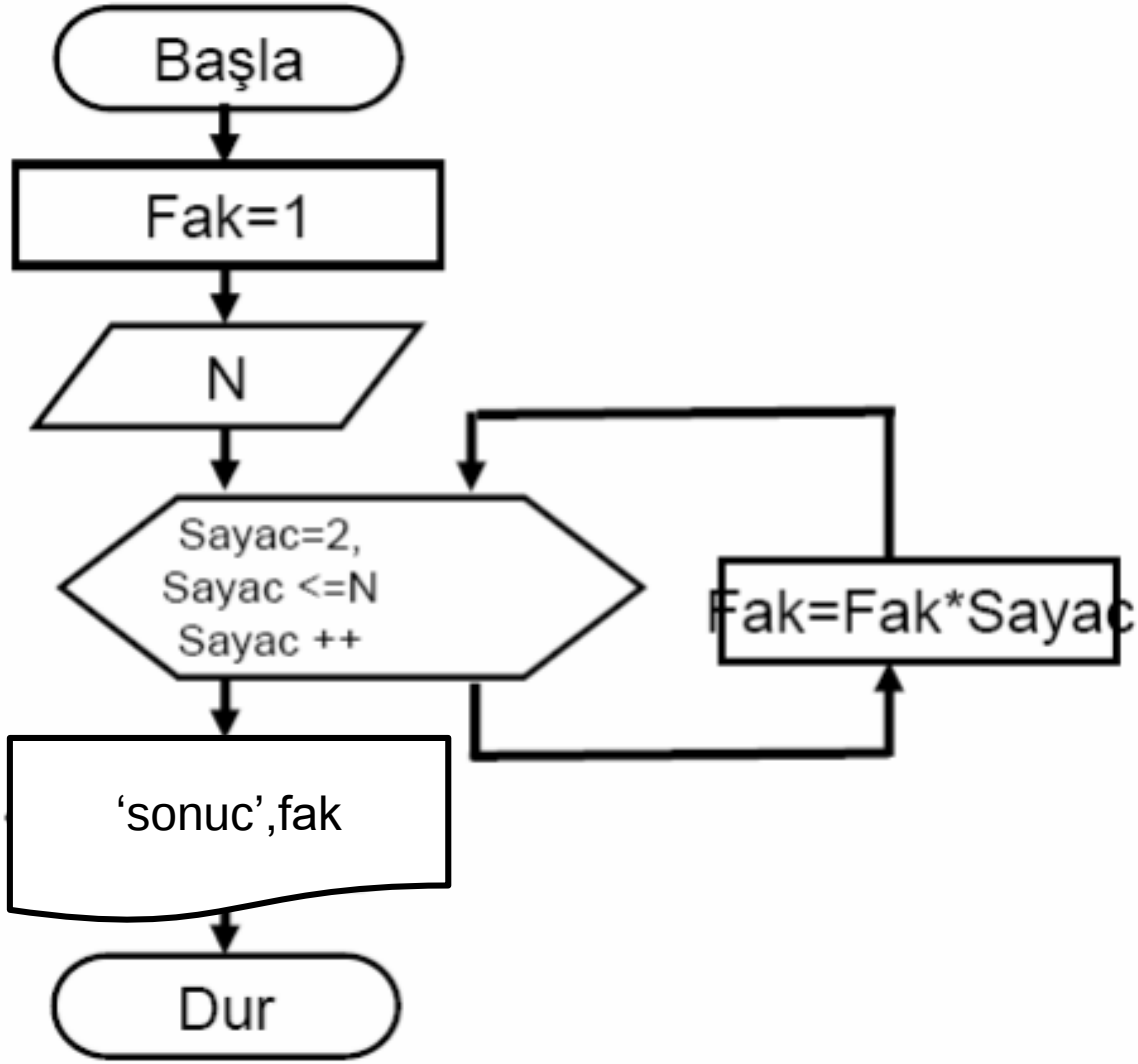
Algoritma Örnekleri

Örnek5: Klavyeden girilen N sayısının faktöriyelini alan algoritmasını yazıp ve akış diyagramını çiziniz?

Çözüm:

A1 : Başla
A2 : Sayısal n, fak,sayac
A3 : Yaz; “Sayıyı girin”
A4 : Oku; n
A5 : sayac=1
A6: fak=0
A7: fak=fak*sayac
A8: sayac=sayac+1
A9 : eğer sayac<=n iken A7 ye git
A10 : yaz: “sonuc ”,&fak
A11 : Bitir

Akış Diyagramı



Algoritma Örnekleri

Örnek6: Klavyeden vize ve final notu girilen öğrencinin not ortalamasını vizenin%40 ını finalin %60 ını alarak hesaplayan programın algoritmasını yazıp ve akış diyagramını çiziniz?

Çözüm:

A1 : Başla
A2 : Sayısal vz, fn,not
A3 : Yaz; “vize notunu girin”
A4 : Oku; vz
A5 : Yaz; “Final notunu girin”
A6 : Oku; fn
A7 : not=fn*0,6+vz*0,4
A8 : yaz: “notunuz”,&not
A9 : Bitir

Algoritma Örnekleri

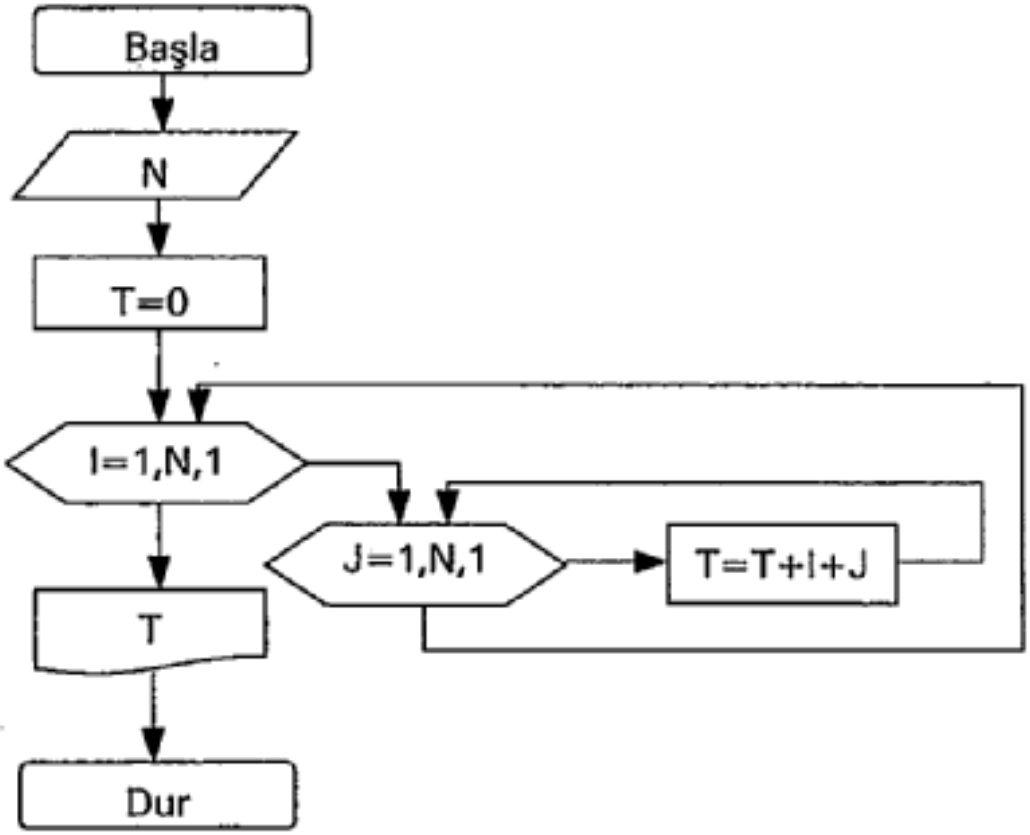
Örnek6: Klavyeden vize ve final notu girilen öğrencinin not ortalamasını vizenin%40 ını finalin %60 ını alarak hesaplayan ve notu 0-24 arası ise "Not Sıfır", 25-44 arası ise "Not Birdir", 45-54 arası ise "Not İkidir", 56-69 arası ise "Not Üçtür", 70-84 arası ise "Not Dörttür", 85-100 arası ise "Not Beştir" programın algoritmasını yazıp ve akış diyagramını çiziniz?

Çözüm:

A1 : Başla
A2 : Sayısal vz, fn,not
A3 : Yaz; "vize notunu girin"
A4 : Oku; vz
A5 : Yaz; "Final notunu girin"
A6 : Oku; fn
A7 : not=fn*0,6+vz*0,4
A7: if ….. İf…..if…..if…
A8 : yaz: "notunuz",&not
A9 : Bitir

Algoritma Örnekleri

Örnek7:Aşağıdaki akış diyagramının ne iş yaptığını, el ile çalıştırarak bulunuz?



Algoritma Örnekleri

Örnek8: 2-16 arasındaki çift sayıların ortalamasını bulan programın algoritma ve akış şemasını yazınız?

Çözüm:

Algoritma Örnekleri

Örnek9: Klavyeden girilen 10 tabanındaki sayıyı ikilik tabana çeviren programın algoritmasını ve akış şemasını yazınız?

Çözüm:

Algoritma Örnekleri

Örnek10: 10 ile 3000 arasındaki sayılardan 3 ve 10 ile tam bölünenlerin toplamını bulan algoritmayı oluşturunuz?

Çözüm:

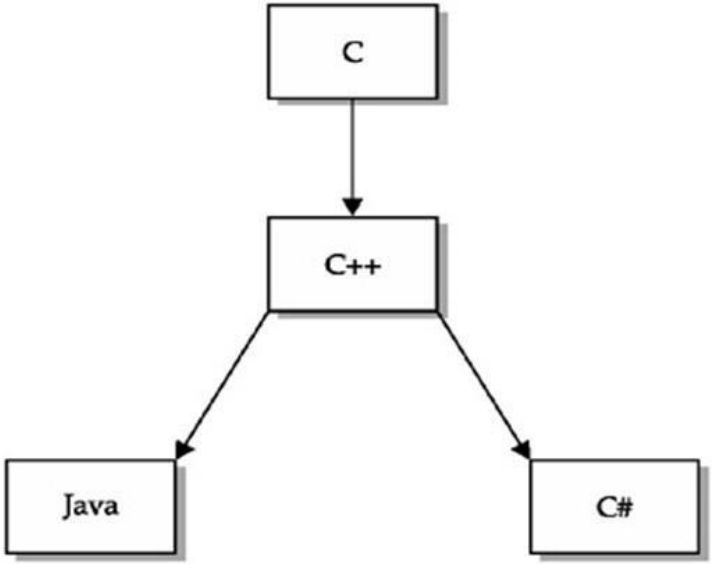
C# Programlama

- Bütün bilgisayar dilleri birbirleri ile ilişkilidir. Programlama dillerinde öncelikle C programlama dili ortaya çıkmıştır. C dili 1960'lı yıllarda **yapısal programlama (structured programming)** ile ortaya çıkmıştır. Yapısal programlama, düzgün kontrol ifadelerinin yer aldığı, bu ifadelerin hiyerarşik bir şekilde birbirini izlemesi şeklindeki bir programlama türüdür. C dili çok büyük projeleri yönetme açısından yetersiz kalan bir dildir ve bu problem yazılımcıları yeni diller aramaya yöneltmiştir.
- Bu problemi çözmek amacı ile **nesne yönelimli programlama(OOP-object orinted programming)** adı verilen yeni bir programlama yöntemi ortaya çıktı. Nesne yönelimli programlamada büyük projelerin yönetimine olanak sağlayan yapılar bulunmaktaydı. C dili bu tür programlamayı desteklemiyordu bu yüzden C++ dili **Bell** laboratuarında geliştirildi ve C dilinin bütün özelliklerini içermektedir.

C# Programlama

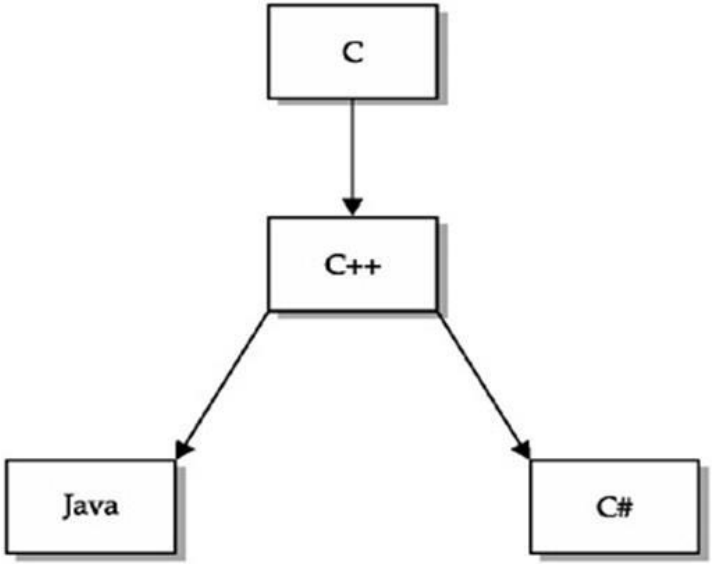
- 1990 'lı yılların sonlarına doğru Microsoft 'un programlama dehası Anders Hejlsberg **C#** dilini yarattı. 2000 yılının ortalarında **C#** 'ın ilk alfa versiyonu piyasaya çıktı. **C#** programlama dili ; C, C++ ve Java ile bağlantılı bir dil olarak yapıldı. Bunun bir tesadüf olduğunu düşünmeyin. Bunlar dünyada en yaygın kullanılan ve sevilen dillerden üçüdür. Eğer bu diller arasında bir bağlantı kurmak gerekirse. **C#** programlama dili; C dilinin söz dizimini ve C++ dilinin nesne yönelimli programlama yöntemlerini ve tekniklerini almıştır.

C# Programlama



- **C#** 'ın Java ile olan ilişkisi ise özel bir durumdur. Zira **C#** 'ta Java gibi C ve C++ dillerinden özellikler almış ama aynı zamanda Java gibi platform bağımsız kod amaçlı tasarlanmıştır. Yani **C#** dili, Java dilinden türememiştir. En iyi tarafı ise C, C++, **C#** veya Java dillerinden birinde iyi olmanız diğer tüm dillerde de size büyük kolaylıklar sağlar.

C# Programlama



- C# kendi başına incelenebilecek bir bilgisayar dili olmasına rağmen C#'ın çalıştırma ortamı (runtime environment) ile, yani .NET Framework ile özel bir ilişkisi vardır. Bunun iki sebebi vardır. Birincisi, C# başlangıçta Microsoft tarafından .NET Framework için kod geliştirmek amacıyla tasarlanmıştı. İkincisi, C# tarafından kullanılan kütüphaneler, .NET Framework tarafından tanımlanan kütüphanelerdi. Yani, C# dilini .NET ortamından ayırmak mümkün olsa dahi her ikisi birbiriyle sıkı sıkıya bağlıdır. Bu nedenle, .NET Framework'ü genel olarak anlamak ve bunun C# için önemini kavramak gereklidir.

.Net Framework

- .NET Framework çatısı farklı dillerin aynı ortamda çalışmasını sağlar. Yani bir projenin bir bölümü VB.NET diğer bölümü C# ile kodlanmış olabilir. .NET Framework barındırdığı ortak dil çalışma zamanı (Common Language Runtime) CLR sayesinde, .NET in destektediği diler ile yazılmış kodları makine diline çevirerek, Yazılımlarda çoklu dil kullanmaya olanak sağlar. Ayrıca .NET kütüphanesi ile kolay,güçlü ve zengin bir programlama ortamı sunar.

C# Derleyicisi

- Csc = C # Compiler (Yani C# kodlarını *.cs derleyen exe dosyasıdır. (csc.exe) Visual Studio içinden cs kodlarını derleme emrini verdiğimizde de csc.exe dosyası ile derleme yapılmaktadır. Csc.exe ile Visual Studio'nun kurulu olmadığı bir bilgisayarda da *.cs kodlarını derleyebiliriz.
- Derleyici programlardan bazıları;

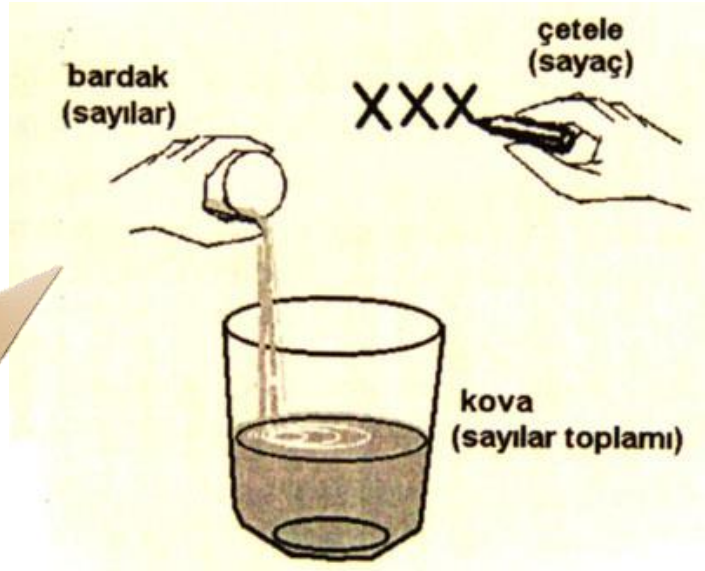
Microsoft® Visual Studio®

Mono Develop (C#)

SharpDevelop

Değişken Nedir?

- **Değişkenler**, girdiğimiz değerleri alan veya programın çalışmasıyla bazı değerlerin atandığı veri tutuculardır. Değişken tanımlama ise girilecek ve ya programın getireceği verinin ne olduğuna bağlı olarak değişken türünün belirlenmesidir. Bir çok farklı türde değişkenler kullanılmaktadır.



Değişken Tanımlama ve Türleri

- Programlamada, geçici verileri tutmak için değişkenleri kullanırız. Değişkenleri kullanmadan önce tanımlamamız gerekir. C# ta değişken tanımlama şekli <veri tipi> <değişken adı> şeklinde olur

- **char** a='g';
  string b="deneme";
  string c=a+b+"Selam millet"+'m';
- **bool** b1=**true**;
  bool b2=**false**;
  bool b3=5>4;
- **object** a=5;
  object b='k';
  object c="metin";
  object d=12.7f;

Değişken Tanımlama ve Türleri

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	$\pm 1.5*10^{-45}$, ..., $\pm 3.4*10^{38}$ (reel sayı)	`float a=5F;` veya `float a=5f;`
double	8 bayt	$\pm 5.0*10^{-324}$, ..., $\pm 1.7*10^{308}$ (reel sayı)	`double a=5;` veya `double a=5d;` veya `double a=5D;`
decimal	16 bayt	$\pm 1.5*10^{-28}$, ..., $\pm 7.9*10^{28}$ (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="Ben bir zaman kaybıyım, beni boşver hocam";`

Değişken Tanımlama Kuralları

- Değişken adları boşluk, simge, özel karakter içeremez.
- Değişkenler bir numerik karakterle başlayamaz.
- C#'ın diğer bütün **komut**, **metot** ve **benzerlerinde** olduğu gibi değişken adlarında **büyük-küçük harf duyarlılığı** vardır. Yani "**degisken**" isimli bir değişkenle "**Degisken**" isimli bir değişken birbirinden farklıdır.
- Değişken adları Türkçe karakterlerden(ğ,ü,ş,ö,ç,ı,İ) oluşamaz.
- Değişken tanımlamada bir standart oluşması için genel kural Camel Case yazımına uygun olmasıdır.

Örn: ogrenciBolumAdi

İlk Programımız

```
namespace ilkprogram
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Merhaba Dünya!");
        }
    }
}
```

- Bu program konsol ekranına Merhaba dünya! yazıp kapanır. Kapanmasını istemiyorsanız programı debug menüsünden start without debug ile başlatın

Programımızı Satır Satır İnceleyelim

- **class ilkprogram** satırıyla ilkprogram adında yeni bir sınıf oluştururuz. C#'ta yazdığımız her programın en az bir sınıf içermesi zorunludur.

- **{** veya **}** işaretleri herhangi bir sınıfın veya metodun içeriğini belirtmek için kullanılır. İlk **{** karakteriyle önceki satırda açtığımız ilkprogram adlı sınıfımızın içeriğine alınacak kodların başladığını, programın son satırındaki **}** karakteriyle de sınıfımızın içeriğine alınacak kodların sona erdiğini belirtiyoruz.

- **static void Main()** satırıyla sınıfımızın içine Main adlı bir metot yerleştirdik. Bu metodun adı mutlaka Main olmalı, yoksa programımız çalışmaz. Ayrıca bu metot mutlaka oluşturduğumuz sınıfın içinde olmalı. Yine **{** ve **}** karakterleriyle metodumuzun içeriğini belirledik. Dikkat ettiyseniz bir iç içe **{** ve **}** karakterleri söz konusu. Bu durumda koyulan ilk } karakteri son açılan { karakterini kapatır.

Programımızı Satır Satır İnceleyelim

- Programımızın pratikte iş yapan tek kısmı ise **System.Console.WriteLine("Merhaba dünya!");** satırı. Bu satırla ekrana Merhaba dünya! yazdırdık, peki nasıl yaptık? Bunun için .Net Framework kütüphanesindeki hazır bir metottan yararlandık. Bu metot System isim alanının altındaki Console sınıfında bulunuyor. Ve bu metodumuzun adı **Write** ve konsol ekranına yazı yazdırmaya yarıyor. Parantezler arasındaki çift tırnaklar arasına alınan metni ekrana yazdırıyor. Satırın sonundaki **;** karakterini ise **{** ve **}** karakterleri açıp kapatmayan bütün C# satırlarında kullanmamız gerekiyor.

Programımızı Geliştirelim

```
namespace ilkprogram
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Merhaba Dünya!");
            Console.ReadLine();
            Console.Write("İşlem Bitti");
        }
    }
}
```

- Bu programımız önce ekrana **Merhaba Dünya!** yazar. Kullanıcı entera bastığında da **İşlem Bitti** yazıp kendini kapatır. Console sınıfına ait olan **ReadLine** metodu programımızın kullanıcıdan bilgi girişi için beklemesini sağlar, yani programımızı entera basılana kadar bekletir. Kullanıcı entera bastığında da diğer satıra geçilir. Dikkat ettiyseniz ReadLine metodunda parantezlerin arasına hiçbir şey yazmıyoruz. C#'ta bazı metotların parantezleri arasına bir şeyler yazmamız gerekirken, bazı metotlarda da hiçbir şey yazılmaması gerekir.

Formatlı Yazım İşlemleri

- Bazı değerleri formatlı bir şekilde yazdırmamız için kullandığımız işaretlerdir. Örneğin 25000 para birimi olarak yazdırmak gibi 25000 TL

Karakter	İsim	Örnek	Çıktı
C veya c	Currency	Console.Write("{0:C}", 2500);	2.500,00 TL
D veya d	Decimal	Console.Write("{0:D5}", 25);	00025
		Console.Write("{0:D10}", 25);	0000000025
E veya e	Scientific	Console.Write("{0:E}", 250000);	2,500000E+005
F veya f	Fixed-point	Console.Write("{0:F0}", 25);	25
		Console.Write("{0:F2}", 25);	25,00
		Console.Write("{0:F5}", 25);	25,00000
N veya n	Number	Console.Write("{0:N}", 2500000);	2.500.000,00
X veya x	Hexadecimal	Console.Write("{0:X}", 250);	FA

Tablo 1.3: Tam sayı tipindeki değişkenlere uygulanacak format listesi

Formatlı Yazım İşlemleri

- Formatlı yazımda kullanılan parametrelerin açıklaması:
- **C**: Sayıyı para birimi şeklinde gösterir.
- **D**: Tek kullanıldığında bir anlam ifade etmez. Yanına sayı yazılarak kullanılır. Formatı alınacak sayının basamak değeri yanında yazılan sayıdan küçükse arta kalan değer kadar yanına sıfır eklenir.
- **E**: Sayıyı 10 üzeri şeklinde gösterir.
- **F**: Sayıların virgülden sonraki basamak sayısı ayarlamada kullanılır.
- **N**: Sayıyı binlik basamaklara ayırarak yazar.
- **X**: Sayıyı hexdecimal(16' lık sayı sistemi) olarak yazar.
- **# işareti:** Formatlı yazımda her bir sayı için # işaretini kullanabiliriz.

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

            Console.Write("{0:(###) ### ## ##}", 2123552154);

            Console.ReadKey();
        }
```

Tarih Formatı

Karakter	İsim	Örnek(DateTime.Now) için)
d	Kısa Tarih	01.04.2005
D	Uzun Tarih	01 nisan 2005 Cuma
t	Kısa Saat	20:24
T	Uzun Saat	20:24:27
f	Tam Tarih ve Saat	01 Nisan 2005 Cuma 20:24
F	Tam Tarih ve Uzun Saat	01 Nisan 2005 Cuma 20:24:27
g	Varsayılan Tarih ve Saat	01.04.2005 20:24
G	Varsayılan Tarih ve Uzun Saat	01.04.2005 20:24:27
M	Gün ve Ay	01 Nisan
r	RFC1123 Tarih Metni	Fri, 01 Apr 2005 20:24:27 GMT
s	Sortable Tarih Metni	2005-04-01 20:24:27
u	Universal sortable, Yerel Zaman	2005-04-01 T20:24:27Z
U	Universal sortable, GMT	01 Nisan 2005 Cuma 17:24:27
	Ay ve Yıl	Nisan 2005

Tablo 1.4: Tarih tipindeki değişkenlere uygulanacak format listesi

Tarih Formatı

Özel tarih formatlama

Karakter	İsmi	Örnek	Örnek Çıktısı
dd	Gün	`Console.Write("{0:dd}", DateTime.Now);`	01
ddd	Gün İsmi	`Console.Write("{0:ddd}", DateTime.Now);`	Cum
dddd	Tam Gün İsmi	`Console.Write("{0:dddd}", DateTime.Now);`	Cuma
hh	Saat	`Console.Write("{0:hh}", DateTime.Now);`	08
HH	Saat(24 Saat)	`Console.Write("{0:HH}", DateTime.Now);`	20
mm	Dakika 00-59	`Console.Write("{0:mm}", DateTime.Now);`	53
MM	Ay 01-12	`Console.Write("{0:MM}", DateTime.Now);`	04
MMM	Ay İsmi	`Console.Write("{0:MMM}", DateTime.Now);`	Nis
MMMM	Tam Ay İsmi	`Console.Write("{0:MMMM}", DateTime.Now);`	Nisan
yy	Yıl, Son İki Karakter	`Console.Write("{0:yy}", DateTime.Now);`	05
yyyy	Yıl	`Console.Write("{0:yyyy}", DateTime.Now);`	2005
:	Ayırıcı	`Console.Write("{0:hh:mm:ss}", DateTime.Now);`	08:53:56
/	Ayırıcı	`Console.Write("{0:dd/MM/yyyy}", DateTime.Now);`	01/04/2005

Tablo 1.5: Özel tarih format listesi

Operatörler

- Her programlama dilinde olduğu gibi C# ‘ta da işlemler yapabilmemizi sağlayan çeşitli operatörler bulunmaktadır.
- Atama, aktarma operatörleri
- Aritmetik işlem operatörleri
- Karşılaştırma operatörleri
- Mantıksal işlem operatörleri

Atama Operatörleri

- Atama operatörleri, değişkenlere değer atamak için kullanılan sembollerdir. C# ta temel atama operatörü **=** dir. Diğer atama operatörleri:

Operatör	Açıklama
=	Sağdaki değeri soldaki değişkene atar.
+=	Soldakine sağdakini ekler, sonucu soldakine atar.
-=	Soldakinden sağdakini çıkarır, sonucu soldakine atar.
*=	Soldakini sağdaki ile çarpar, sonucu soldakine atar.
/=	Soldakini sağdakine böler, sonucu soldakine atar.
%=	Soldaki ile sağdakinin modula işleminin sonucunu soldakine atar.

Atama Operatörleri

- Programlamada değişkenlerin yazım sırasını belirlemek için tutamaçlar kullanılır. {0} {2} {1}

```
class Program
{
    static void Main(string[] args)
    {
        string adi = "Mutlu YAPICI";
        int yas = 28;
        bool dogrumu = true;

        Console.Write("Adınız {0} Yaşınız {1} dir. Bilgilerin Doğruluğu {2}",adi,yas,dogrumu);
        Console.ReadLine();
        Console.Write("İşlem Bitti");
    }
}
```

Atama Operatörleri

```
int sayi1 = 5;
int sayi2 = 3;

Console.WriteLine("Sayı1 = {0}  ve  Sayı2 = {1}",sayi1, sayi2);
Console.WriteLine("Sayı1 + Sayi2 = {0} dir.", sayi1 + sayi2);
sayi1 += sayi2;
Console.WriteLine("Sayı1 + = Sayi2 yaparsak Sayı1 = {0}  ve Sayı2 = {1} olur.", sayi1 , sayi2);
Console.WriteLine("**************************************************************************");

Console.WriteLine("Sayı1 = {0}  ve  Sayı2 = {1}", sayi1, sayi2);
Console.WriteLine("Sayı1 - Sayi2 = {0} dir.", sayi1 - sayi2);
sayi1 -= sayi2;
Console.WriteLine("Sayı1 - = Sayi2 yaparsak Sayı1 = {0}  ve Sayı2 = {1} olur.", sayi1, sayi2);
Console.WriteLine("**************************************************************************");

Console.WriteLine("Sayı1 = {0}  ve  Sayı2 = {1}", sayi1, sayi2);
Console.WriteLine("Sayı1 * Sayi2 = {0} dir.", sayi1 * sayi2);
sayi1 *= sayi2;
Console.WriteLine("Sayı1 *= Sayi2 yaparsak Sayı1 = {0}  ve Sayı2 = {1} olur.", sayi1, sayi2);
Console.WriteLine("**************************************************************************");

Console.WriteLine("Sayı1 = {0}  ve  Sayı2 = {1}", sayi1, sayi2);
Console.WriteLine("Sayı1 / Sayi2 = {0} dir.", sayi1 / sayi2);
sayi1 /= sayi2;
Console.WriteLine("Sayı1 *= Sayi2 yaparsak Sayı1 = {0}  ve Sayı2 = {1} olur.", sayi1, sayi2);
Console.WriteLine("**************************************************************************");

Console.WriteLine("Sayı1 = {0}  ve  Sayı2 = {1}", sayi1, sayi2);
Console.WriteLine("Sayı1 % Sayi2 = {0} dir.", sayi1 / sayi2);
sayi1 %= sayi2;
Console.WriteLine("Sayı1 %= Sayi2 yaparsak Sayı1 = {0}  ve Sayı2 = {1} olur.", sayi1, sayi2);
Console.WriteLine("**************************************************************************");
```

Atama Operatörleri

İşlemlerin Sonuçlarına Göre a, b ,c nin Değerlerini İnceleyin

```
int a =10
int b= 3
int c=2;
```

```
c=a+b
c+=a+b
c++
a--
b**
```

```
c=b--
c=--b
b=-a+b--
c=a%b
a%=c
```

```
a = c++ - --b;
a += --c + b++ - ++b;
a += --c + b++ - ++d;
```

Atama Operatörleri

İşlemlerin Sonuçlarına Göre x ve y nin Değerlerini İnceleyin

```
int x =5;
int y= 3;
```

x++ = 6	x-- = 4	++x = 6	--x = 4
y++ = 16.63	y-- = 14.63	++y = 16.63	-- y= 14.63

x=5 iken --x * --x = 12

x=5 iken ++x * ++x = 42

x=5 iken x-- * x-- = 20

x=5 iken x++ * x++ = 30

Aritmetik İşlem Operatörleri

- Matematiksel işlemler için kullanılan sembollerdir. İşlem öncelikleri matematikteki sırayla aynıdır.

Operatör	*Açıklama*
+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
%	Modulo
++	1 artırma
--	1 eksiltme

Aritmetik İşlem Operatörleri

```
static void Main(string[] args)
{
 double x = 13;
 double y = 7;
 double sonuç = x % y;
 Console.WriteLine("x % y = {0}", sonuç);
}
```

```
static void Main(string[] args)
{
 double x = 10.57;
 double y = 4.83;
 double sonuç = x % y;
 Console.WriteLine("x % y = {0}", sonuç);
}
```

Aritmetik İşlemler

- Aşağıdaki aritmetik işlemleri gerçekleştirin
- (a/b):(c/d)
- (a+b)*c-(a-c)/2
- a%(b-c)
- $a^2+b^2+c^2$
- $((a^2-b^2)*c^2)/2$

Tür Dönüşümleri

- Modern programlamada birçok kez değişkenlerde tür dönüşümüne ihtiyaç duyulur.
- Örneğin **string** türündeki sayılarla ("5" veya "2" gibi) matematiksel işlem yapmamız gerektiğinde tür dönüşümü yapmamız gerekir.
- Aslında bahsettiğimiz tam olarak tür dönüşümü değildir, sadece bir değişkenin değişik türdeki hâlinin başka bir değişkene atanmasıdır.
- Tür dönüşümleri bilinçli tür dönüşümü ve bilinçsiz tür dönüşümü olmak üzere ikiye ayrılır.

Bilinçsiz Tür Dönüşümü

- C#'ta düşük kapasiteli bir değişken, sabit ya da değişken ve sabitlerden oluşan matematiksel ifade daha yüksek kapasiteli bir değişkene atanabilir.
- Buna bilinçsiz tür dönüşümü denir, bunun için herhangi bir özel kod gerekmez.

```
static void Main()
{
byte a=5;
short b=10;
sbyte c=30;
int d=a+b+c;
string e="deneme";
char f='k';
object g=e+f+d;
long h=d;
float i=h;
double j=i;
double k=12.5f;
Console.WriteLine(j+k);
}
```

Bilinçsiz Tür Dönüşümü Yapılabilecek Türler

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

Bilinçli Tür Dönüşümü

- Programımızda da görebileceğiniz gibi (byte)a ifadesi, a değişkeninin byte hâlini tuttu.

```
using System;
class TurDonusumu
{
static void Main()
{
int a=5;
byte b=(byte)a;
Console.WriteLine(b);
}
}
```

Bilinçli Tür Dönüşümü

- Reel türler tam sayı türlere dönüşürken ondalık kısım atılır.
- **Bilinçsiz** tür dönüşümüyle yalnızca küçük türler büyük türlere dönüşebiliyordu, yani veri kaybı olması imkansızdı.
- Halbuki **bilinçli** tür dönüşümünde veri kaybı gerçekleşebilir.
- Eğer dönüşümünü yaptığımız değişkenin tuttuğu değer dönüştürülecek türün kapasitesinden büyükse veri kaybı gerçekleşir

```
static void Main()
{
byte b=(byte)12.5f;
Console.WriteLine(b);
}
```

C# ta Yorum Satırı Ekleme

- Yorumlar, programa etki etmeyen ancak kendimizin veya kodu inceleyen bir başkasının nerede ne yapıldığını anlamasını sağlayacak yazılardır.
- Yorumlar tek satırlık ve çok satırlık olmak üzere ikiye ayrılır
- C#, iki tane slash'ten (//) sonra gelen aynı satırdaki yazıları yorum sayar.

```
int a=0; //Burada a'yı tanımladık ve sıfır değerini verdik.
int b=5; //Burada b'yi tanımladık ve beş değerini verdik.
Console.Write(a+b); //Burada toplamlarını ekrana yazdırdık.
```

- C#, /* ve */ arasına yazılan her şeyi yorum sayar. Dolayısıyla da çok satırlık yorumlar yazılabilir.

```
Console.Write("deneme"); /*Burası birinci satır,
                           burası ikinci satır,
                           işte burada bitti.*/
```

İlişkisel (Karşılaştırma) Operatörleri

- Programlamada iki veya daha fazla değişkeni kıyaslamak için kullanılan sembollerdir.

Operatörler	C# 'deki karşılığı	C# 'de örneği	C#'de anlamı
Eşitlik Operatörleri			
=	==	x == y	x eşittir y
≠	!=	x != y	x eşit değildir y
Karşılaştırma Operatörleri			
>	>	x > y	x büyüktür y
<	<	x < y	x küçüktür y
>=	>=	x >= y	x büyüktür yada eşittir y
<=	<=	x <= y	x küçüktür yada eşittir y

İlişkisel (Karşılaştırma)

```
static void Main(string[] args)
{
    int x = 10;
    int y = 4;
    bool sonuç;

    sonuç = (x > y);
    Console.WriteLine("x > y = {0}", sonuç);
    sonuç = (x < y);
    Console.WriteLine("x < y = {0}", sonuç);
    sonuç = (x <= y);
    Console.WriteLine("x <= y = {0}", sonuç);
    sonuç = (x >= y);
    Console.WriteLine("x >= y = {0}", sonuç);
    sonuç = (x == y);
    Console.WriteLine("x == y = {0}", sonuç);
    sonuç = (x != y);
    Console.WriteLine("x != y = {0}", sonuç);
}
```

İlişkisel (Karşılaştırma)

```
static void Main()
{
bool a=4<6;
bool b=6>5;
bool c=7<=7;
bool d=9>=12;
bool e=10==12;
bool f=1!=8;
Console.Write("{0}\n{1}\n{2}\n{3}\n{4}\n{5}",a,b,c,d,e,f);
}
```

Mantıksal Operatörler

- Bu operatörler true veya false sabit ya da değişkenleri mantıksal ve, veya, değil işlemine sokarlar. Bunlar && (ve), || (veya) ve ! (değil) operatörleridir.

Operatör	Türü	C# teki kullanımı	Örnek
&& (VE)	Mantıksal	&&	a && b = true/false
|| (Veya)	Mantıksal	||	a || b = true/false
!= (Değil)	Mantıksal	!=	a != b = true/false

Mantıksal Operatörler

```
using System;
class Operatorler
{
static void Main()
{
bool b1=35>10&&10==50; //false
bool b2=35>10&&10!=50; //true
bool b3=5 is int||12*3==200; //true
bool b4=5<4||45/5==9; //true
bool b5=!(5<4); //true
Console.Write(b1+" "+b2+" "+b3+" "+b4+" "+b5);
}
}
```

Mantıksal Operatörler

```
public static void Main()
{
    int x = 5;
    int y = 4;

    Console.WriteLine(5 < 6-1 || 7 > 6  );
    Console.WriteLine(5 >= 4 || 7 < 6 + 3);
    Console.WriteLine(5 != 4 || 7 - 1 == 6);
    Console.WriteLine(!(5 == 4) || 7 > 6);
}
```

Bitsel Operatörler

- Bu operatörler bitsel karşılaştırma ve işlem için kullanılan operatörlerdir. . Bunlar && (ve), || (veya) ve ! (değil) operatörleridir.
- Bitsel operatörler sayıların kendisi yerine sayıların bitleriyle ilgilenirler. Diğer bir deyişle sayıları ikilik sisteme dönüştürüp öyle işlem yaparlar.
- Bu operatörler yalnızca tam sayı sabit, değişken ya da ifadelerle kullanılabilirler.
- Eğer bitsel operatörler **bool** türünden değişken, sabit ya da ifadelerle kullanılırsa mantıksal operatörlerin gördüğü işin aynısını görürler

Bitsel Operatörler

Operatör	Türü	C# teki kullanımı	Örnek
& (Bitsel VE)	Bitsel	&	a & b
| (Bitsel Veya)	Bitsel	|	a | b
~ (Bitsel Değil)	Bitsel	~	a ~ b
^ (Bitsel Özel Veya)	Bitsel	^	a ^ b

```
using System;
class Operatorler
{
static void Main()
{
byte a=5&3; // 00000101&00000011 -->sonuç 00000001 çıkar. Yani 1 çıkar.
byte b=5|3; // 00000101|00000011 -->sonuç 00000111 çıkar. Yani 7 çıkar.
byte c=5^3; // 00000101^00000011 -->sonuç 00000110 çıkar. Yani 6 çıkar.
byte d=5;
byte e=(byte)~d; // 00000101'in değili -->sonuç 11111010 çıkar. Yani 250 çıkar.
byte f=(byte)~(a+b); // 00001000'ın değili -->sonuç 11110111 çıkar. Yani 247 çıkar.
byte g=(byte)~(a+7); // 00001000'ın değili --->sonuç 11110111 çıkar. Yani 247 çıkar.
Console.Write(a+" "+b+" "+c+" "+e+" "+f+" "+g);
}
}
```

Bitsel Kaydırma (Shift)Operatörleri

Operatör	Türü	C# teki kullanımı	Örnek
<< (Sola Kaydırma)	Bitsel	<<	a << 1 (1 bit)
>> (Sağa Kaydırma)	Bitsel	>>	a >>2 (2 bit)

```
using System;
class Operatorler
{
static void Main()
{
byte b=100; // 01100100
byte c=(byte)(b<<1); //b bir kez sola kaydırıldı ve 11001000 oldu. (200 oldu.)
byte d=50; // 00110010
byte e=(byte)(d>>2); // d iki kez sağa kaydırıldı ve  00001100 oldu. (12 oldu.)
Console.Write(c+" "+e);
}
}
```

Convert.ToString(sayi1,2)

İS Operatörü

- Verilen değişken, sabit ya da ifadenin **türünü kontrol** eder. Eğer söz konusu değişken, sabit ya da ifade verilen türle uyumluysa **true** değilse **false** değeri üretir. Eğer söz konusu değişken, sabit ya da ifadenin türü her zaman true ya da false üretiliyorsa derleyici uyarı verir, ancak bu uyarı derlemeye engel değildir.

İS Operatörü

```
using System;
class Operatorler
{
static void Main()
{
int i=50;
bool b1=i is int;
bool b2=i is double;
bool b3=12 is byte;
bool b4='c' is string;
bool b5=12f+7 is int;
Console.Write("{0}\n{1}\n{2}\n{3}\n{4}",b1,b2,b3,b4,b5);
}
}
```

- Bu program alt alta True, False, False, False ve False yazacaktır. Dikkat ettiyseniz 12 sabitini byte saymadı ve 12f+7 sabitini de int saymadı. Çünkü C#, harfsiz tam sayı sabitleri int sayar. 12f+7 sabiti de floatlaşmıştır.

?: Operatörü

- ?: operatörü C#'ta üç operand alan tek operatördür. Verilen koşula göre verilen değerlerden (object türünden) birini tutar. Kullanımı şöyledir:
- koşul**?**doğruysa_değer**:**yanlışsa_değer

```
static void Main(string[] args)
{
    string cevap;
    Console.WriteLine("Programlamayı Seviyormusunuz Cevap (E , H) olmalıdır.");
    cevap= Console.ReadLine();
     Console.WriteLine(cevap == "e" ? "Başarılar Dileriz." : "Sağlık Olsun. Çok Çalış");
}
```

?: Operatörü

- Girilen sayı 50 den küçük ise b değişkenine 10 değilse b değişkenine 20 değerini atayan ?: operatörü ile yazılmış program

```
Console.Write("Bir değer giriniz:");
int a = Convert.ToInt32(Console.ReadLine());
int b;

b = (a < 50) ? 10 : 20;
Console.WriteLine(b);
```