

**T.C.
MİLLÎ EĞİTİM BAKANLIĞI**

BİLİŞİM TEKNOLOJİLERİ

BASİT KODLAR
482BK0122

Ankara, 2011

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR	ii
GİRİŞ	1
ÖĞRENME FAALİYETİ-1	2
1. DEĞİŞKENLER VE SABİTLER	2
1.1. Değişkenler	2
1.2. Değişkenleri İsimlendirme Kuralları.....	4
1.3. Veri tipleri	4
1.4. Sabitler	6
1.5. Atama İşlemi	7
1.6. Çıkış İşlemleri	10
1.6.1. Bir metin ifadesini ekrana yazdırma	10
1.6.2. İlk değer atanan değişken değerini ekrana yazdırma	12
1.6.3. Formatlı çıkış işlemleri	16
1.7. Giriş İşlemleri	18
1.7.1. Klavyeden değişkene değer atama.....	18
1.8. Giriş-çıkış işlemleri hata mesajları	21
1.9. Açıklama Satırları	25
UYGULAMA FAALİYETİ	26
ÖLÇME VE DEĞERLENDİRME	27
ÖĞRENME FAALİYETİ-2	29
2. OPERATÖRLER	29
2.1. Aritmetiksel Operatörler	29
2.1.1. Dört İşlem	29
2.1.2. Mod Alma.....	32
2.2. İlişkisel Operatörler.....	34
2.3. Mantıksal Operatörler	36
2.4. İşlem Önceliği	37
UYGULAMA FAALİYETİ	38
ÖLÇME VE DEĞERLENDİRME	39
MODÜL DEĞERLENDİRME	40
CEVAP ANAHTARLARI	42
KAYNAKÇA	43

AÇIKLAMALAR

KOD	482BK0122
ALAN	Bilişim Teknolojileri
DAL/MESLEK	Alan Ortak
MODÜLÜN ADI	Basit Kodlar
MODÜLÜN TANIMI	Bir programla dilinde kullanılan basit kodlara ait bilgilerin verildiği öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	
YETERLİK	Basit kodlar yazmak
MODÜLÜN AMACI	Genel Amaç Bu modül ile gerekli ortam sağlandığında temel programlamada basit kodlar yazabileceksiniz. Amaçlar <ol style="list-style-type: none">1. Değişken ve sabit kullanabileceksiniz.2. Operatörleri kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Ortam: Bilgisayar laboratuvarı Donanım: Kâğıt, kalem, akış diyagramları ile ilgili panolar, bilgisayar, lisanslı programlama yazılımı
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Okul yaşantınızda öğreneceğiniz her konu, yaptığınız her uygulama ve tamamladığınız her modül bilgi dağarcığınızı geliştirecek ve ileride atılacağınız iş yaşantınızda size başarı olarak geri dönecektir. Eğitim sürecinde daha özverili çalışır ve çalışma disiplini kazanırsanız; başarılı olmamanız için hiçbir neden yoktur.

Günümüzde Windows tabanlı görsel programlama dillerinin hızla gelişmekte olduğu ve kullanımının oldukça yaygınlaştığı görülmektedir. Bu programlama dilleri ile sizler programlama mantığını ve becerisini çok daha kolay kavrayacaksınız.

Bu modülle, bir programlama dilinde kullanılabilecek basit kodları ve uygulamalarını öğreneceksiniz.

Bu modülde anlatılan konuların tümünü öğrendiğinizde, bir programlama dilinde kullanılan değişken ve sabit kavramları, tanımlama ve program içinde kullanım şekillerini öğreneceksiniz. Aynı zamanda program içinde tanımladığımız değişken ve sabitler üzerinde işlemler yapabilmek için operatörlerin kullanımlarını kavramış olacaksınız.

ÖĞRENME FAALİYETİ-1

AMAÇ

Bu öğrenme faaliyetinde bir programlama diline uygun değişken ve sabitleri kullanabileceksiniz.

ARAŞTIRMA

- Çevrenizde değişken olarak nitelendirebileceğiniz durumları listeleyiniz.
- Bir gün içinde kullandığınız nesneleri türlerine göre gruplandırınız.

1. DEĞİŞKENLER VE SABİTLER

1.1. Değişkenler

Değişkenler bir programlama dilinde verilerin depolanma alanlarını temsil eder. Tanımlanan her değişkene bellek bölgesinden bir alan ayrılır. Bu bellek bölgesine okuma ve yazma işlemleri ise değişken ismi üzerinden sağlanır. Genel olarak değişkenler aşağıdaki şekilde tanımlanır.

`<veri tipi> <değişken adı>;`

Örneğin;

`int i;`

Yukarıdaki örnekte bir int(sayı) veri tipinde bir değişken tanımlanmıştır. Böylelikle bellek bölgesinde bir veri saklamak ve ileride kullanmak üzere 4 byte'lık bir alan açmış bulunuyoruz. Program içinde bu bellek bölgesine erişmek için değişken ismi olan i ifadesini kullanıyoruz.

Örneğin;

```
int i; //i adında bellekte 4 bytelık bir bölge aç. ;  
i = 5; //i adının temsil ettiği bellek bölgesine 5 değerini yaz. ;
```

- Bir değişkene değer atama işlemi tanımlarken yapılabilir.
- Bir değişkene değer atama işlemi yukardaki örnekte olduğu gibi program içinde herhangi bir satırda yapılabilir.
- Bir veri tipi altında birden fazla isimle farklı değişkenler tanımlanabilir.

Örnek 1.1-1:

```
bool dogrumu = false;  
double yuzde = 98.32, ortalama = 35.32;  
char karakter = 'A';
```

Bazı programlama dillerinde yerel bir değişken tanımlıyorsak bu değişkeni kullanmadan önce bir değer ataması yapmak zorundayız.

```
static void Main(string[] args)  
{  
    int sayi;  
    //sayi = 5;  
    Console.WriteLine(sayi);  
    Console.ReadKey();  
}
```

Yukarıdaki kod blokunda sayi değişkenine değer atamadığınız için hata verecektir (**Use of unassigned local variable 'sayi'**). Çünkü sayi değişkeni yerel bir değişkendir. Eğer `//sayi = 5;` açıklama satırındaki `//` işaretlerini siler ve bu satırı koda dâhil ederseniz hatanın ortadan kalktığını görürsünüz.

Tanımlanan yerel değişkenler sadece tanımlandıkları bloktan erişebilir. Yaşam döngüleri sadece o blok olduğu için başka bir bloktan erişilemez.

Örnek 1.1-2:

```
namespace megepBilisim  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            { //Birinci blok  
                int a = 10;  
            }  
            { //İkinci blok  
                int a = 20;  
            }  
            Console.WriteLine(a); // a değişkeni bu blokta tanımlanmadığı  
            için program hata verecektir.  
        }  
    }  
}
```

Yukarıdaki örnekte birinci ve ikinci blokta tanımlanan "a" isimli değişkenler sadece kendi bloklarında geçerlidir. Main bloğunda kullanılmaya çalışıldığı zaman program hata verecektir.

1.2. Değişkenleri İsimlendirme Kuralları

- Değişkenlerin isimleri alfabede bulunan karakterlerle veya _(alt çizgi) ile başlamalıdır. Ama ilk harf hariç diğer karakterler sayı olabilir.
- Bazı programlama dilleri büyük ve küçük harf duyarlıdır. Yani **Sayı**, **sayı** ve **SAYI** hepsi ayrı değişken olarak algınalır.
- Değişken isimleri birden fazla kelime olduğu zaman; kelimelerin arasına boşluk konmaz. Bu tür değişkenleri ya kelimeleri birleştirerek veya kelimeler arasına _(alt çizgi) karakteri koyarak isimlendiririz.
- Değişkenlerin isimleri !, ?, {, } gibi karakterler içeremez.
- Programlama dili için tanımlanmış anahtar kelimelerini de değişken isimleri olarak kullanamayız.

Standart yazım şekilleri:

Camel notasyonunda isim küçük harfle başlar, eğer değişken isminde birden fazla kelime geçiyor ise isimdeki diğer kelimeler büyük harfle başlar.

Camel Notasyonu:

```
maas ;  
maasMiktari ;  
massMiktariAciklama ;
```

Pascal Notasyonunda kelime büyük harfle başlar. Camel Notasyonunda da olduğu gibi diğer kelimelerde büyük harfle başlar.

Pascal Notasyonu:

```
Maas ( ) ;  
MaasHesapla ( ) ;
```

Bu notasyonların kullanımı mecburi değildir. Fakat sürekli olarak bu tür bir notasyona uyarak kodlarınızı yazarsanız, kodlarınız daha anlaşılır bir hâle girer.

1.3. Veri Tipleri

.Net de iki çeşit veri tipi vardır:

- Değer tipleri (value type)
- Referans tipleri(reference type)

Değişkenler bellekte bulunan verilerdir. Bir değişkeni kullandığımız zaman o değişkenin bellekte bulunduğu yerdeki bilgiyi kullanırız. Değer tipleri belleğin ‘stack’ bölgesinde saklanır ve veriyi direkt olarak bellek bölgesinden alırken referans tipleri bellekte ‘heap’ alanında saklanır. Yani referans tipleri içinde veri değil bellekteki ‘heap’ alanının adres bilgisini tutarlar.

int, double, float gibi veri tipleri değer tiplerine örnek gösterilebilir. Herhangi bir sınıf türü ise referans tipine örnek gösterilebilir. Değer tipleri birbirine eşitlenirken değişkenin barındırdığı değer bir diğer değişkene kopyalanır. Böylece iki farklı bağımsız değişken oluşur. Referans tipleri ise eşitleme sırasında değişkenlerin taşıdıkları veri değil 'heap' bölgesinde işaret ettikleri adres kopyalanır. Böylece eğer iki referans değişkeni birbirine eşitledi isek ve daha sonra bunlardan birinde bulunan veriyi değiştirdi ise otomatik olarak diğer referans değişkeninin değeri de değişir. Çünkü adresde bulunan veri değişince bu adresi işaret eden iki değişkende yeni veri bilgisine ulaşır.

Toplam 15 veri tipi vardır bunlardan 13'ü değer tipindedir, 2'si ise referans tipindedir.

Değer tipleri

Adı	CTS Karşılığı	Açıklama	Max ve Min aralık yada değeri
sbyte	System.Byte	8 bit işaretli tam sayı	-128 : 127
short	System.Int16	16 bit işaretli tam sayı	-32.768 : 32.767
int	System.Int32	32 bit işaretli tam sayı	-2.147.483.648 : 2.147.483.647
long	System.Int64	64 bit işaretli tam sayı	-9.223.372.036.854.775.808 : -9.223.372.036.854.775.807
byte	System.Byte	8 bit işaretsiz tam sayı	0,177083333
ushort	System.UInt16	16 bit işaretsiz tam sayı	0 : 65.535
uint	System.UInt32	32 bit işaretsiz tam sayı	0 : 4.294.967.295
ulong	System.UInt64	64 bit işaretsiz tam sayı	0 : 18.446.744.073.709.551.615
float	System.Single	32 bit tek kayan sayı	+yada - $1,5 \cdot 10^{-45}$: + ya da - $3,4 \cdot 10^{38}$
double	Sytem.Double	64 bit çift kayan sayı	+yada - $5 \cdot 10^{-324}$: + ya da - $1,7 \cdot 10^{308}$
decimal	System.Decimal	128 bit ondalıklı sayı	+yada - $1,5 \cdot 10^{-28}$: + ya da - $7,9 \cdot 10^{28}$
bool	System.Boolean		true ya da false
char	System.Char	Karakterleri temsil eder	16 Unicode karakterleri

Tablo 1.1: Değer tip listesi

Referans tipleri

Adı	CTS Karşılığı	Açıklama
object	System.Object	Bütün veri türlerinin türediği kök eleman
string	System.String	Unicode karakterlerinden oluşan string

Tablo 1.2: Referans tip listesi

1.4. Sabitler

Program boyunca sabit kalacak veriler için kullanılan tanımlamalardır. Bir sabit tanımlamak için **const** anahtar kelimesini kullanırız.

- Sabitler tanımlanırken ilk değer ataması yapılmak zorundadır.
- Program boyunca sabit değeri değiştirelemez.

Kullanımı: **const** <veri tipi> <değişken adı> = değer;

Örnek 1.4-1:

```
const double PI = 3.14;  
const double PI;
```

Yukarıdaki tanımlanan sabitlerden birincisi doğru ikincisi ise yanlış bir tanımlamadır. Çünkü sabitler tanımlanırken ilk değer ataması yapılmak zorundadır.

Sabit kullanım hata mesajları :

- **The left-hand side of an assignment must be a variable, property or indexer:** Tanımlanan sabit değişkenin değeri değiştirilmeye çalışıldığı zaman oluşacak hata mesajıdır. Bu yüzden sabit değişken tanımlanırken atanan değer program boyunca sabit kalmalıdır.

Örnek 1.4-2:

```
const double pi = 3.14159265;  
pi = 2 * pi; //Burada pi değerini değiştirdiğiniz için hata mesajı  
alırsınız.
```

- **A const field requires a value to be provided:** Tanımlanan sabit değişkene ilk değer ataması yapılmadıysa oluşan hata mesajıdır. Dolayısıyla sabit değişken tanımlanırken ilk değer ataması yapılmalıdır.

Örnek 1.4-3:

```
const int pi; // Burada pi sabitine değer atanmadığı için hata mesajı  
alırsınız.
```

1.5. Atama İşlemi

= operatörü: Genel atama işlemlerinde kullanılır. Eşitliğin sağındaki değer eşitliğin solundaki değişkene atanır.

Örnek 1.5-1:

```
int x, y=5; // 5 değerini y değişkenine atamak için = operatörü kullanılmıştır.
```

```
x = y + 2; // y değişken değeri ile 2 sayısı toplanarak x değişkenine atamak için = operatörü kullanılmıştır.
```

+= operatörü: Eşitliğin sağındaki değerle eşitliğin solundaki değişken değerini toplayıp tekrar eşitliğin solundaki değişkene atar.

Örnek 1.5-2:

```
int x=0, y=0, z=0;
```

```
x += 5; //x'e 5 ekle ve x'e eşitle 2.yol x = x + 5 şeklinde de yazılabilir.
```

```
y += 7; //y'ye 7 ekle ve y'ye eşitle 2.yol y = y + 7 şeklinde de yazılabilir.
```

```
z += x; //z'ye x'i ekle ve z'ye eşitle 2.yol z = z + x şeklinde de yazılabilir.
```

İşlem sonucu: x=5, y=7, z=5 olur.

Not: Bir bir artırma işlemi için $x+=1$ (veya $x=x+1$) yerine $x++$ işlemi kullanılabilir.

Örnek 1.5-3:

```
int x=0, y=0, toplam;
```

```
x++; //x'i bir artır
```

```
y++; //y'yi bir artır
```

```
toplam = x + y; //x ve y'yi toplayarak toplam değişkenine ata.
```

İşlem sonucu: x=1, y=1, toplam=2 olur.

➤ **++ değişkenden sonra kullanılırsa önce atama işlemi yapılır sonra artırma yapılır.**

Örnek 1.5-4:

```
int x=0, y=0, toplam;
```

```
x=y++;
```

```
toplam = x + y;
```

önce x y'ye eşitlenir, daha sonra y artırılır. İşlem sonucu: x=0, y=1, toplam=1 olur.

➤ ++ değişkenden önce kullanılırsa önce artırım yapılır daha sonra atama işlemi yapılır.

Örnek 1.5-5:

```
int x=0, y=0, toplam;
```

```
x=++y;
```

```
toplam = x + y;
```

önce y artırılır daha sonra x y'ye eşitlenir. İşlem sonucu: x=1, y=1, toplam=2 olur.

-= operatörü: Eşitliğin sağındaki değeri eşitliğin solundaki değişken değerinden eksilterek tekrar eşitliğin solundaki değişkene atar.

Örnek 1.5-6:

```
int x=50, y=50, z=100;
```

```
x -= 5; //x'den 5'i çıkar ve x'e eşitle 2.yol x = x - 5 şeklinde de yazılabilir.
```

```
y -= 7; //y'den 7 yi çıkar ve y'ye eşitle 2.yol y = y - 7 şeklinde de yazılabilir.
```

```
z -= x; //z'den x'i çıkar ve z'ye eşitle 2.yol z = z - x şeklinde de yazılabilir.
```

İşlem sonucu: x=45 , y=43 , z=55 olur.

Not: Bir bir azaltma işlemi için x-=1 (veya x=x-1) yerine x-- işlemi kullanılabilir.

Örnek 1.5-7:

```
int x=20, y=10, fark;
```

```
x--; //x'i bir azalt
```

```
y--; //y'yi bir azalt
```

```
fark = x - y; //x ve y'yi çıkararak fark değişkenine ata.
```

İşlem sonucu: x=19 , y=9 , fark=10 olur.

➤ -- değişkenden sonra kullanılırsa önce atama işlemi yapılır, sonra azaltma yapılır.

Örnek 1.5-8:

```
int x=10, y=10,fark;
```

```
x=y--;
```

```
fark = x - y;
```

önce x y'ye eşitlenir, daha sonra y azaltılır. İşlem sonucu: x=10, y=9 , fark=1 olur.

➤ -- değişkenden önce kullanılırsa önce azaltma yapılır daha sonra atama işlemi yapılır.

Örnek 1.5-9:

```
int x = 10, y = 10, fark;
```

```
x = --y;
```

```
fark = x - y;
```

önce y artırılır daha sonra x y'ye eşitlenir. İşlem sonucu : x=9 , y=9 , fark=0 olur.

***= operatörü:** Eşitliğin sağındaki değerle eşitliğin solundaki değişken değeri çarpılıp tekrar eşitliğin solundaki değişkene atar.

Örnek 1.5-10:

```
int x = 2, y = 3, z = 2;
```

```
x *= 2; //x ile 2'yi çarp ve x'e eşitle 2.yol x = x * 2 şeklinde de yazılabilir.
```

```
y *= 2; //y ile 2 yi çarp ve y'ye eşitle 2.yol y = y * 2 şeklinde de yazılabilir.
```

```
z *= x; //z ile x'i çarp ve z'ye eşitle 2.yol z = z * x şeklinde de yazılabilir.
```

İşlem sonucu: x=4 , y=6 , z=8 olur.

/= operatörü: Eşitliğin solundaki değişken değerini eşitliğin sağındaki değere bölerek tekrar eşitliğin solundaki değişkene atar.

Örnek 1.5-11:

```
int x = 4, y = 10, z = 64;
```

```
x /= 2; //x'i 2'ye böl ve x'e eşitle 2.yol x = x / 2 şeklinde de yazılabilir.
```

```
y /= 2; //y'yi 2'ye böl ve y'ye eşitle 2.yol y = y / 2 şeklinde de yazılabilir.
```

```
z /= x; //z'yi x'e böl ve z'ye eşitle 2.yol z = z / x şeklinde de yazılabilir.
```

İşlem sonucu: x=2, y=5, z=32 olur.

1.6. Çıkış İşlemleri

1.6.1. Bir Metin İfadesini Ekrana Yazdırma

Bir metin ifadesini ekrana yazdırmak için iki metot kullanılır:

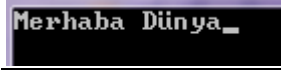
1. Metot **Console.Write()**: Yazdırma işleminden sonra imleç yazdırılan ifadenin yanında bekler.

Örnek 1.6-1:

```
static void Main(string[] args)
{
    Console.Write("Merhaba Dünya");

    Console.ReadKey();// Klavyeden bir tuşa basılana kadar bekle.
}
```

Çıktısı:



Resim 1.1: Örnek 1.6-1 ekran çıktısı

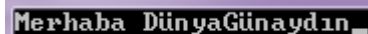
Dikkat ederseniz imleç yazının sonunda beklemektedir. Bir başka metin yazdırmaya çalıştığımız zaman imlecin bulunduğu yerden devam eder.

Örnek 1.6-2:

```
static void Main(string[] args)
{
    Console.Write("Merhaba Dünya");
    Console.Write("Günaydın");

    Console.ReadKey();
}
```

Çıktısı:



Resim 1.2: Örnek 1.6-2 ekran çıktısı

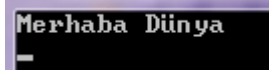
2. Metot Console.WriteLine(): Yazdırma işleminden sonra imleç yazdırılan ifadenin alt satırında bekler.

Örnek 1.6-3:

```
static void Main(string[] args)
{
    Console.WriteLine("Merhaba Dünya");

    Console.ReadKey();
}
```

Çıktısı:



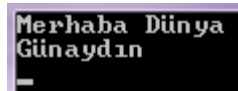
Resim1.3: Örnek 1.6-3 ekran çıktısı

Örnek 1.6-4:

```
static void Main(string[] args)
{
    Console.WriteLine("Merhaba Dünya");
    Console.WriteLine("Günaydın");

    Console.ReadKey();
}
```

Çıktısı:



Resim 1.4: Örnek 1.6-4 ekran çıktısı

Programı çalıştırdığımızda ikinci metin bir alt satıra yazılmıştır.

Çıkış parametreleri:

\n: Bir alt satıra geçmek için kullanılır.

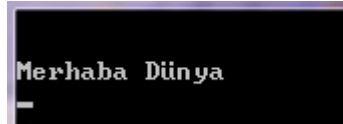
\r: Paragraf başı yapmak için kullanılır.

Örnek 1.6-5:

```
static void Main(string[] args)
{
    Console.WriteLine("\n\nMerhaba Dünya");

    Console.ReadKey();
}
```

Çıktısı:



Resim 1.5: Örnek 1.6-5 ekran çıktısı

Ekran çıktısına baktığımızda 2 tane `\n` kullandığımız için yazımız 2 satır aşağıdan başlamıştır.

1.6.2. İlk Değer Atanan Değişken Değerini Ekranı Yazdırma

Değişken tanımlamayı daha önce görmüştük. Şimdi değer atadığımız değişkeni ekrana yazdırma işlemlerini göreceğiz.

Değişkene ilk değeri tanımlarken veya tanımladıktan sonra atayabiliriz.

```
int x = 5; // ilk değeri tanımlanırken atadık.
```

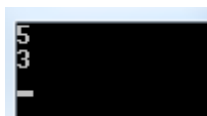
```
int y;  
y = 3; // ilk değeri tanımlandıktan sonra atadık.
```

Bu değişkenleri ekrana nasıl yazdırabileceğimize bakalım.

Örnek 1.6-6:

```
static void Main(string[] args)  
{  
    int x = 5; // ilk değeri tanımlanırken atadık.  
  
    int y;  
    y = 3; // ilk değeri tanımlandıktan sonra atadık.  
  
    Console.WriteLine(x);  
    Console.WriteLine(y);  
  
    Console.ReadKey();  
}
```

Çıktısı:

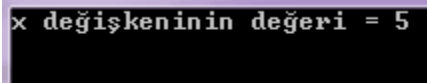


Resim 1.6: Örnek 1.6-6 ekran çıktısı

Örnek 1.6-7:

```
static void Main(string[] args)
{
    int x = 5; // ilk değeri tanımlanırken atadık.
    int y;
    y = 3; // ilk değeri tanımlandıktan sonra atadık.
    Console.WriteLine("x değişkeninin değeri = {0}", x); // x'in
    değerini süslü parantez içinde belirtilen sıfır (0) yazan yere yaz.
    Console.ReadKey();
}
```

Çıktısı:



Resim 1.7: Örnek 1.6-7 ekran çıktısı

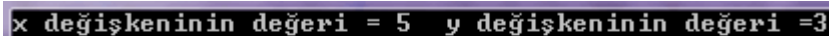
{0} anlamı: Virgülden sonra tanımlanan ilk değişken değerini sıfırın yerine yazınız.

Eğer aynı anda birden fazla değişken değeri ekrana yazdırılmak isteniyorsa {0},{1},{2}... şeklinde devam eder. Değişkenler virgülle ayrılarak tanımlanır.

Örnek 1.6-8:

```
static void Main(string[] args)
{
    int x = 5; // ilk değeri tanımlanırken atadık.
    int y;
    y = 3; // ilk değeri tanımlandıktan sonra atadık.
    Console.WriteLine("x değişkeninin değeri = {0} y
    değişkeninin değeri = {1}", x, y);
    Console.ReadKey();
}
```

Çıktısı:



Resim 1.8: Örnek 1.6-8 ekran çıktısı

Değişken değerlerini ekrana birlikte yazdırmanın bir diğer yoluda + işaretini kullanmaktır. + işareti ifadeleri birleştirme işlemini gerçekleştirir.

Örnek 1.6-9:

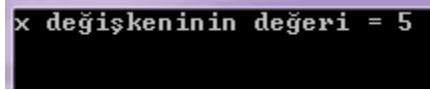
```
static void Main(string[] args)
{
    int x = 5; // ilk değeri tanımlanırken atadık.

    int y;
    y = 3;     // ilk derğeri tanımlandıktan sonra atadık.

    Console.WriteLine("x değişkeninin değeri = " + x);

    Console.ReadKey();
}
```

Çıktısı:



Resim 1.9: Örnek 1.6-9 ekran çıktısı

Görüldüğü üzere süslü parantez ile yapılan örnekle artı işareti kullanılarak yapılan örneğin çıktıları aynıdır.

Örnek 1.6-10:

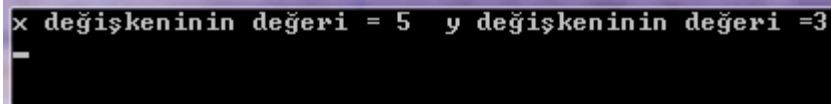
```
static void Main(string[] args)
{
    int x = 5; // ilk değeri tanımlanırken atadık.

    int y;
    y = 3;     // ilk derğeri tanımlandıktan sonra atadık.

    Console.WriteLine("x değişkeninin değeri = " + x + "
y değişkeninin değeri =" + y);

    Console.ReadKey();
}
```

Çıktısı:



Resim 1.10: Örnek 1.6-10 ekran çıktısı

Örnek 1.6-11:

```
static void Main(string[] args)
{
    string ad = "Süleyman"; //İlk değer tanımlanırken atandı.
    string soyad;
    soyad = "GÜRHAN"; //İlk değeri tanımladıktan sonra atadık.

    Console.WriteLine(ad + " " + soyad);

    Console.ReadKey();
}
```



Resim 1.11: Örnek 1.6-11 ekran çıktısı

Örnek 1.6-12:

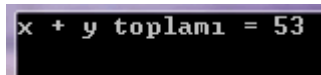
```
static void Main(string[] args)
{
    int x = 5; // ilk değeri tanımlanırken atadık.

    int y;
    y = 3; // ilk derğeri tanımlandıktan sonra atadık.

    Console.WriteLine("x + y toplamı = " + x + y);

    Console.ReadKey();
}
```

Çıktısı:



Resim 1.12: Örnek 1.6-12 ekran çıktısı

Yapmak istediğimiz işlem aslında x ve y değişken değerlerini toplayıp ekrana yazdırmaktı. Fakat + işaretinin buradaki görevi ifadeleri birleştirmek olduğundan x ve y değişken değerlerini yan yana yazarak yanlış sonuç üretmiştir.

Örnek 1.6-13:

```
static void Main(string[] args)
{
    int x = 5; // ilk değeri tanımlanırken atadık.

    int y;
    y = 3; // ilk derğeri tanımlandıktan sonra atadık.
```

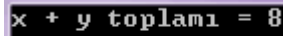
```

        Console.WriteLine("x + y toplamı = " + (x + y)); // x + y
işlemini parantez içine alırsak + işareti toplama işlemini
gerçekleştirecektir.

        Console.ReadKey();
    }

```

Çıktısı:



Resim 1.13: Örnek 1.6-13 ekran çıktısı

`Console.WriteLine("x + y toplamı = " + (x + y));` satırında `x + y` işlemini parantez içine alınca doğru sonuç elde edilmiştir.

Kod satırımızı `Console.WriteLine("x + y toplamı = {0}", x + y);` şeklinde de yazsaydık aynı sonucu elde ederdik.

1.6.3. Formatlı Çıkış İşlemleri

Tam sayı tipinde tanımlanmış değişkenler üzerinde uygulanabilecek format biçimleri aşağıdaki tablola belirtilmiştir.

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

Tablo 1.3: Tam sayı tipindeki değişkenlere uygulanacak format listesi

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 artı 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.

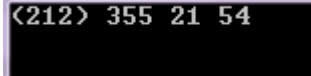
X: Sayıyı hexadecimal(16' lık sayı sistemi) olarak yazar.

işareti: Formatlı yazımda her bir sayı için # işaretini kullanabiliriz.

Örnek 1.6-14:

```
static void Main(string[] args)
{
    Console.WriteLine("{0:(###) ### ## #}", 2123552154);
    Console.ReadKey();
}
```

Çıktısı:



Resim 1.14: Örnek 1.6-14 ekran çıktısı

Tarih formatlama

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

Özel tarih formatlama

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

Tablo 1.5: Özel tarih format listesi

1.7. Giriş İşlemleri

1.7.1. Klavyeden Değişkene Değer Atama

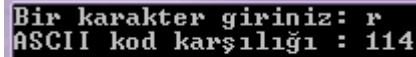
Console.Read(): Console sınıfının Read() metodu kullanıcının klavyeden giriş yapmasını sağlar tek karakter okur ve geriye tam sayı tipinde bir değer döndürür. Bu değer okunan karakterin 'ascii' kod karşılığıdır.

Örnek 1.7-1:

```
static void Main(string[] args)
{
    int x;
    Console.WriteLine("Bir karakter giriniz: ");
    x = Console.Read();
    Console.WriteLine("ASCII kod karşılığı : {0}", x);
}
```

```
} Console.ReadKey();
```

Çıktısı:



```
Bir karakter giriniz: r
ASCII kod karşılığı : 114
```

Resim 1.15: Örnek 1.7-1 ekran çıktısı

Read() metoduyla klavyeden istediğiniz kadar değer okutabiliriz, ama geriye sadece ilk karakterin ascii kod karşılığını döndürecek. Burda dikkat etmemiz gereken bir başka nokta ise Read() metodu geriye tam sayı bir değer döndürdüğü için atamayı tam sayı tipinde bir değişkene yapmalıyız. Aksi hâlde hata mesajı alırız (**Cannot implicitly convert type 'int' to 'string'**).

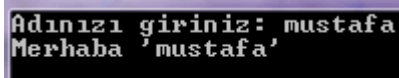
Console.ReadLine(): Console sınıfının ReadLine() metodu kullanıcının klavyeden bir değer girmesini sağlar ve bu değeri metin(string) bir ifade olarak geri döndürür.

Örnek 1.7-2:

```
static void Main(string[] args)
{
    string x;
    Console.Write("Adınızı giriniz: ");
    x = Console.ReadLine();
    Console.WriteLine("Merhaba '{0}' ", x);

    Console.ReadKey();
}
```

Çıktısı:



```
Adınızı giriniz: mustafa
Merhaba 'mustafa'
```

Resim 1.16: Örnek 1.7-2 ekran çıktısı

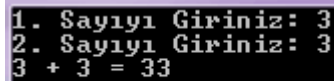
ReadLine() metodu ile geri döndürülen değer metin(string) tipindedir. Dolayısıyla okunan değeri metin(string) tipinde bir değişkene atamalıyız. Eğer ki matematiksel bir işlem yapılacaksa değeri sayısal ifadeye çevirmemiz gerekmektedir. Bu işlem için 'Convert' sınıfı veya veri türlerinin 'Parse' özelliğinden faydalınır.

Örnek 1.7-3:

```
static void Main(string[] args)
{
    string x, y;
    Console.Write("1. Sayıyı Giriniz: ");
    x = Console.ReadLine();
    Console.Write("2. Sayıyı Giriniz: ");
    y = Console.ReadLine();
    Console.WriteLine("{0} + {1} = {2} ", x, y, (x+y));

    Console.ReadKey();
}
```

Çıktısı:



```
1. Sayıyı Giriniz: 3
2. Sayıyı Giriniz: 3
3 + 3 = 33
-
```

Resim 1.17: Örnek 1.7-3 ekran çıktısı

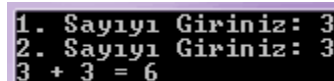
Görüldüğü üzere string ifadeler üzerinde matematiksel işlemler yapamıyoruz. Matematiksel işlem yapacaksa değişkenimizin tipini sayısal ifadeye çevirmeliyiz.

Örnek 1.7-4:

```
static void Main(string[] args)
{
    string x, y;
    Console.Write("1. Sayıyı Giriniz: ");
    x = Console.ReadLine();
    Console.Write("2. Sayıyı Giriniz: ");
    y = Console.ReadLine();
    Console.WriteLine("{0} + {1} = {2} ", x, y,
        (Convert.ToInt16(x)+ Convert.ToInt16(y)));

    Console.ReadKey();
}
```

Çıktısı:



```
1. Sayıyı Giriniz: 3
2. Sayıyı Giriniz: 3
3 + 3 = 6
```

Resim 1.18: Örnek 1.7-4 ekran çıktısı

(Convert.ToInt16(x) ve Convert.ToInt16(y) şeklinde tip dönüşümü yapılarak string tipindeki değişken değerlerimizi tam sayı tipine dönüştürüyoruz. Böylelikle üzerinde matematiksel işlemler yapabiliyoruz.

1.8. Giriş-Çıkış İşlemleri Hata Mesajları

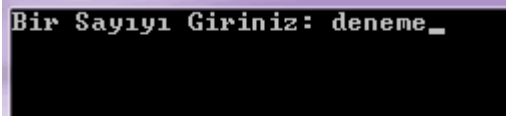
Hazırlayacağımız programın en önemli özelliklerinden biri de stabil çalışması olmalıdır. Stabil çalışması programımızın hatalara karşı ne kadar hazırlıklı ve kullanıcıya verdiği geri dönüşle eş değerdir. Programımızın çalışması sırasında oluşabilecek hatalar genellikle kullanıcı girişlerinden kaynaklanır. Bu yüzden kullanıcı girişlerini kontrol altına alarak çalışma zamanında oluşabilecek hataları en aza indirmek ise biz programcıların görevidir. Önce hatalar oluştuğunda programın nasıl sonlandığını görelim daha sonra bunun için bir çözüm arayalım.

Örnek 1.8-1:

```
static void Main(string[] args)
{
    int x;
    Console.Write("Bir Sayıyı Giriniz: ");
    x = Convert.ToInt16(Console.ReadLine());

    Console.ReadKey();
}
```

Programımızı çalıştıralım ve sayı yerine string bir ifade girelim.

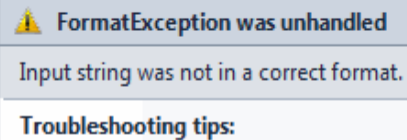
A screenshot of a console window with a black background and white text. The text displayed is "Bir Sayıyı Giriniz: deneme_".


Resim 1.19: Örnek 1.8-1 ekran çıktısı

Programda istenilen sayı yerine 'string' bir ifade girdiğimiz zaman ekran çıktısı aşağıdaki gibi olacaktır. 'String' ifade 'int' tipine çevrilmede zorlanılacağından program duracaktır. Giriş dizesinin doğru olmadığına dair bir hata verecektir (**Input string was not in a correct format.**) .

```
static void Main(string[] args)
{
    int x;
    Console.Write("Bir Sayıyı Giriniz: ");
    x = Convert.ToInt16(Console.ReadLine());

    Console.ReadKey();
}
```

A screenshot of a .NET error message box. The title bar says "FormatException was unhandled". The main text says "Input string was not in a correct format." Below this, it says "Troubleshooting tips:". There is a yellow warning icon on the left.

 FormatException was unhandled
Input string was not in a correct format.
Troubleshooting tips:

Resim 1.20: Giriş dizesi doğru biçimde değil hata mesajı

Örnek 1.8-2:

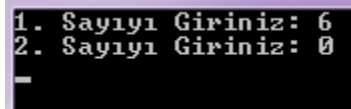
```
static void Main(string[] args)
{
    int x, y;
    Console.Write("1. Sayıyı Giriniz: ");
    x = Convert.ToInt16(Console.ReadLine());

    Console.Write("2. Sayıyı Giriniz: ");
    y = Convert.ToInt16(Console.ReadLine());

    Console.WriteLine("{0} / {1} = {2}", x, y, x / y);

    Console.ReadKey();
}
```

Programımızı çalıştıralım ve aşağıdaki gibi değerleri girelim.



Resim 1.21: Örnek 1.8-2 ekran çıktısı

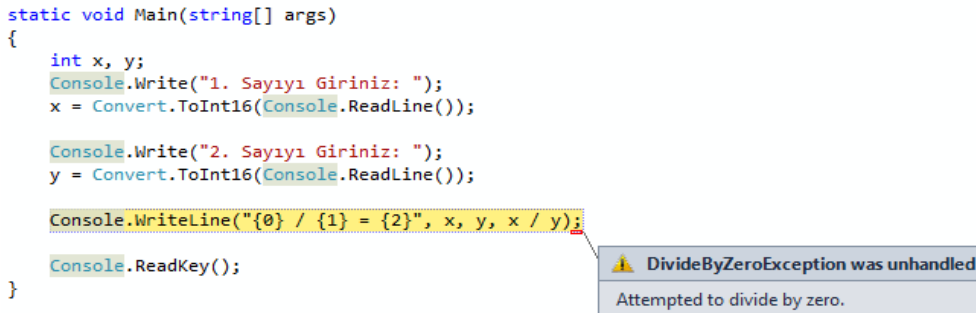
Girilen sayılardan biri sıfır olduğu zaman programımız sıfıra bölme hatasıyla karşılaşacağından dolayı duracaktır (**Attempted to divide by zero**) .

```
static void Main(string[] args)
{
    int x, y;
    Console.Write("1. Sayıyı Giriniz: ");
    x = Convert.ToInt16(Console.ReadLine());

    Console.Write("2. Sayıyı Giriniz: ");
    y = Convert.ToInt16(Console.ReadLine());

    Console.WriteLine("{0} / {1} = {2}", x, y, x / y);

    Console.ReadKey();
}
```



Resim 1.22: Sıfıra bölem hata mesajı

Yukardaki kodlarda da görüldüğü gibi program esnasında kullanıcılar tarafından yapılan hatalı girişler programın hatayla karşılaşmasına ve durmasına sebep olmaktadır. Biz bu hataları program içinde nasıl yakalarız ve kullanıcıya hata hakkında nasıl mesaj veririz buna bakalım.

.Net programcılıkta biz bu tür hatalara istisnalar(Exception) diyoruz. İstisnalar, programımızın çalışma zamanında yani program çalışırken ortaya çıkan olağan dışı durumlardır. .NET ortamında her şey gibi istisnalar da sınıflar kullanılarak oluşturulmakta ve tüm istisnalar temel System.Exception nesnesinden türetilmektedir.

İstisnaları program esnasında yakalamak ve kullanıcıya hata mesajını vermek için `try{} catch{} finally{}` bloklarını kullanıyoruz.

try{} bloku: İstisnanın çıkması muhtemel kodların yazıldığı bloktur.

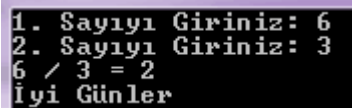
catch{} bloku: Oluşan istisnanın yakalandığı ve kullanıcıya sunulduğu bloktur.

finally{} bloku: Try bloku içinde hata olsa da olmasa da çalışmasını istediğimiz kodların yazıldığı bloktur. Finally bloğu genellikle bazı kaynakları serbest bırakmak için kullanılır. Kullanılması isteğe bağlı bir bloktur. En sık kullanıldığı yerler açık olan veri tabanı bağlantılarının program kırılsa da kırılmasa da kapatılması durumlarıdır.

Örnek 1.8-3:

```
static void Main(string[] args)
{
    int x, y;
    Console.Write("1. Sayıyı Giriniz: ");
    x = Convert.ToInt16(Console.ReadLine());
    Console.Write("2. Sayıyı Giriniz: ");
    y = Convert.ToInt16(Console.ReadLine());
    try
    {
        Console.WriteLine("{0} / {1} = {2}", x, y, x / y);
    }
    catch (Exception e)
    {
        Console.WriteLine("Hata Oluşturdu : {0}", e);
    }
    finally
    {
        Console.WriteLine("İyi Günler");
    }
    Console.ReadKey();
}
```

Programımızı çalıştırıp aşağıdaki gibi değerleri girdiğimiz zaman herhangi bir hatayla karşılaşmadığından istediğimiz sonucu üretecektir.



```
1. Sayıyı Giriniz: 6
2. Sayıyı Giriniz: 3
6 / 3 = 2
İyi Günler
```

Resim 1.23: Örnek 1.8-3 ekran çıktısı

Fakat prgoramımıza aşağıdaki değerleri girdiğimiz zaman hatayla karşılaşılacak ve programımız olduğu yerde durup `catch{}` blokuna atlayacak ve buradan çalışmaya devam edecektir.

```
1. Sayıyı Giriniz: 6
2. Sayıyı Giriniz: 0
Hata Oluşturdu : System.DivideByZeroException: Attempted to divide by zero.
   at megepBilisim.Program.Main(String[] args) in C:\Users\TEMHEM\documents\visual studio 2010\Projects\megepBilisim\megepBilisim\Program.cs:line 18
İyi Günler
```

Resim 1.24: Örnek 1.8-3 ekran çıktısı

Dikkat ederseniz her iki durumda da finally bloku içine yazdığımız kodlar çalıştırılmaktadır.

Örnek 1.8-4:

```
static void Main(string[] args)
{
    byte x ;

    try
    {
        Console.Write("0-255 Arasında Bir Sayıyı Giriniz: ");
        x = Convert.ToByte(Console.ReadLine());

        Console.WriteLine("Doğru Değer Girdiniz");
    }
    catch (Exception e)
    {
        Console.WriteLine("Yanlış Değer Girdiniz");
        Console.WriteLine("Hata Oluşturdu : {0}", e);
    }
    finally
    {
        Console.WriteLine("İyi Günler");
    }

    Console.ReadKey();
}
```

```
0-255 Arasında Bir Sayıyı Giriniz: 34
Doğru Değer Girdiniz
İyi Günler
```

Resim 1.25: Örnek 1.8-4 ekran çıktısı

Eğer bizden istenildiği gibi 0-255 arası bir değer girersek programımız herhangi bir hatayla karşılaşmayacağı için try bloku içindeki tüm kodlar icra edilip finally bloğuna atlar ve çalışmasına ordan devam eder. Yukardaki ekran çıktısında görüldüğü gibi

```
0-255 Arasında Bir Sayıyı Giriniz: 300
Yanlış Değer Girdiniz
Hata Oluşturdu : System.OverflowException: Value was either too large or too small
for an unsigned byte.
   at System.Byte.Parse(String s, NumberStyles style, NumberFormatInfo info)
   at System.Convert.ToByte(String value)
   at megepBilisim.Program.Main(String[] args) in C:\Users\TEMHEM\documents\visual
studio 2010\Projects\megepBilisim\megepBilisim\Program.cs:line 14
İyi Günler
```

Resim 1.26: Örnek 1.8-4 ekran çıktısı

Fakat istenilen değer dışında bir değer girdiğimiz zaman program hatayla karşılaşacağından hatanın olduğu satırda program durdurulur ve catch blokuna atlanır ve çalışmasına ordan devam eder. Yukardaki ekran çıktısında görüldüğü üzere hata `x = Convert.ToByte(Console.ReadLine());` satırında olduğundan bir sonraki satır icra edilmeden catch blokuna atlanmış ve program buradan devam etmiştir.

1.9. Açıklama Satırları

Açıklama satırları programcıya kod içinde tanımlama metinleri yazma imkânı sağlar. Bu sayede kod parçacıklarının ne iş yaptıkları anlatılmış olur. Kodlarımız arasına açıklama satırları eklemek oldukça önemlidir. Az satırlı program kodlarında birşey ifade etmeyebilir fakat büyük programlarda kod bloklarının ne işe yaradıkları yazılarak programcının ileride karşılaşacağı problemleri kolay çözmesinde yardımcı olacaktır. Ayrıca açıklama satırları program derlenirken dosya içerisine alınmadığından oluşan dosyanın boyutunu ya da çalışmasını etkilememektedir.

```
static void Main(string[] args)
{
    //Bu Satır ekrana Merhaba Dünya Yazar.
    Console.WriteLine("Merhaba Dünya");
}
```

// karakterlerinden sonra gelen ve satırın sonuna kadar olan sözcükler yorum satırlarıdır ve programlama dili derleyicisi tarafından görünmez. Aynı zamanda birden fazla satıra yorum eklemek istiyorsak `/* */` karakterleri arasına yorum yazarız.

```
static void Main(string[] args)
{
    /* Bu Satır ekrana Merhaba Dünya Yazar.
       Bu Kısım derleyici tarafından yok sayılır.
    */
    Console.WriteLine("Merhaba Dünya");
}
```

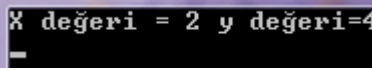
UYGULAMA FAALİYETİ

Aşağıdaki uygulamaları yapınız.

İşlem Basamakları	Öneriler
➤ Bir okul programında kullanılabilecek değişkenleri maddeler hâlinde yazınız.	➤ Değişken isimlendirme kurallarına dikkat ediniz.
➤ Tanımladığınız her değişkene bir tip belirleyiniz.	➤ Her değişkenin bir tipi olması gerektiğini unutmayınız.
➤ Tanımladığımız her değişkene bir değer atayınız.	➤ Tanımlanan her yerel değişkenin bir değeri olması gerektiğini unutmayınız.
➤ Tanımladığımız değişkenlerden sabit olarak ayırabileceklerimizi seçiniz.	➤ Sabitlerin değeri tanımlandıklarında atanır ve program boyunca değiştirilemez.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

- Aşağıdaki değişken isimlerinden hangisi hatalıdır?
A) ogrenciNo B) OgrenciAd
C) ogrenci_soyad D) ogrenci sinif
- Aşağıdakilerden hangisi veri tiplerinden değildir?
A) string B) slong C) int D) byte
- 8 bit işaretli tam sayı tipi aşağıdakilerden hangisidir?
A) byte B) int C) long D) short
- Program boyunca sabit kalacak veriyi hangi kelime ile tanımlarız?
A) float B) double C) bool D) const
- Aşağıdaki kullanımlardan hangisi ile önce artırma işlemi yapıp daha sonra atama işlemi yapılır?
A) ++a B) a++ C) +a D) a+
- Aşağıdaki yazımlardan hangisinin ekran çıktısı  şeklindedir.
A) `Console.WriteLine("X değeri = x y değeri=y");`
B) `Console.WriteLine("X değeri = {0} y değeri={1}" + 2 + 4);`
C) `Console.WriteLine("X değeri = {0} y değeri={1}", 2, 4);`
D) `Console.Write("X değeri = {0} y değeri={1}", 2, 4);`
- Hangi tarih formatını kullanırsak ekran çıktısı "Nisan" şeklinde olur?
A) `Console.Write("{0:MMM}", DateTime.Now);`
B) `Console.Write("{0:MMMM}", DateTime.Now);`
C) `Console.Write("{0:yy}", DateTime.Now);`
D) `Console.Write("{0:dddd}", DateTime.Now);`

Aşağıda verilen cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

- 8.** Bir metin ifadesini ekrana yazdırmak için veya metodunu kullanırız.
- 9.** Metni bir alt satırdan başlayarak yazdırmak için parametresini kullanırız.
- 10.** Muhtemel istisna oluşmasını düşündüğümüz kodları blokları arasına alırız.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

Bu öğrenme faaliyetinde operatörleri kullanabileceksiniz.

ARAŞTIRMA

- Matematik dersinde kullanılan işlem önceliği kurallarını liste hâlinde hazırlayınız.
- Öncelikle kendinize bir hedef belirleyiniz. Bu hedefe ulaşmak için gerekli olan tüm şartları sıralayınız. Hangi şartları gerçekleştirdiğinizde hedefinize ulaşabileceğinizi işaretleyiniz.

2. OPERATÖRLER

Programlama dillerinde tanımlanmış sabit ve değişkenler üzerinde işlemler yapmamızı sağlayan karakter ya da karakter topluluklarına **operatör** denir.

Örneğin;

`int sayi = 2 + 3;`

Yukardaki örnekte + ve = karakterleri birer operatördür. + karakteri 2 ve 3 sabitlerini toplama yapıyor ve = karakteri ise toplanan değeri tanımlanan değişkene atama işlemini gerçekleştiriyor.

2.1. Aritmetiksel Operatörler

Aritmetik işlemler yaparken kullandığımız operatörlerdir.

2.1.1. Dört İşlem

Operatör	Açıklama
+	Toplama İşlemi
-	Çıkarma İşlemi
*	Çarpma İşlemi
/	Bölme İşlemi

Tablo 2.1: Dört işlem operatör listesi

Örnek 2.1-1:

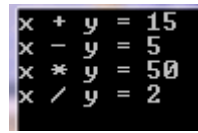
```
using System;
namespace Operatorler
{
    class DortIslem
    {
        static void Main(string[] args)
        {
            int x = 10;
            int y = 5;

            Console.WriteLine("x + y = {0}", x + y);
            Console.WriteLine("x - y = {0}", x - y);

            Console.WriteLine("x * y = {0}", x * y);
            Console.WriteLine("x / y = {0}", x / y);

            Console.ReadKey();
        }
    }
}
```

Çıktısı:



```
x + y = 15
x - y = 5
x * y = 50
x / y = 2
```

Resim 2.1: Örnek 2.1-1 ekran çıktısı

Örnek 2.1-2:

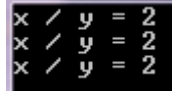
```
using System;
namespace Operatorler
{
    class DortIslem
    {
        static void Main(string[] args)
        {
            int x = 10;
            int y = 4;

            int intSonuc = x / y;
            float floatSonuc = x / y;
            double doubleSonuc = x / y;

            Console.WriteLine("x / y = {0}", intSonuc);
            Console.WriteLine("x / y = {0}", floatSonuc);
            Console.WriteLine("x / y = {0}", doubleSonuc);

            Console.ReadKey();
        }
    }
}
```

Çıktısı:



Resim 2.2: Örnek 2.1-2 ekran çıktısı

Aslında sonuç 2.5 olması gerekirken 2 olarak çıktı. Bunun sebebi ise x ve y değişkenleri int yani tam sayı tipinde tanımlandıklarından çıkan sonucun da kesirli kısmı atılıp tam sayı kısmı alınmaktadır.

Örnek 2.1-3:

```
using System;

namespace Operatorler
{
    class DortIslem
    {
        static void Main(string[] args)
        {
            int x = 10;
            int y = 4;

            int sonuc;
            float floatSonuc;

            sonuc = x / y;
            Console.WriteLine("x / y = {0}", sonuc);

            floatSonuc = (float)x / y;
            Console.WriteLine("x / y = {0}", floatSonuc);

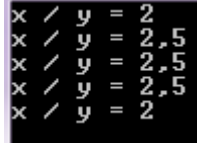
            floatSonuc = x / (float)y;
            Console.WriteLine("x / y = {0}", floatSonuc);

            floatSonuc = (float)x / (float)y;
            Console.WriteLine("x / y = {0}", floatSonuc);

            floatSonuc = (float)(x / y);
            Console.WriteLine("x / y = {0}", floatSonuc);

            Console.ReadKey();
        }
    }
}
```

Çıktısı:



Resim 2.3: Örnek 2.1-3 ekran çıktısı

Birinci çıktı: $\text{sonuc} = x / y = 10/4$ bir tam sayı bölme işlemi olduğundan bölümün tam sayı kısmı alınmıştır.

İkinci çıktı: $\text{floatSonuc} = (\text{float})x / y = (\text{float})10 / 4$ deyiminde, önce 10 sayısı float tipine dönüştürülüyor, sonra 4 sayısına bölünüyor. Bir float tipin bir tam sayıya bölümü yine float tipindendir. Dolayısıyla, $(\text{float})10 / 4 = 2.5$ 'tir.

Üçüncü çıktı: Bu çıktı ikinci çıktının simetriğidir. $\text{floatSonuc} = x / (\text{float})y = 10 / (\text{float})4$ deyiminde, önce 4 sayısı float tipine dönüştürülüyor, sonra 10 tam sayısı 4.0 sayısına bölünüyor. Bir tam sayı tipin bir float tipine bölümü yine float tipindendir. Dolayısıyla, $10 / (\text{float})4 = 2.5$ 'tir.

Dördüncü çıktı: İkinci ve üçüncü çıktının birleşimidir. $\text{floatSonuc} = (\text{float})x / (\text{float})y = (\text{float})10 / (\text{float})4$ deyiminde, önce 10 ve 4 sayılarının her ikisi de float tipine dönüştürülür. Sonra iki float tipin birbirine bölümü yapılır. Bu işlemin sonucu, doğal olarak bir float tipidir. Dolayısıyla, $(\text{float})10 / (\text{float})4 = 2.5$ 'tir.

Beşinci çıktı: $\text{floatSonuc} = (\text{float})(x / y) = (\text{float})(10/4)$ deyiminde, önce $(10 / 4)$ bölme işlemi yapılır. Bu bir tam sayı bölme işlemi olduğu için birinci çıktıda olduğu gibi çıkan sonuç 2 dir. $(\text{float})2 = 2.0000000$ olduğundan çıktı 2'dir.

2.1.2. Mod Alma

Bir sayının başka bir sayıya bölümünden kalan sonucu alma işlemine mod alma denir. Bu işlemi yapmak için % karakteri kullanılır.

Örnek 2.1-4:

```
using System;

namespace Operatorler
{
    class ModAlma
    {
        static void Main(string[] args)
        {
            double x = 10;
            double y = 4;

            double sonuc = x % y;

            Console.WriteLine("x % y = {0}", sonuc);
        }
    }
}
```

```

    Console.ReadKey();
}
}
}

```

Çıktısı :

$x \% y = 2$

Örnek 2.1-5:

```

using System;

namespace Operatorler
{
    class ModAlma
    {
        static void Main(string[] args)
        {
            int x, y;

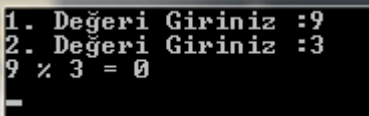
            Console.Write("1. Değeri Giriniz :");
            x = Convert.ToInt16(Console.ReadLine());
            Console.Write("2. Değeri Giriniz :");
            y = Convert.ToInt16(Console.ReadLine());

            Console.WriteLine("{0} % {1} = {2}", x, y, x % y);

            Console.ReadKey();
        }
    }
}

```

Çıktısı :



```

1. Değeri Giriniz :9
2. Değeri Giriniz :3
9 % 3 = 0
_

```

Resim 2.4: Örnek 2.1-5 ekran çıktısı

2.2. İlişkisel Operatörler

İlişkisel operatörler iki değerin karşılaştırılması işlemi için kullanılır. Programımızda koşul ifadelerinde kullanılarak programın akışını değiştirmemizi sağlar. Karşılaştırma sonucunda doğru(true) ve yanlış(false) olmak üzere boolean bir değer döndürür.

Örneğin, günlük hayattan bir örnek verelim.



Şema 2.1: İki değerin karşılaştırılması

Hava yağmurlu mu sorusu bizim için bir şart ve bu şart içinde havanın durumunu yağmurla karşılaştırıyoruz. Bu soruya dışarıda yağmur yağıyorsa evet(true) yağmıyorsa hayır(false) şeklinde boolean bir cevap veriyoruz. Dolayısıyla verdiğimiz cevaba göre de programımızın akışı yön değiştirmektedir.

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

Tablo 2.2: İlişkisel operatör listesi

== Operatörü: Aynı türdeki iki değerin birbirine eşitliğinin kontrolü için kullanılan operatördür.

```
int x = 10;  
int y = 4;
```

```
string str1 = "megep";  
string str2 = "megep";
```

```
x == y // false  
str1 == str2 // true
```

```
3 == 3 // true
```

```
3 == "3" // hatalı kullanım. int tipi ile string tipi karşılaştırılamaz.
```

!= Operatörü: Aynı türdeki iki değerin birbirine eşit olmadığının(eşit değil) kontrolü için kullanılan operatördür.

```
int x = 10;
int y = 4;

string str1 = "megep";
string str2 = "megep";

x != y // true
str1 != str2 // false

3 != 3 // false
1 != 3 // true
```

> Operatörü: Bir değerin aynı türdeki başka bir değerden büyüklüğünün kontrolünün yapıldığı operatördür. Bu operatör string işlemlere uygulanmaz.

```
int x = 10;
int y = 4;

x > y // true
y > x // false

3 > 3 // false
5 > 3 // true
```

< Operatörü: Bir değerin aynı türdeki başka bir değerden küçüklüğünün kontrolünün yapıldığı operatördür. Bu operatör string işlemlere uygulanmaz.

```
int x = 10;
int y = 4;

x < y // false
y < x // true

3 < 3 // false
1 < 3 // true
```

>= Operatörü: Bir değerin aynı türdeki başka bir değerden büyük veya eşitliği kontrolünün yapıldığı operatördür. Bu operatör string işlemlere uygulanmaz.

```
int x = 10;
int y = 4;

x >= y // true
y >= x // false

3 >= 3 // true
```

<= Operatörü: Bir değerin aynı türdeki başka bir değerden küçük veya eşitliği kontrolünün yapıldığı operatördür. Bu operatör string işlemlere uygulanmaz.

```
int x = 10;
int y = 4;

x <= y // false
y <= x // true

3 <= 3 // true
```

2.3. Mantıksal Operatörler

Mantıksal operatörler birden fazla şartın olduğu durumlarda kullanılır. Birden çok boolean değeri tek bir boolean değere indirmek için kullanılır.

Operatör	Açıklama
&&	Ve
	Veya
!	Değil

Tablo 2.3: Mantıksal operatör listesi

&& Operatörü: ‘Ve’ anlamındadır. Sorgulanan tüm şartlar doğru(true) olduğu zaman doğru(true), şartlardan birinin yanlış(false) olması durumunda yanlış(false) değerini döndürür.

```
int x = 10, y = 4;
string str1 = "megep";

x == y && "megep" == str1 // 1. şart = false, 2. şart = true -> sonuç = false

x == 10 && y == 4 && true == true // 1. şart = true, 2. şart = true, 3.
şart = true -> sonuç = true
```

|| Operatörü: ‘Veya’ anlamındadır. Sorgulanan şartlardan birinin doğru(true) olması durumunda doğru(true), şartların hepsinin yanlış(false) olması durumunda yanlış(false) değerini döndürür.

```
int x = 10, y = 4;
string str1 = "megep";

x == y || "megep" == str1 // 1. şart = false, 2. şart = true -> sonuç = true

x == 4 || y == 10 || "megep" == str1 // 1. şart = false, 2. şart = false,
3. şart = true -> sonuç = true

x == 4 || y == 10 || true == false // 1. şart = false, 2. şart = false,
3. şart = false -> sonuç = false
```


! Operatörü: ‘Değil’ anlamındadır. ! işareti değeri tersine çevirir.

```
(!true) // sonuç = false  
(!false) // sonuç = true
```

2.4. İşlem Önceliği

İşlem öncelik sırası aşağıdaki tabloda en yüksekten en düşüğe doğru sıralanmıştır.

En Yüksek
()
!
*, /, %
+, -
<, >
=, !=
&&
En Düşük

Tablo 2.4: İşlem önceliği listesi

Yapılan işlemde yukarıdaki sıra tamamlandıktan sonra eğer aynı tür işlemler kaldıysa işlem soldan sağa doğru yapılır.

Örnek 2.4-1: $3+5*2$ işleminin sonucu nedir?

Yukardaki işlemde önce 3 ile 5’i toplar ve sonucu 2 ile çarparsanız sonuç yanlış çıkar. İşlem önceliğine göre önce 2 ile 5’i çarpıp çıkan sonuçla 3’ü topladığımız zaman sonuç doğru çıkar.

$3+5*2 = 8*2 = 16$ // yanlış cevap
 $3+5*2 = 3 + 10 = 13$ // doğru cevap

Örnek 2.4-2: $10/5*2$ işleminin sonucu nedir?

Yukardaki işlemde önce 5 ile 2’yi çarpıp ve 10’u çıkan sonuca bölerseniz yanlış cevap çıkar. Bölme ve çarpmanın işlem önceliği eş değer olduğu için işlem soldan sağa doğru işleyecektir.

$10/5*2 = 10 / 10 = 1$ // yanlış cevap
 $10/5*2 = 2 * 2 = 4$ // doğru cevap

Örnek 2.4-3: $(5+2)*4-6/2$ işleminin sonucu nedir?

$(5+2)*4-6/2 = 7*4-6/2 = 28-6/2 = 28-3 = 25$ // doğru cevap

UYGULAMA FAALİYETİ

Aşağıdaki işlemleri uygulayınız.

İşlem Basamakları	Öneriler
➤ Tanımlanan değişkenlere sayısal hesaplamalar yaparak değer aktarınız.	➤ Önce basit matematiksel işlemleri deneyebilirsiniz, yaşınızı hesaplamak gibi
➤ Elde edilen değeri ekranda gösteriniz.	➤ Sadece sonucu göstermek anlamlı olmaz. ➤ Mesela: "Yaşım: 16" gibi yazdırınız.
➤ Karmaşık matematiksel formüllerde gerekli yerlere parantez ekleyerek işlem önceliklerini belirleyiniz.	➤ Deneyeceğiniz formülleri hesap makinesi ve hesap tablosu programları ile deneyip sonucu karşılaştırınız. Güvenilir sonuçlar elde etmeye çalışınız.
➤ Metin birleştirme operatörü ile birden fazla metin değeri birleştiriniz.	

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Mod alma işlemini aşağıdaki karakterlerden hangisi gerçekleştirir.
A) > B) ! C) % D) &
2. Aşağıdakilerden hangisi ilişkisel operatörler arasında yer almaz.
A) || B) <= C) != D) >
3. Ve mantıksal operatörü aşağıdakilerden hangisidir.
A) != B) == C) & D) &&
4. $(5+3)*4-6/2*3-(2*3)$ işleminin sonucu aşağıdakilerden hangisidir.
A) 33 B) 17 C) 25 D) 19

Aşağıda verilen cümlelerde boş bırakılan yerlere doğru sözcüğü yazınız.

5. $4 > 2$ işlemi sonucunu üretir.
6. Koşul ifadelerinin karşılaştırılmasında kullanılan operatörlere denir.
7. ... operatörü ‘değil’ anlamındadır. Değeri tersine çevirir.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise “Modül Değerlendirme”ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki soruyu dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdaki yerel değişken tanımlamalarından hangisi doğru yapılmıştır?

- A) int16 sayi B) metin ad C) byte yas D) float 12

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

2. () Tanımlanan bir değişkene, ancak tanımlandığı blok içinde ulaşılabilir.

3. () ogrNo ile OgrNO aynı hafıza bölgesini temsil eder.

4. () Birden fazla şartın olduğu durumlarda aritmetiksel operatörler kullanılır.

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

5. Aşağıdakilerden hangisi referans tipler arasında yer alır?

- A) string B) float C) double D) bool

6. Aşağıdaki atama operatörlerinden hangisi doğru kullanılmıştır?

- A) a= = 5 B) a+- C) a++= 2 D) a+= 3

7. Aşağıdaki yazımlardan hangisi doğrudur?

- A) Console.WriteLine("Boyunuz : {0} Kilonuz : {1}", boy, kilo);
B) Console.WriteLine("Boyunuz : boy Kilonuz : kilo");
C) Console.WriteLine("Boyunuz : [0] Kilonuz : [1]", boy, kilo);
D) Console.WriteLine("Boyunuz : {0} Kilonuz : {1} " + boy + kilo);

8. Console.Write("{0:C}", 3000); komutunun ekran çıktısı aşağıdakilerden hangisidir?

- A) 3.000 TL B) 3000 TL C) 3.000,00 TL D) 0003 TL

9. 10 % 3 işleminin sonucu kaçtır?

- A) 4 B) 1 C) 3 D) 2

10. (8/2*4)+4-10*2 işleminin sonucu kaçtır?

- A) 0 B) -15 C) -4 D) 15

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ 1 CEVAP ANAHTARI

1	D
2	B
3	A
4	D
5	A
6	C
7	B
8	Console.Write() Console.WriteLine()
9	\n
10	Try ..Catch

ÖĞRENME FAALİYETİ 2CEVAP ANAHTARI

1	C
2	A
3	D
4	B
5	true
6	İlişkisel
7	!

MODÜL DEĞERLENDİRME CEVAP ANAHTARI

1	C
2	DOĞRU
3	YANLIŞ
4	YANLIŞ
5	A
6	D
7	A
8	C
9	B
10	A

KAYNAKÇA

- <http://www.ecma-international.org/>
- <http://www.gorselprogramlama.com/> (15.03.2011- 12:00)
- <http://www.yazilimutfagi.com/> (15.03.2011- 12:00)
- <http://www.csharpnedir.com/> (15.03.2011- 12:00)
- <http://www.programmersheaven.com/> (15.03.2011- 12:00)
- http://tr.wikipedia.org/wiki/Ana_Sayfa