

C#

Console, SQL Database

Eğitimci/Yazar-Mehmet Akif Arvas

Bu kitabın her türlü yayın hakkı, Mehmet Akif Arvas'a aittir.

Yazarın Özgeçmişİ

Mehmet Akif Arvas

Elektronik ve bilgisayar alanında; Lise eğitimimi, Van Endüstri Meslek Lisesinde tamamladıktan sonra Üniversite lisans eğitimimi, Pamukkale Üniversitesi TEF Elektronik ve Bilgisayar Eğitimi ABD, Elektronik Öğretmenliği Bölümünü bitirip, iş hayatına öğretmene olarak başladım. İlk kitabı, **Seçkin Yayıncılık, Arduino ile Robotik Programlama** alanında yazdım. İkinci kitabı olan, **Arm İşlemci Programlama** ile mühendislik alanında yapılacak olan projelere kaynak referans bir kitap olmayı hedefledim.

C# kitabını, değerli okuyucularıma, robotik alanında seri porta aktarılacak arayüz programlarını, kendilerinin yapabilecek seviyeye gelmesi için istifadenize ücretsiz olarak sunuyorum.

Elektronik ve Bilgisayar alanında bugüne kadar yapmış olduğum;

Arm işlemci Programlama ve Gömülü Sistem Geliştirme,

Akıllı otomasyon sistemleri,

Karth/Mobil giriş çıkış sistemleri,

Mobil Uygulamalar(Play Store Yayımcı Adı: Mrha),

Uzaktan Cihaz Kontrol Sistemleri (RF, IR, Bluetooth ve Wifi ile)

Robot kol tasarımları ve kodlaması,

Drone/İHA,

Özel istekli ve amaçlı elektronik devre tasarımları,

3 boyutlu yazıcı,

3 Boyutlu Cnc Router,

PCB/Lazer kesim ve çizim makinesi

Web Tasarım ve Programlama (Asp.net ile),

v.b çeşitli projeler yapmakla beraber, yine aynı alanda proje desteği sundum. 4 yıl özel bir okulda öğretmen olarak çalışmaktan sonra, Kurumsal firmalara, Makine bakım onarım desteği sundum. Aynı zamanda Elektronik ve Bilgisayar alanında, kendi firmamı kurdum(Akıllı Otomasyon sistemlerin üretimi ve satışı). 1 yıl kadar kendi işyerimi işlettikten sonra, tekrar eğitici olarak görev'e başladım. Halen görevimi eğitici olarak devam etmekteyim.

Yazarın Kitapları

- Projeler Eşliğinde Arduino İle Robotik Programlama – Mehmet Akif Arvas, Seçkin Yayıncılık
- Projeler Eşliğinde Arm İşlemci Programlama – Mehmet Akif Arvas, Seçkin Yayıncılık

Uzmanından Öğrenin, Rahat Edin!

Projeler Eşliğinde

Arduino ile Robotik Programlama

Sensörler – Veri İletişimi – Wifi, Bluetooth, IR

Mehmet Akif Arvas



66 Adet Denenmiş ve Çalıştırılmış Uygulama Projesi ile

Kitaptaki program kodlarını seckin.com.tr'den indirebilirsiniz.

Uzun yıllar, elektrik, elektronik, mekanik bakım ve onarım konusunda çalışmalarında bulunan ve büyük bir markanın üretim bandında da görev alan yazar aynı zamanda çeşitli eğitim kurumlarında Robotik, Arduino ve Temel Elektronik konularında eğitimler veren yazar, bu eğitimlerde edindiği bilgi ve tecrübeleri kaleme almıştır.

Kitapta; Arduino yazılımı ve projeleri, Elektronik kartları, robotik/otomasyon alanında yapılan uygulamalar, ayrıntılı bir biçimde çok sayıda görsel eşliğinde anlatılmıştır. Eserde bizzat uygulanmış ve çalıştırılmış 66 farklı proje anlatılarak konunun sadece teorik kısmına yer verilmemiş aynı zamanda uygulamasına da yer verilmiştir.

Kitabı bitirdiğinizde, konuya ilgili özgün projeler üretebilmeniz ve bu projelerinizi yapabilecek seviyeye gelmeniz hedeflenmiştir.

Kitapta yer alan denenmiş ve çalıştırılmış 66 projenin uygulama kodlarını www.seckin.com.tr'den indirebilirsiniz..

Kitapta Bulunan Konu Başlıkları

- ISIS, Fritzing, Eagle, Tinkercad Simülasyon Programları Kullanımı
- Led, Button ve Röle Kullanımı
- LCD Ekran Kullanımı
- Sıcaklık Sensörü Kullanımı
- Işık Sensörü Kullanımı
- RFID Kart Okuyucu Kullanımı
- Servo Motor, DC Motor, Brushless (Fırçasız) Motor Kullanımı
- IR Alıcı ve Verici Kullanımı
- Otomatik Vantilatör Çalıştırma
- Bluetooth ile Uzaktan Kumanda
- WIFI (ESP8266) Kullanımı
- IOT Nesnelerin İnterneti

Künye Bilgileri: 2018 Baskı | 16x24 cm. | 223 Sayfa | 34,90 TL | 9789750249778



SEÇKİN YAYINCILIK

Eskişehir Yolu, Mustafa Kemal Mah. 2158. Sokak No:13 Çankaya/ANKARA

Tel: 0-312-435 30 30 | Faks: 0-312-435 24 72 - satis@seckin.com.tr

Uzmanından Öğrenin, Rahat Edin!

Projeler Eşliğinde

Arm İşlemci Programlama

Arm Mbed OS, RTOS, Thread, RTC, Multi Tasking

<https://www.seckin.com.tr/kitap/926844928>

Mehmet Akif Arvas

Tüm elektronik, yapay zeka ve kontrol ünitelerinde Arm işlemci kullanılabilmektedir.



Uygulanmış 100 Proje Eşliğinde

Kitapta Arm yapısının karmaşıklığını gideren ve kullanıcıya basit arayüz ve sade anlaşılır kod yapısını kullanarak projeler gerçekleştirmeye yarayanMBED OS online platformu kullanarak konular anlatılmaya çalışılmış olup, bizzat uygulanmış ve çalıştırılmış 100 farklı proje anlatılarak konunun sadece teorik kısmına yer verilmemiş aynı zamanda uygulamasına da yer verilmiştir.

Yine aynı şekilde C++ ile yazılmış kod yapıları da sade ve basit bir dille anlatılmaya çalışılmıştır. Kitaptaki örnek projelere bakarak kendi özgün projenizi oluşturabilir veya bu projeleri kendi sisteminizde kullanabilirsiniz.

Kitapta yer alan denenmiş ve çalıştırılmış 100 projenin uygulama kodlarını www.seckin.com.tr'den indirebilirsiniz.

Kitapta Bulunan Konu Başlıkları

- Gömülü Yazılım Geliştirme
- Cortex Mikrodenetleyici Yazılım Arayüzü Standardı (CMSIS)
- Online Arm Mbed OS
- Sensör Kullanımları
- Managing tasks (Görevleri Yönetme)
- RTOS (Real Time Operation System)
- Thread (İş Parçacıkları) Oluşturma
- Dosya Veri Yazma
- Okuma İşlemleri
- Kablolu ve Kablosuz Veri Aktarma
- Mühendislik Uygulamaları
- Aynı Anda Bağımsız Çalışan Uygulamalar
- RTC ve Güç Yönetimi

Künye Bilgileri: 2019 Baskı | 16x24 cm. | 295 Sayfa | 39,90 TL | 9789750252518



SEÇKİN YAYINCILIK

Eskişehir Yolu, Mustafa Kemal Mah. 2158. Sokak No:13 Çankaya/ANKARA

Tel: 0-312-435 30 30 | Faks: 0-312-435 24 72 - satis@seckin.com.tr

İçindekiler

Yazarın Özgeçmişİ	1
Yazarın Kitapları	2
Uzmanından Öğrenin, Rahat Edin!	3
Uzmanından Öğrenin, Rahat Edin!	5
C# Değişkenler	10
Değişkeni console uygulamamız içinde kullanma	11
Const Sabit Değişkeni.....	15
Değişkenler Uygulama Örneği	15
Tür Dönüşümleri	16
Operatörler.....	18
Mod Alma	19
Arttırma ve Azaltma	19
Mantıksal Operatörler	21
Karşılaştırma Operatörleri.....	22
as operatörü	23
is Operatörü	23
? : Operatörü.....	24
Bitsel Operatörler	25
Bitsel Kaydırma Operatörleri(<< , >>)	27
Operatörler Uygulama Örneği	28
C# Karar Kontrol Mekanizmaları.....	28
if else Kullanımı	28
İç içe yazılmış if else yapısı	30
if, else if, else Kontrol Yapısı	31
if else Uygulama Örnekleri	32
switch case Kullanımı	38
C# Döngü Yapıları	47
for Döngüsü	47
Continue komutu:.....	58
while Döngüsü	59
Break komutu:.....	61
do while Kullanımı	62
Karar Kontrol ve Döngülerle ilgili Uygulama Örnekleri	63
Diziler	67

foreach Kullanımı	70
Çok Boyutlu Diziler	71
Dizilerde Kullanılan Metotlar	74
GetLength() Metodu.....	74
Dizilerde Kopyalama Metodu	75
CreateInstance, Dizi Sıralama ve Dizi Arama Metotları	75
ArrayList Add Metodu	78
ArrayList Insert Metodu	78
Hata Yakalama Mekanizması(try,catch,finally)	79
Rastgele Sayı Üretme	82
Metotlar	82
Main Metodu.....	84
Metotlar İle İlgili Özellikler.....	84
Params Metodu.....	87
Aynı Metot Adına Sahip, Değişken Türleri Farklı Olan Metotlar	89
Özyineli (Reküratif-Recursive) Metotlar	91
Metinsel Metotlar	92
Compare() Metodu	93
Concat() Metodu	93
Format() Metodu	94
IsNullOrEmpty() Metodu	95
Contains() Metodu.....	95
EndsWith() ve StartsWith () Metotları.....	96
ToUpper() ve ToLower() Metotları	97
IndexOf()ve LastIndexOf Metodu	97
PadLeft () ve PadRight () Metodu.....	98
Remove () Metodu	99
Replace() Metodu.....	99
Split() Metodu	100
Trim() Metodu	100
Matematiksel Metotlar.....	101
Sqrt() Metodu.....	101
Sin(), Cos(), Tan(), Abs() Metodları	102
Diğer Matematiksel Metotlar	102
Tarih/Saat (DateTime) Metotları.....	103

Dosya Giriş Çıkış İşlemleri	104
Dosya Açma	104
Dosya Silme	105
FileStream ile Dosya Oluşturma	105
Dosyaya Veri Yazma ve Güncelleme	106
Dosyadan Veri Okuma	107
Dosyada Veri Arama	108
Console Ekranında Renklendirme İşlemleri	109
ForegroundColor ile Yazıyı Renklendirme	109
BackgroundColor ile Arka Plan Renklendirme	109
Veri Tabanı İşlemleri	110
SQL Programlama Dili	112
SQL Komutları	112
Select Komutu	112
INSERT Komutu	118
UPDATE Kullanımı	121
DELETE Komutu	122
Veritabanı Uygulama Örneği	123

C# Değişkenler

Değişkenler, programlama yaparken ifadelerin hangi türden olacağını belirtmemiz için kullanılan veri tipleridir. Örneğin sayılar için bir tanımlama yapılırken bu tanımlanacak olan sayının bir tamsayımı yada bir kesirli sayımı olacağını farklı değişken türlerini kullanılarak tanımlama yapılır. Bunu şöylele izah edebiliriz doğadaki herşey bir cisim, maddedir. Bu maddeleri isimlendirirken bu maddeleri ayırt etmek için her birine farklı bir isim bırakılmıştır. Ör. Ağaç, taş, kaya, dağ, yağmur v.s. gibi.

Csharp programlama dilinde veri tipleri tanımlanırken aşağıdaki değişken türleri kullanılır.

Tür	Boyun	Açıklama	Kapasite
sbyte	1 Byte	Tamsayı	-128 ile 127
short	2 Byte	Tamsayı	-32.768 ile 32.767
int	4 Byte	Tamsayı	-2.147.483.648 ile 2.147.483.647
long	8 Byte	Tamsayı	-9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807
byte	1 Byte	Tamsayı	0 ile 255
ushort	2 Byte	Tamsayı	0 ile 65.535
uint	4 Byte	Tamsayı	0 ile 4.294.967.295
ulong	8 Byte	Tamsayı	0 ile 18.446.744.073.709.551.615
float	4 Byte	Reel sayı	+— 1,5*10-45+— 3,4*1038
double	8 Byte	Reel sayı	+— 5*10-324+— 1,7*10308
decimal	16 Byte	Reel sayı	+— 1,5*10-28+— 7,9*1028
bool	—		true ya da false
char	2 Byte		
String	Sınırsız		
DateTime	8 byte	Tarih ve Zaman Tutar	

Değişken tanımlamaları şu şekilde yapılmaktadır.

(Değişken Türü) (Değişen Adı) = (Değeri)

```
int x=2343243;
double y=254.255;
float z=987.55f;
string m="Programlama Temelleri";
char t='a';
```

bool

Koşullu yapılarda kullanılır. Bool türünden değerlere true yada false gibi örnek verilebilir.

Örnekler:

```
bool x = true;
```

```
bool y = false;  
bool z = 5>2;
```

object

Bu değişken türüne her türden veri atanabilir. Akliniza şu soru gelmiş olabilir; Madem her türden veri ataması yapılabiliyor. Neden farklı farklı değişken türlerini kullanıyoruz. Cevap olarak;

- Object değişken türünün boyutunu yüksek olması.
- Her yerde object değişkenin kullanılması durumunda program boyutunun artması.
- Kod satırlarının boyutu arttığından dolayı işleyişide yavaş olacaktır.

Bir yazılım ne kadar az kod satırlarından ve ne kadar az bir boyut kaplıyorsa çalışma hızı verimliliği artmış olacaktır. Apple ile Windows, Android diğer işletim sistemleri ayıran farkta bu noktada farkını ortaya çıkarmaktadır. İşletim sistemlerinin kullanmış olduğu donanım parçaları aynı farklı kodlanmış olan yazılım dilinde.

```
object x = "asfdsfdsf????--";
```

```
object x = 25456.66;
```

int **x;**
Değişken **Değişken**
türü **adi**

Değişken Adı Verirken Dikkat Edilecek Durumlar

- ✓ Tanımlama yaparken büyük-küçük harf ayımı vardır.
- ✓ Değişken adları rakamla başlayamaz.
- ✓ Class, namespace ve kontrol isimleri gibi program tarafından kullanılan isimler verilemez.
- ✓ Aynı kod bloğu içerisinde aynı isimden birden fazla değişken tanımlanamaz.
- ✓ Özel karakter içermez örneğin /,*,-,+.
- ✓ Boşluk kullanılamaz.
- ✓ Türkçe karakter kullanılmamalıdır. Ş, ğ, ü, ö, ç, ı gibi...
- ✓ Özel sözcükler (if, else, random, vb.) kullanılmaz.

Değişkeni console uygulamamız içinde kullanma

Örnek:

```
namespace ConsoleApplication1  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int x = 54;  
            Console.WriteLine(x);  
            Console.ReadKey();  
        }  
    }  
}
```

```
        }
    }
}
```

Yukarıdaki örnekte console ekranına x'in değeri yani 54 yazıldığı görülecektir. Console.ReadKey yazılı olan kod satırında uygulamayı kapatmak için klavyeden herhangi bir tuşa basılmasını bekleyecektir. Herhangi bir tuşa basılmadığı sürece console ekranımız açık kalacaktır.

Örnek:

```
namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            char x = '?';
            string y = "C# Programlama";
            Console.WriteLine(x);
            Console.WriteLine(y);
            Console.ReadKey();
        }
    }
}
```

Yukarıdaki örnekte ekrana alt alta gelecek şekilde ? ve C# Programlama yazıldığı görülecektir. Eğer Console.WriteLine yerine Console.Write kullanmış olsaydık ekrana yazdırılacak olan ifadeler yan yana gelmiş olacaktı.

Not: Program çalıştırıldığında kodlar satır satır işlenir. Yapacağımız işi ne kadar az kod kullanarak yapabilirsek o kadar iş yükü azalmış olur. Buda yaptığımız uygulamanın hızlı çalışması az yer kaplaması gibi artı avantajlar sağlamış olacaktır. Az kod çok iş.

Şimdi yukarıda yapmış olduğumuz örneği tek satır kod kullanarak yazdırıralım.

```
namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            char x = '?'; // Değişken Tanımlama
            string y = "C# Programlama"; // Değişken Tanımlama
            Console.WriteLine(x+"\n"+y);
            Console.ReadKey(); // Klavyeden herhangi bir tuşa basılmasını bekle
        }
    }
}
```

Yukarıda da görüldüğü gibi bir satırlık kod parçasını silip aynı işlemi yapan \n kod parçasını kullanarak alt satıra yazdırma işlemini yapmış olduk. \n kodu kullanarak tüm alt satıra yazma işlemlerinde kullanabiliriz.

Ayrıca console.ReadKey kısmında // ile satır sonunda açıklama kısmı oluşturulmuştur. Bu açıklama kısmı kodsatırları işlenirken işleme tabi tutulmaz. İstediniz her türden veya dilden

açıklama satırı bu şekilde oluşturulabilir. Eğer birden fazla kod satırı alt alta kullanılacaksa aşağıdaki gibi kullanmak daha mantıklı olacaktır.

Örnek:

```
namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            char x = '?';
            string y = "C# Programlama";
            Console.WriteLine(x+"\n"+y);

            Console.ReadKey(); /* Klavyeden herhangi bir tuşa basılmasını bekle
            Birden fazla açıklama kısmı bu şekilde
            alt alta yazılabilir */
        }
    }
}
```

Örnek:

```
namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            string x;
            x = Console.ReadLine(); // Klavyeden okunan değeri x'in içerisinde at
            Console.Write("Klavyeden okunan değer: " + x);
            Console.ReadKey();
        }
    }
}
```

Yukarıda ki örnekte klavyeden okunan değeri tanımlanan x değişkenin içerisinde atamasını yaptık. Klavyeden okuma işlemi yapılacaksa tanımlanacak değişken türü string türünden olması gerekiyor. Başka bir değişken türüne klavyeden okuma işlemi yapılip atama yapılacaksa tür dönüşümleri konumuzda nasıl yapıldığına bakabilirsiniz.

Örnek (Hatalı kullanım):

```
namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            string x;
            int y;
            x = Console.ReadLine();
            y = x; //string değişkeni int değişkenine direk atılamaz.Hatalı

            Console.Write("y değişkenin değeri: " + y);
            Console.ReadKey();
        }
    }
}
```

Tanımlanan int y tam sayı değerinin içeresine doğrudan string ile tanımlanmış değer ataması yapılamaz. Yine burada yapılması gereken değişiklik tür dönüşümleri konumuzda nasıl yapılacağını izah edeceğiz.

Örnek (Hatalı kullanım):

```
namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            double x;
            int y, z="metin"; //hatalı z değişkeni kullanımı
            y = x; /*int y değerinin içeresine doğrudan double ile tanımlanmış değer
ataması yapılamaz.*/
            Console.WriteLine("y değişkenin değeri: " + y);
            Console.ReadKey();
        }
    }
}
```

Yine bu örnekte de tür dönüşümü yapılmadan hatalı bir atama yapılmıştır. Ayrıca tanımlanınan int z değerinin içeresine de metin ataması yapılmıştır. Bu kullanımda hatalıdır. Değişken türü int double float ...v.s. olacaksa sadece sayı ataması yapılması yada değişken türü metin olacaksa string yada object değişken türü kullanılması gerekir.

Örnek (Hatalı Kullanım):

```
namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            char x='1',y='2';
            int z=x+y; // hatalı
            double x = 325.85;//birden fazla aynı değişken adı kullanılamaz

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

Char yada string olarak tanımlanmış değişkenler ile matematiksel işlemler yapılamaz. Burada hatalı kullanım olsada programımızı çalıştırduğumızda ekranımıza bir sayı değeri çıkacaktır. Bu örnekte çıkacak sayı değeri 99 olarak ekranımıza yazacaktır. Peki neden bu sayı çıkıyor? Aslında her bir sayı, harf ve özel karakterlerin Unicode denen bir sayısal karşılığı vardır. İşte bu tür tanımlanan değerler ile hesaplama yaptığımızda Unicode değerleri ile işlem yapılır.

Ayrıca bir değişken adı birden fazla değişken türleri tanımlanırken kullanılamaz. Değişken adları tanımlanırken benzersiz olması gereklidir. Yukarıda örneğimizde x değişken adı iki yerde kullanılmış bu durum hatalıdır.

Const Sabit Değişkeni

Değerini heryerde aynı saklı tutan hiç değişmeyecek değişken türü tanımlamak isteğimizde değişken türünün başına const ifadesi eklenmesi gereklidir. Sabit değişken olarak tanımlanan ifade ilk atama ne bırakılmış ise o şekilde kalır. Mesela pi'nin değeri heryerde aynıdır. Olası değişimlerede karşı korumak sonradan değiştirmesini engellemek için sabit değişken olarak ifadenin değişken türünün başına const ifadesi eklenmesi gereklidir.

Örnek:

```
static void Main(string[] args)
{
    const char x = '9';
    const int y = 6, z = 12;
    const string metin = "Programlama";
}
```

Örnekler (Hatalı Kullanım):

```
class Program
{
    static void Main(string[] args)
    {
        const char x;
        x = '*'; //hatalı
    }
}
```

Yukarıdaki örnekte x değişkeni sabit olarak tanımlanıp aynı satırda değer ataması yapılmadığı için hatalı kullanım olacaktır. Değişken sabit olacağsa değerin tanımlandığı yerde ataması yapılması gereklidir.

```
class Program
{
    static void Main(string[] args)
    {
        double x = 9;
        const double y = x + 11; //hatalı kullanım
    }
}
```

Değişkenler Uygulama Örneği

Double türünden tanımlanan iki sayıyı toplayan, çikaran, çarpan, bölen uygulamasını yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
```

```

static void Main(string[] args)
{
    double sayi1=45, sayi2=20, toplama,cikarma,carpma,bolme;

    toplama = sayi1 + sayi2;
    Console.WriteLine("Toplama Sonucu= " +toplama);

    cikarma = sayi1 - sayi2;
    Console.WriteLine("Çıkarma Sonucu= " + cikarma);

    carpma = sayi1 * sayi2;
    Console.WriteLine("Çarpma Sonucu= " + carpma);

    bolme = sayi1 / sayi2;
    Console.WriteLine("Bölme Sonucu= " + bolme);

    Console.ReadKey();
}
}

```

```

file:///C:/Users/mehmet/documents/visual studio 2015/Projects/ConsoleApplic...
Toplama Sonucu= 65
Çıkarma Sonucu= 25
Çarpma Sonucu= 900
Bölme Sonucu= 2.25

```

Tür Dönüşümleri

Farklı türdeki tanımlı değişken türlerinin bazen birbirine ataması gerektiği durumlar olabiliyor. Örneğin string türünden tanımlı bir değişkenin içeriğindeki tanımlı “5”, “1254” v.b gibi sayıların int değişken türüne atamasını yapacağımız zaman direkt ataması yapılamıyor bunu değişken türlerinde bazı örneklerimizde hatalı kullanımdan bahsetmiştik. Bu değişken türlerinin birbirine nasıl değiştirilebileceğini en basit ve her yerde kullanılan metot ile gösteremeye çalışalım.

Tür	Metot Kullanımı
bool	Convert.ToBoolean(x)
byte	Convert.ToByte(x)
sbyte	Convert.ToSbyte(x)
short	Convert.ToInt16(x)
ushort	Convert.ToUInt16(x)

int	Convert.ToInt32(x)
uint	Convert.ToInt32(x)
long	Convert.ToInt64(x)
ulong	Convert.ToInt64(x)
float	Convert.ToSingle(x)
double	Convert.ToDouble(x)
decimal	Convert.ToDecimal(x)
char	Convert.ToChar(x)

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            string metin1, metin2;
            float sayi1, sayi2, sonuc;

            Console.Write("Birinci sayıyı girin= ");
            metin1 = Console.ReadLine();

            Console.Write("İkinci sayıyı girin= ");
            metin2 = Console.ReadLine();

            sayi1 = Convert.ToSingle(metin1);
            sayi2 = Convert.ToSingle(metin2);

            sonuc = sayi1 + sayi2;
            Console.WriteLine("Toplama sonucu= " + sonuc);

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

Klavyeden okuma yapabilmek için değişken türümüz string olması gerekiyor. Örnekte klavyeden okunan değer string ile tanımlanmış değişkenlerin içerisine ataması yapılmıştır. Atanan değerler metinsel olduğu için matematiksel ifadelerde kullanabilmek için sayısal verilere dönüştürülmesi gerekiyor. Bunun içinde convert metodu kullanarak metinsel ifadeleri float türünden sayısal ifadeye dönüşümü sağlandı.

Not: Yukarıdaki örneğimiz için klavyeden girilen değerin sayı olması gerekiyor. Eğer sayı haricinde herhangi bir şey girilirse hata mesajı verecektir. Hatalı durumlarda ne gibi bir yapı kullanılacağını ileriki konularımızda göreceğiz.

Örneğimizi daha kısa ve az kod satırı kullanarak sadeleştirelim.

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            float sayi1, sayi2, sonuc;
            Console.Write("Birinci sayıyı girin= ");
            sayi1 = Convert.ToSingle(Console.ReadLine());

            Console.Write("İkinci sayıyı girin= ");
            sayi2 = Convert.ToSingle(Console.ReadLine());

            sonuc = sayi1 + sayi2;
            Console.Write("Toplama sonucu= " + sonuc);
            Console.ReadKey();
        }
    }
}

```

Klavyeden okunan değer convert.tosingle ile sayı değerine çevrilip direkt sayıl değişkenin içerişine ataması yapıldı. Ayrıca burada string tanımlamaya gerek kalmadan tek satırda hem okuma hemde atama işlemi yapıldı. Böylelikle 3 kod satırı yazılmayıp programımızda gereksiz kod yığını oluşmasının önüne geçilmiş oldu.

Operatörler

Programlama mantığında tek başlarına bir işe yaramayın kullanılan yerlerde çeşitli görevler üstlenen karakterlere operatör denir. Operatörlerin etki ettikleri sabit ya da değişkenlere ise operand denir. Aşağıdaki tabloda kullanılan operatör karakterleri verilmiştir.

Arttırma Azaltma	x++, x--, --x, ++x
Carpma, Bölme Mod Alma	*, /, %
Toplama ve Çıkarma	+, -
Kaydırma Operatörleri	<<, >>
İlişkisel ve Tür Testi Operatörleri	<, >, <=, >=, is, as
Atama ve işlemli atama operatörleri	=, *=, /=, %=, +=, -=, <<=, >>=, &=, ^=, =
Eşitlik Operatörü	==, !=
Mantıksal Ve	&&
Mantıksal Veya	
Koşul Operatörü	?:
Ve (AND)	&
Özel Veya (XOR)	^
Veya (OR)	
Tek Operand alan Operatörler	+, -, !, ~

Operatör kullanımıyla ilgili bazı örnek uygulamalar yapalım.

Mod Alma

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            x = 5 % 2;

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

Modüler aritmetikte işlem tanımlanırken;

$$a \equiv b \pmod{m}$$

şeklinde gösterim yapılır. a 'nın m 'ye bölümünden kalan sonuç b 'dir. Matematikte bu şekilde gösterim yapılrken yazılım kısmında bunu karşılığı yukarıdaki örneğimiz gibi kullanılır. $x=5\%2$ ifadesi: 5'in 2'ye bölümünden kalan sonuç x değişkenin içerisinde atama yapılmıştır. Ekrana yazdırılacak sonuç 1 olacaktır.

Arttırma ve Azaltma

Bu operatör türünü birçok yerde kullanıldığını göreceğiz. C#'ın en çok kullanılan operatördür diyebiliriz. Sayılarda arttırma ve azaltma yapmak için kullanılır. 2 farklı kullanımı vardır. Bunu örneklerle kullanımını görelim.

Örnek (Arrtırma):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5,y;
            y= x++; //x'in değerini 1 arttır sonraki işlemde kullan
            Console.WriteLine(y);
            Console.ReadKey();
        }
    }
}
```

```
    }  
}
```

Buradaki kullanımda y değişkenin içerisinde x'in ilk değeri atanıp daha sonra x'in değeri 1 arttırılır sonraki işlemde kullanılmak üzere x'in yeni değeri atanmış olur. Program çalıştırıldığında ekrana 5 değeri yazdırılacaktır.

Aynı örneği aşağıdaki şekilde yaptığımızda:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace ConsoleApplication3  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int x = 5,y;  
            y= ++x; //Önce x'i artır sonra atama yap  
            Console.WriteLine(y);  
            Console.ReadKey();  
        }  
    }  
}
```

Ekrana yazdırılacak sayı bu kez 6 olacaktır. Buradaki kullanımda önce x'in değeri artırılıyor. Daha sonra y değişkenin içerisinde x'in son halinin değerinin ataması yapılıyor. Aslında ikiside 1 artırma yapıyor. İlk örneğimizde sayı artırılıyor daha sonraki işlemde kullanılmak üzere ataması yapılıyor. Artırma operatörü başa geldiğinde hemen artırma yapıp atamasında aynı anda yapıyor.

Örnek (Azaltma):

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace ConsoleApplication3  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int x = 60,y;  
            y= x--;//x'in değerini y değişkenine ata, sonra 1 azalt  
            Console.WriteLine(y);  
            Console.ReadKey();  
        }  
    }  
}
```

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

Bu kod satırında öncelikle x'in değeri y değişkenine aktarılır. Daha sonra x'in değeri 1 azaltılır sonraki işlemde kullanılır. Bu örnekte ekranda gösterilecek sayı 60 olacaktır.

Aynı örneği aşağıdaki gibi değiştirdiğimizde:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 60, y;
            y= --x;//Önce x'in değerini azalt sonra atama yap
            Console.WriteLine(y);
            Console.ReadKey();
        }
    }
}
```

y= --x;

Bu kod satırında öncelikle x'in değeri 1 azaltılır sonra y değişkenine aktarılır. Azaltma ve atama aynı zamanda yapılır. Bu örnekte ekranda gösterilecek sayı 59 olacaktır.

Mantıksal Operatörler

Mantıksal işlemlerin ve(&&), veya(||), değil(!) kontrollerini sağlamada kullanılır.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            bool x = 10 > 5 && 20 == 20; //True
            bool y = 125 > 250 && 45 != 50; //false
            bool z = 2.5 is double || 25 == 200; //true
            bool t = 32 > 4 || 5 == 9; //true
            bool d =! (3 < 4); //false

            Console.Write("{0}\n{1}\n{2}\n{3}\n{4}", x, y, z, t, d);
            Console.ReadKey();
        }
    }
}
```

```
}
```

Program çalıştırıldığında ekrana True False True True False alt alta gelecek şekilde sonuç değeri verilir.

Karşılaştırma Operatörleri

Değişken türlerinden bahsederken bool türünü daha önce işlevinden bahsettik. Burada karşılaştırma operatörlerini kullanırken bool değişken türünü kullanarak işlevi yerine getireceğiz.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            bool x = 51 < 65;
            Console.Write("{0}",x);
            Console.ReadKey();
        }
    }
}
```

```
bool x = 51 < 65;
```

Karşılaştırma kısmında 65 sayısı 51'den büyük olduğundan program çalıştırıldığında ekrana true yazacaktır.

```
Console.Write("{0}",x);
```

Bu kod satırında süslü parantez içeresine yazılmış 0 değeri x'in içerisindeki ifadeyi alıp bunun içeresine atama yapmadır.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            bool x = 51 < 65;
            bool y = 3 > 4;
            bool z = 7 <= 7;
            bool t = 9 >= 12;
        }
    }
}
```

```

        Console.WriteLine("{0}\n{1}\n{2}\n{3}", x, y, z, t);
        Console.ReadKey();
    }
}

```

```
Console.WriteLine("{0}\n{1}\n{2}\n{3}", x, y, z, t);
```

Program çalıştırıldığında ekranın alt alta gelecek şekilde True False True False yazacaktır. Bu kod satırındaki süslü parantezler içerisinde yazılan değerler sırasıyla x,y,z,t değerlerinin içerisinde erişim yapabilmesi için tanımlanmıştır.

as operatörü

Object değişkeninden string değişken türüne dönüşüm yapmak için kullanılır.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            object x = "Programlama Teknikleri";
            string s = x as string;
            Console.WriteLine(x);
            Console.ReadKey();
        }
    }
}

```

is Operatörü

Verilen değişkenin türünü kontrol eder. Tür kontrolü yaparken bool değişkeni kullanılır. İfadeden sabit olması yada türünün farklı olması durumunda true yada false değerleri üretilmiş olur.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {

```

```

        float x = 50.5f;
        bool y1 = x is float;
        bool y2 = x is double;
        bool y3 = 12 is uint;
        bool y4 = "mehmet" is string;
        bool y5 = 25.3f is int;

        Console.WriteLine("{0}\n{1}\n{2}\n{3}\n{4}", y1, y2, y3, y4, y5);
        Console.ReadKey();
    }
}

```

Program çalıştırıldığında ekrana True, False, False, True, False yazacaktır. Kod satırlarını inceleyelim:

```
bool y1 = x is float;
```

x değişkenimiz float türünden olduğu için True ifadesi yazılacaktır.

```
bool y2 = x is double;
```

x değişkenimiz double türünden olmadığı için False yazacakır.

```
bool y3 = 12 is uint;
```

x değişkenimiz uint türünden olmadığı için False yazacakır.

```
bool y4 = "mehmet" is string;
```

“mehmet” metinsel ifade string türünden olduğu için True yazacakır.

```
bool y5 = 25.3f is int;
```

25.3f tanımlı ifade int olmadığı, float türünden olduğu için False yazacakır.

? : Operatörü

Koşullun doğru olduğunda hangi işlemlerin yapılabacağı yanlış olduğunda ne yapacağını belirleyen operatör türüdür.

Koşul	?	Doğru olmasındaki değer ne olacağı	:	Yanlış olmasındaki değer ne olacağı
devam==	"e" ?	“Loading...”	:	“İptal Edildi”

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)

```

```

    {
        string devam;
        Console.Write("Sistem Yüklemesi Yapılsın mı? (e, h): ");
        devam = Console.ReadLine();

        Console.WriteLine(devam == "e" ? "Loading..." : "İptal Edildi.");
        Console.ReadKey();
    }
}

```

Program çalıştırıldığında sadece e harfine basıldığında Loading ekrana yazdırılacak diğer bütün giriş durumlarında ekrana İptal edildi yazdırılacaktır.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string devam;
            Console.Write("Sistem Yüklemesi Yapılsın mı? (e, h): ");

            devam = Console.ReadLine();

            Console.WriteLine((devam == "e") || (devam == "y") ? "Loading..." : "İptal Edildi.");
            Console.ReadKey();
        }
    }
}

```

Program çalıştırıldığında sadece e veya y harfine basıldığında Loading ekrana yazdırılacak diğer bütün giriş durumlarında ekrana İptal edildi yazdırılacaktır.

Bitsel Operatörler

İkili sayı sistemine göre işlem yapan operatör türüdür. İkili sayı sistemi 1 ve 0'lardan oluşur. Kullanılacak değişken tamsayı olması gereklidir.

Örneğin 2 sayısının ikili sayı sistemindeki karşılığı 00000010 makine kodudur. Aslında bilgisayarda işlenen verilerin hepsi bu şekilde makine koduyla işlenir. Makine kodunu bilmek bize birçok avantaj sağlar. Elektronik alanında bu ikili sayı sistemi çokça kullanılmaktadır. Arduino, Raspberry, Pic Microchip programlamada bu ikili sayı sistemi bilmek gereklidir. Mesela Pic microşlemcinin herhangi bir bacağını kullanacağımız zaman bitsel işlemler

şeklinde ifade edilerek kontrol sağlanır. Bir sayının ikili sayı sistemindeki kod karşılığını bazı örneklerle tablo halinde gösterelim.

Sayı	Binary(ikili Sayı) Sistemi
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
15	00001111

Aşağıdaki tabloda iki binary sayının karşıtlı işlenmesinde kullanılacak tablo verilmiştir.

Ve(&)			Veya()			Özel Veya(^)		
A	B	Sonuç	A	B	Sonuç	A	B	Sonuç
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 4 & 2; // 00000100 & 00000010 -->sonuç 00000000 çıkar. Sonuç 0 çıkar.
            int y = 8 | 5; // 00001000 | 00000101 -->sonuç 00001101 çıkar. Sonuç 13 çıkar.
            int z = 9 ^ 6; // 00001001 ^ 00000110 -->sonuç 00001111 çıkar. Sonuç 15 çıkar.
            byte d = 5;

            byte e = (byte)~d; // 00000101'in değil -->sonuç 11111010 çıkar. Sonuç 250 çıkar.

            Console.WriteLine("{0}\n{1}\n{2}\n{3}\n{4}", x, y, z, d, e);
            Console.ReadKey();
        }
    }
}
```

Dönüşümün nasıl yapıldığını tablo üzerinden açıklayalım.

Sayı	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0

3	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	0
50	0	0	1	1	0	0	1	0

Not: 0x... Şeklinde olan sayılar 16'lık düzende yazılmıştır. Aynı zamanda bu kullanım en çok Microişlemcilerin programlaşması yapılırken hangi bacağını kullanacağımızı belirtmemizde de kullanılır.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 0xf0;
            Console.Write(x); //Ekrana 240 yazacaktır.
            Console.ReadKey();
        }
    }
}
```

Bitsel Kaydırma Operatörleri(<< , >>)

Sayıların bit karşılığı olan ikili sayı sistemini sağa sola kaydırmak için kullanılır

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            byte x = 50; // 00110010
            byte y = (byte)(x << 1); // x bir kez sola kaydırıldı 01100100. Sonuç 100

            byte z = 50; // 0 0 1 1 0 0 1 0
            byte d = (byte)(z >> 2); // d iki kez sağa kaydırıldı ve 00001100. Sonuç 12

            Console.WriteLine(y + " " + d);
            Console.ReadKey();
        }
    }
}
```

Operatörler Uygulama Örneği

Klavyeden girilen 2 sayıyı bölüp, çıkan sonuç 1 arttırıp oluşan yeni değeri 100 den büyük ve 120'ye eşit değilse ekrana true yazdırılsın, değilse false yazdırılsın. İşlem devamında bolum sonucunun bitsel olarak bir sola kaydırılıp eğer sonuç 100'e eşitse "Kaydırma işlemi 100'e eşit" değilse "Kaydırma işlemi 100'e eşit değil" yazısını ekrana yazdırılsın.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace operator_ugulama
{
    class Program
    {
        static void Main(string[] args)
        {
            double sayi1, sayi2, bolum;
            Console.Write("1. Sayıyı Giriniz= ");
            sayi1 = Convert.ToDouble(Console.ReadLine());

            Console.Write("2. Sayıyı Giriniz= ");
            sayi2 = Convert.ToDouble(Console.ReadLine());

            bolum = sayi1 / sayi2;
            bolum++;

            bool x = bolum > 100 && bolum != 120;
            Console.WriteLine(x);

            byte z = (byte)bolum;
            byte y = (byte)(z << 1);

            Console.WriteLine(y == 100 ? "Kaydırma 100'e eşit" : "Kaydırma işlemi 100'e eşit değil");
            Console.ReadKey();
        }
    }
}
```

C# Karar Kontrol Mekanizmaları

Operatörler konusunda ?: aslında bir karar kontrol mekanizması da denebilir. Ama yapabileceğim tek bir basit kontrol ifadesi vardır. Birden fazla işlem ve karmaşık yapımlarda bu kontrol yapısı kullanılamamaktadır. Ve bu ana kadar istediğimiz kod satırları olumsuz bir durumda neler olacağı ile ilgili bir sonuçta kararsız kalacaktır. Olumlu ve olumsuz durumda hangi işlemlerin yürütüleceğini işte tam da burada bu görevi yerine getirecek yapılardan bahsedeceğiz.

if else Kullanımı

Eğer koşul doğrusa if koşuluna bağlı ifadeleri çalıştırılacağı, koşulun sağlanmadığı durumlarda else yapısına bağlı komutların çalıştırılacağı anlamına gelen kontrol mekanizmasıdır. Bir başka deyişle olumlu durumda if yapısı olumsu durumda else yapısı devreye girecektir.

```
if (Koşul)
{
    Komutlar;
}
else
{
    Komutlar;
}
```

Not: Eğer if ve else yapısının hemen altında tek bir komut çalıştırılacaksa süslü parantez ile açılıp kapatılmasına gerek yok. Kodlarımızı tek bir komut için şu şekilde yazabiliriz:

```
if (Koşul)
    Komut;
else
    Komut;
```

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5;
            if (x == 5)
                Console.WriteLine("x değişkeni 5 sayısına eşittir");
            else
                Console.WriteLine("x değişkeni 5 sayısına eşittir");
            Console.ReadKey();
        }
    }
}
```

Program çalıştırıldığında ekrana x değişkeni 5 sayısına eşittir yazısı çıkacaktır. Çünkü x değişkenimiz 5 olarak tanımlanmış olup koşul x değişkeni 5 sayısına eşit mi ifadesi ile if yapısında kontrolü sağlanırken koşul sağlandığından hemen altındaki komutları çalıştıracaktır.

Not: if else kullanımında eğer olumsuz bir durumda kullanım olmayacağırsa else yazılmayabilir. else kullanımı zorunlu değildir. Örneğimizi else olmadan da şu şekilde sadece olumlu durumu belirtecek şekilde yazabilirez.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
```

```

    static void Main(string[] args)
    {
        int x = 5;
        if (x == 5)
            Console.WriteLine("x değişkeni 5 sayısına eşittir");
        Console.ReadKey();
    }
}

```

İç içe yazılmış if else yapısı

Birden fazla if ve else yapısı iç içe kullanılabilir.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Lütfen başlamak için Tamam yazın: ");
            string m = Console.ReadLine(); // Klaveden Okuma İşlemi Yapıldı.

            if (m == "Tamam") // m değişkeni Tamam ifadesine eşit mi?
            {
                Console.Write("Onay işlemi tamam. Yükleme işlemi yapılsın mı? ");
                string b = Console.ReadLine(); // Klaveden Okuma İşlemi Yapıldı.

                if (b == "Evet") // b değişkeni Evet ifadesine eşit mi?
                    Console.WriteLine("Loadind....");

                else
                /*
                    Evet'in haricinde herhangi birşey girilirse else devreye girip
                    programı bitirecektir.

                    */Console.WriteLine("Yükleme devam edilemedi....");
            }

            else // m değişkeni Tamam ifadesine eşit değilse burası devreye girecektir.
            {
                Console.WriteLine("Onay işlemi tamamlanamadı.");
                Console.WriteLine("Lütfen 10'dan büyük bir sayı giriniz");

                string c = Console.ReadLine(); // Klaveden Okuma İşlemi Yapıldı.

                int sayı = Convert.ToInt32(c); // Klavyeden okunan değeri int türüne dönüştürüldü.

                if (sayı > 10) // Sayı 10'dan büyükse işlemi yap
                {
                    sayı++;
                    Console.WriteLine("Girilen sayının değeri bir arttırıldı. Sonuç: " + sayı);
                }
            }
        }
    }
}

```

```
        }
        else
            Console.WriteLine("Girilen sayı değeri 10'dan küçük.");
    }
    Console.ReadKey();
}
}
```

Birden fazla if kullanımında bütün if satırları sağlanın yada sağlanmasın tek tek koşul denemesi yapılacaktır.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Lütfen sayı giriş yapınız: ");
            string m = Console.ReadLine(); // Klaveden Okuma İşlemi Yapıldı.
            int x = Convert.ToInt32(m);

            if (x == 10) // Eğer x eşitse
                Console.WriteLine("x :" + x);

            if (x > 21) // Eğer x büyüğe
                Console.WriteLine("x :" + x);

            if (x <= 5) // Eğer x küçük eşitse
                Console.WriteLine("x :" + x);

            if (x != 458) // Eğer x eşit değilse 458
                Console.WriteLine("x :" + x);

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

Yukarıdaki örnekte bütün if yapıları çalıştığı için girilen değer ekranda iki kez yazdırılacaktır. Sadece 458 girildiğinde aşağıdaki kod satırı koşul işlenmeyeceğinden x değişkeni ekrana 1 kez yazdırılacaktır. 458 değerinin haricinde bu kod satırı hep çalışacaktır.

```
if (x != 458) // Eğer x eşit değilse 458
```

if, else if, else Kontrol Yapısı

Eğer koşulumuz sağlandıysa diğer kontrollere bakmadan çıkış yapılması isteniyorsa o zaman aşağıdaki kontrol yapısı kullanılmalıdır.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Lütfen sayı girişini yapınız: ");
            string m = Console.ReadLine();
            int x = Convert.ToInt32(m);

            if (x == 10)
                Console.WriteLine("x :" + x);

            else if (x > 21)
                Console.WriteLine("x :" + x);

            else if (x <= 5)
                Console.WriteLine("x :" + x);

            else
                Console.WriteLine("Koşul sağlanamadı");
            Console.ReadKey();
        }
    }
}

```

İlk if koşulu her zaman çalışacaktır. Eğer bu ilk if koşulu sağlanmasa hemen altındaki else if koşuluna bakılacaktır. Bu koşul değeri sağlarsa diğer kontrol yapılarına bakılmadan program sonuç verip bitirecektir. Burada gereksiz kod satırı işlenmesinin önüne geçilmiş olmaktadır. Sürekli if yapısına kullanarak kontrol işlemi gerçekleştirmek gereksiz satır işlenmesine neden olacağı gibi programın yavaşlamışına da neden olacaktır. Unutulmamalıdır ki her kod satırı bir iş yükü demek. Bu iş yükünü ne kadar azaltılırsa programımız o kadar hızlı neticeler elde edecektir.

if else Uygulama Örnekleri

1. Örnek:

Klavyeden giren iki sayının aritmetik ortalamasını alıp, eğer sonuç 45'ten büyük ve eşitse Dersten başarılı bir şekilde geçtiniz, değilse Üzgünüm dersten başarılı olmadınız programını yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace if_else_uygulama

```

```

{
    class Program
    {
        static void Main(string[] args)
        {
            float sayi1, sayi2, sonuc;
            Console.Write("1. sınav notunu giriniz: ");
            sayi1 = Convert.ToSingle(Console.ReadLine());

            Console.Write("2. sınav notunu giriniz: ");
            sayi2 = Convert.ToSingle(Console.ReadLine());

            sonuc = (sayi1 + sayi2)/2;

            if (sonuc >= 45)
                Console.WriteLine("Dersten başarılı bir şekilde geçtiniz. Ortalamanız= "+sonuc);

            else
                Console.WriteLine("Üzgünüm dersten başarılı olamadınız.");
            Console.ReadKey();
        }
    }
}

```

Program ekran çıktısı:

```

file:///C:/Users/mehmet/Desktop/prg/if_else_ugulama/if_else_ugulama/bin/D...
ifElseUygulama
1. sınav notunu giriniz: 60
2. sınav notunu giriniz: 40
Dersten başarılı bir şekilde geçtiniz. Ortalamanız= 50

```

2. Örnek:

Kullanıcı adı ve şifre girişi yapıldığında eğer doğru kullanıcı bilgileri girilmişse Giriş yapıldı. Yanlış giriş yapılmış ise Hatalı kullanıcı adı veya şifre uyarısı yapılın.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace if_else_login
{
    class Program
    {
        static void Main(string[] args)
        {
            String kuladi, sifre;
            Console.Write("Kullanıcı adını giriniz: ");
            kuladi = Console.ReadLine();

            Console.Write("Şifre giriniz: ");
            sifre = Console.ReadLine();

            if (kuladi=="mehmet" && sifre=="123456")
                Console.WriteLine("Giriş yapıldı.");

            else
                Console.WriteLine("Hatalı kullanıcı adı veya şifre");
            Console.ReadKey();
        }
    }
}

```

```

file:///C:/Users/mehmet/Desktop/prg/if-else-login/if-else-login/bin/Debug/if...
Kullanıcı adını giriniz: mehmet
Şifre giriniz: 999
Hatalı kullanıcı adı veya şifre

```

Bu örneğimizin biraz daha gelişmiş halini aşağıdaki gibi kullanabiliriz.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace if_else_login_1
{
    class Program
    {
        static void Main(string[] args)
        {
            string kuladi, sifre;
            Console.Write("Kullanıcı adını giriniz:");

```

```

kuladi = Console.ReadLine();

if (kuladi == "mehmet")
{
    Console.WriteLine("Kullanıcı adı doğru");
    Console.Write("Şifre giriniz:");
    sifre = Console.ReadLine();

    if (sifre == "123456")
        Console.WriteLine("Giriş yapıldı");

    else
    {
        Console.Write("Kalan şifre giriş kullanım hakkı 1. Şifre giriniz:");
        sifre = Console.ReadLine();

        if (sifre == "123456")
            Console.WriteLine("Giriş yapıldı");

        else
            Console.WriteLine("Hesabınız bloke edildi.");
    }
}
else
    Console.WriteLine("Kullanıcı Adı veya Şifre Hatalı");
}
}
}

```

3. Örnek

Klavyeden girilen sayının tek mi, çift mi olduğunu bulan programı yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace if_else_tek_cift
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayı;
            Console.Write("Sayı giriniz: ");
            sayı = Convert.ToInt32(Console.ReadLine());

            if (sayı%2 == 0)
                Console.WriteLine("Girilen Sayı Çift");

            else
                Console.WriteLine("Girilen Sayı Tek");
            Console.ReadKey();
        }
    }
}

```

4. Örnek:

Bir mağazada satın alınan ürünlerin toplam fiyatı 0 ile 50tl arası olduğunda toplam fiyata artı kargo 5tl ücreti eklensin. 50 ile 100tl arası olduğunda toplam fiyata artı kargo 2tl ücreti eklensin. 100tl ve üzeri olduğunda Toplam ürün fiyatı ve ücretsiz kargo çıktısını ekrana yazdırın programı yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace if_else
{
    class Program
    {
        static void Main(string[] args)
        {
            double fiyat;
            Console.Write("Ürün toplam fiyatı: ");
            fiyat = Convert.ToDouble(Console.ReadLine());

            if(fiyat<50)
            {
                fiyat = fiyat + 5;
                Console.WriteLine("Toplam ürün fiyatı: "+fiyat);
            }

            else if(fiyat>50 && fiyat<100)
            {
                fiyat = fiyat + 2;
                Console.WriteLine("Toplam ürün fiyatı: "+fiyat);
            }

            else if(fiyat>=100)
                Console.WriteLine("Ücretsiz kargo. Toplam ürün fiyatı: " + fiyat);

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

5. Örnek:

Klavyeden girilen 2 sayıyı seçilecek olan işleme göre toplama, çıkarma, çarpma, bölme yapan programı yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace if_else_toplama_cikarma
{
    class Program
    {
        static void Main(string[] args)
        {
```

```

float sayi1, sayi2, sonuc;
string secim;
Console.WriteLine("Birinci Sayıyı Giriniz: ");
sayi1 = Convert.ToSingle(Console.ReadLine());

Console.WriteLine("İkinci Sayıyı Giriniz: ");
sayi2 = Convert.ToSingle(Console.ReadLine());

Console.WriteLine("Lütfen yapmak istediğiniz işlemi seçiniz");
Console.WriteLine("Toplama işlemi için: +");
Console.WriteLine("Çıkarma işlemi için: -");
Console.WriteLine("Çarpma işlemi için: x");
Console.WriteLine("Bölme işlemi için: /");
secim = Console.ReadLine();

if (secim == "+")
{
    sonuc = sayi1 + sayi2;
    Console.WriteLine("Toplama Sonucu:" + sonuc);
}

else if (secim == "-")
{
    sonuc = sayi1 - sayi2;
    Console.WriteLine("Çıkarma Sonucu:" + sonuc);
}

else if (secim == "x")
{
    sonuc = sayi1 * sayi2;
    Console.WriteLine("Çarpma Sonucu:" + sonuc);
}

else if (secim == "/")
{
    sonuc = sayi1 / sayi2;
    Console.WriteLine("Bölme Sonucu:" + sonuc);
}

else
    Console.WriteLine("Seçim yapmadınız");
Console.ReadKey();
}
}
}

```

Program ekran çıktısı:

```
Birinci Sayınızı Giriniz: 20
İkinci Sayınızı Giriniz: 5
Lütfen yapmak istediğiniz işlemi seçiniz
Toplama işlemi için: +
Çıkarma işlemi için: -
Çarpma işlemi için: x
Bölme işlemi için: /
x
Çarpma Sonucu:100
```

switch case Kullanımı

Bazı durumlarda if else yapısını kullanmak programın çok güç ve karmaşık olmasına neden olabilir. Böyle bir programda if else yapısını kullanmaktadır basit, sade ve anlaşılır olması bakımından switch case yapısının kullanılması daha mantıklı olacaktır. Tabi burada switch case yapısının basit düzeyde if else kullanımının yerine geçmesi buda switch case yapısının gereksiz bir kullanıma sebep olacağını da unutmuyalım.

```
switch (Anahtar Değişken)
{
    case 1:
        Komut;
        break;
    case 2:
        Komut;
        break;
    default:
        Komut;
        break;
}
```

switch case yapısında anahtar değişkene gelen veri türüne bağlı olarak case'deki hangi ifadeye karşılık geliyorsa oradaki komutlar işlenir. Anahtar ifadeye gelen veri eğer case satırlarında yoksa default kısmı devreye girecektir. Ayrıca switch case yapısında, case'e ait komutların break satırı ile sonlandırılması gereklidir. Eğer break satırı ile sonlandırılmazsa program hata verecektir.

Switch case yapısında dikkat edilmesi gereken noktalar.

- Bir case ifadesin birden fazla aynı durumu olamaz.
- Case satırları break komutu ile sonlandırılmalıdır. Goto anahtar kullanılacaksa break komutunu kullanmaya gerek yoktur.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 2;
            switch (x)
            {
                case 1:
                    Console.Write("C#");
                    break;
                case 2:
                    Console.Write("Java");
                    break;
                case 3:
                    Console.Write("Php");
                    break;
                default:
                    Console.Write("Seçim yapılmadı");
                    break;
            }
            Console.ReadKey();
        }
    }
}

```

Yukarıdaki örnekte anahtar ifademiz x olarak tanımlanmıştır. x değişken türüne ön tanımlı 2 değeri atanmıştır. Program çalıştırıldığında case 2 komutu çalıştırılıp, ekranımıza Java yazdırılacaktır.

Bu örneğimizi klavyeden okuma işlemi yapacak şekilde dönüştürelim:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            x = Convert.ToInt32(Console.ReadLine());
            switch (x)
            {
                case 1:
                    Console.Write("C#");
                    break;
                case 2:
                    Console.Write("Java");
                    break;
            }
        }
    }
}

```

```
        case 3:  
            Console.Write("Php");  
            break;  
        default:  
            Console.Write("Seçim yapılmadı");  
            break;  
    }  
    Console.ReadKey();  
}  
}
```

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            x = Convert.ToInt32(Console.ReadLine());
            switch (x)
            {
                case 1:
                    string devam;
                    Console.Write("Sistem Yüklemesi Yapılsın mı? (e, h): ");
                    devam = Console.ReadLine();
                    Console.WriteLine((devam == "e") || (devam == "y") ? "Loading..." : "İptal Edildi.");
                    break;

                default:
                    Console.WriteLine("Seçim yapılmadı");
                    break;
            }
            Console.ReadKey();
        }
    }
}
```

Yukarıdaki örneğimizde klavyeden 1 girilirse ?: operatörüne bağlı komutlar çalışacaktır. Tek bir case komutunu kullandık. 1'in haricinde girilen sayılar default kısmında işlenecektir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
```

```

{
    static void Main(string[] args)
    {
        int x;
        x = Convert.ToInt32(Console.ReadLine());
        switch (x)
        {
            case 1:
            case 2:
                Console.Write("Java");
                break;

            case 3:
            case 4:
                Console.Write("Php");
                break;

            default:
                Console.Write("Seçim yapılmadı");
                break;
        }
        Console.ReadKey();
    }
}

```

Eğer farklı case sabitlerinin aynı komutları çalıştırmasını istiyorsak yukarıdaki gibi kullanılabilir. Klavyeden girilen değerin 1 veya 2 olması durumunda ekrana java yazdırılacaktır. Klavyeden girilen değerin 3 veya 4 olması durumunda ekrana php yazdırılacaktır.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            x=Convert.ToInt32(Console.ReadLine());
            switch (x)
            {
                case 1:
                    Console.Write("C#");
                    break;

                case 2:
                    Console.Write("Java");
                    break;

                case 3:
                    Console.Write("Php");
                    goto case 1;
            }
        }
}

```

```
        default:  
            Console.WriteLine("Seçim yapılmadı");  
            break;  
    }  
    Console.ReadKey();  
}  
}
```

Yukarıdaki örnekte goto anahtar kodu kullanılarak, klavyeden girilen değerin 3 olması durumunda ekrana php yazıp case 1 satırına gidecektir. Burada da case 1 satırı işlenip php'nin yanına C# yazdırılacaktır.

Örnek:

Klavyeden girilen 2 sayıyı ekrana yazdırılan işleme göre seçim yapıp toplama, çıkarma, çarpma, bölme sonucunu switch case yapısını kullanarak yapınız.

```
        Console.WriteLine("Çarpma Sonucu:" + sonuc);

        break;
    case "/":
        sonuc = sayi1 / sayi2;
        Console.WriteLine("Bölme Sonucu:" + sonuc);

        break;
    default:
        Console.WriteLine("Seçim Yapmadınız.");
        break;
    }
    Console.ReadKey();
}

}
```

Örnek:

Klavyeden girilen sayı ile haftanın kaçinci günü olduğunu bulan programı yapınız.

```
        break;
    default:
        Console.WriteLine("lütfen 1 ile 7 arasında sayı giriniz");
        break;
    }
    Console.ReadKey();
}
}
```

Değişken türümüzü byte olarak tanımladık. Çünkü 1 ile 7 arasında değer girileceği için int yada başka değişken türünü kullanmamız programımızın gereksiz hafıza alanı işgal etmesine neden olacaktır. Bu neden en düşük değer alabilen byte değişken türünü kullandık.

Örnek:

Gösterim yapılan işleme göre seçim yaptığımızda case satırında karşılık gelen ifadenin çalıştırılmasını sağlayan uygulamamızı yapalım.

```
        default:
            Console.WriteLine("lütfen 1 ile 5 arasında sayı giriniz");
            break;
    }
    Console.ReadKey();
}
}
```

Program ekran çıktısı:



Goto anahtar yapısını kullanarak örneğimizi aşağıdaki gibi değiştirelim.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace switch_case_kulanimi_goto
{
    class Program
    {
        static void Main(string[] args)
        {
            byte x;
            Console.WriteLine("Lütfen seçiminizi yapınız:\n*****\n");
            // \n bir alt satıra geç
            Console.WriteLine(" Mercedes\t(1)"); // \t ifadesi bir tab tuşu kadar boşluk bırakır
            Console.WriteLine(" Bmw\t\t(2)");
            Console.WriteLine(" Ford\t\t(3)");
            Console.WriteLine(" Audi\t\t(4)");
            Console.WriteLine(" Volkswagen\t(5)");
            Console.Write("\n*****\nSeçiminiz? : ");
            x = Convert.ToByte(Console.ReadLine());

            switch (x)
            {
```

```
        case 1:
            Console.WriteLine("Seçilen Araç Modeli: Mercedes");
            break;
        case 2:
            Console.WriteLine("Seçilen Araç Modeli: Bmw");
            break;
        case 3:
            Console.WriteLine("Seçilen Araç Modeli: Ford");
            goto case 4;

        case 4:
            Console.WriteLine("Seçilen Araç Modeli: Audi ");
            break;
        case 5:
            Console.WriteLine("Seçilen Araç Modeli: Volkswagen ");
            goto case 2;//



    default:
        Console.WriteLine("lütfen 1 ile 5 arasında sayı giriniz");
        break;
    }
    Console.ReadKey();
}

}
```

Klavyeden 5 girildiğine Seçilen araç modeli wolksvagen ve alt satırda seçilen araç modeli bmw yazacaktır. Klavyeden 3 girildiği zaman seçilen araç modeli ford ve alt satırda seçilen araç modeli audiyazacaktır.

Goto anahtar yapısını kullanarak herhangi bir yere dönüş yapılabılır. Sadece switch case yapısında kullanılmaz. Etiket tanımlanan her yerde kullanılabilir. Tabi goto yapısını her yerde kullanmak mantıksız olacaktır. Özellikle döngü gibi sürekli işlemi tekrar eden durumlarda kullanıldığı zaman program sürekli çalışacaktır. Hatalı bir durum olduğu zaman veya çıkış yapmak istediğimiz zaman buna müsaade etmeyecektir. Örneğin aşağıdaki gibi goto yapısını kullanarak örneğimizi sürekli döngü haline sokmuş oluruz.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace switch_case_kulanimi_goto
{
    class Program
    {
        static void Main(string[] args)
        {
            git://goto yapısı için etiket tanımlandı
            byte x;
            Console.WriteLine("Lütfen seçiminizi yapınız:\n\n*****\n");
            // \n bir alt satırda geç
            Console.WriteLine(" Mercedes\t(1)"); // \t ifadesi bir tab tuşu kadar boşluk
        }
    }
}
```

```

        Console.WriteLine(" Bmw\t\t(2)");
        Console.WriteLine(" Ford\t\t(3)");
        Console.WriteLine(" Audi\t\t(4)");
        Console.WriteLine(" Volkswagen\t(5)");
        Console.Write("\n*****\nSeçiminiz? : ");
        x = Convert.ToByte(Console.ReadLine());

        switch (x)
        {

            case 1:
                Console.WriteLine("Seçilen Araç Modeli: Mercedes");
                break;
            case 2:
                Console.WriteLine("Seçilen Araç Modeli: Bmw");
                break;
            case 3:
                Console.WriteLine("Seçilen Araç Modeli: Ford");
                break;
            case 4:
                Console.WriteLine("Seçilen Araç Modeli: Audi ");
                break;
            case 5:
                Console.WriteLine("Seçilen Araç Modeli: Volkswagen ");
                break;
            default:
                Console.WriteLine("lütfen 1 ile 5 arasında sayı giriniz");
                break;
        }
        goto git;// git etiketine dön

    }
}

```

C# Döngü Yapıları

for Döngüsü

Döngü yapıları belirli bir koşula bağlı olarak sürekli veya birden fazla sayıda kod satırlarımızın çalıştırılması istendiğinde aynı blok içerisinde tekrar etmesini istediğimizde for döngü yapısı kullanılır. Döngü bloğunun koşula bağlı şart yerine getirildiği sürece tekrar eder. Koşul sağlandığında döngüden çıkış olur.

```

for (Tanımlı değişken türü ve içeriği; Koşul; Koşula bağlı olarak neler yapılacağı)
{
    Komut veya komutlar;
}

```

Not: for bloğunun içerisinde tek bir komut kümeli kullanılacaksa { } ifadelerinin kullanılmasına gerek yoktur. If else, switch case karar kontrol mekanizmalarında da bu durum geçerlidir.

Örnek:

0'dan 10'a kadar sayıları alt alta gelecek şekilde ekrana yazdırın programı for döngüsü ile yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_dongusu
{
    class Program
    {
        static void Main(string[] args)
        {

            for(int y=0;y<=10;y++)
            {

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

Program çalıştırıldığında ekranda alt alta 0 ve 10 sayıları da dahil olmak üzere yazdıracaktır. Program kod satırlarının işleyişine bakalım:

```
for(int y=0;y<=10;y++)
```

Buradaki yapıda int y değişkeni tanımlayıp içerisinde 0 değeri atandı. Bu değer başlangıç değeri olarak da denilebilir. Başlangıçtan hangi değer olacağsa bu sayı ataması yapılır. Sayı ataması yapılmadan direkt değişken tanımı yapılp bırakılamaz. $y \leq 10$ ifadesinde; y değişkenimiz 10'dan küçük veya eşitse, $y++$ değişkenini çalıştırıp bir artırma yapacaktır. Bu artırma işlemi tâki 10'a eşit olana kadar döngü işlemi devam edecektir. y değeri 10'a eşit olursa döngü son kez işlenip sonlandırılacaktır.

Örnek:

1'den 4'e kadar olan sayıları toplayıp ekrana yazdırın programı for döngüsü ile yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_dongusu
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 0;
            for(int y=0;y<5;y++)
            {
                x += y;
                Console.WriteLine(x);
            }
        }
    }
}
```

```

        Console.ReadKey();
    }
}

```

Program çalıştırıldığında ekrana 0, 1, 3, 6, 10 alt alta gelecek şekilde yazdırılacaktır. y değeri 1 den 10 kadar artırma işlemi yaparken her yeni atanan x değeri ile toplama işlemi yapılır. Tabloda bunun daha net açıklamasını gösterelim.

$x + y$	y'değeri	Sonuç(x'in son hali)
0	0	0
$0+1 = 1$	1	1
$2+1 = 3$	2	3
$3+3 = 6$	3	6
$4+6 = 10$	4	10

Kod satırlarına inceleyelim:

```

x += y;

```

x'ın her zaman son atanan değeri koruyup bir sonraki işlemde atanan değer ile toplama işlemi yapacaktır. Bunu şöylede yazabilirdik:

```

x = x + y;

```

Aynı sonuç bu kullanımda da olacaktır.

Örnek:

1'den 5'e kadar olan sayıları çarpıp ekrana yazdırın programı yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_dongu
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 1;
            for(int y=1;y<=3;y++)
            {
                x *= y;
                Console.WriteLine(x);
            }
            Console.ReadKey();
        }
    }
}

```

Program çalıştırıldığında ekrana 1,2,6 sayıları yazılmış olacaktır. Burada dikkat edilmesi gerekn nokta ilk değerlerin 0 olmamasıdır. Çarpma işleminden ilk değerin 0 olması bütün

sonuçların 0 çıkmasına neden olacaktır. Yani bu örneğimizi aşağıdaki gibi değiştirdiğimizde ekrana 0, 0, 0 sayıları yazdırılmış olacaktır.

```
static void Main(string[] args)
{
    int x = 0;
    for(int y=1;y<=3;y++)
    {
        x *= y;
        Console.WriteLine(x);
    }
    Console.ReadKey();
}
```

Not: for içerisinde写的 koşul ifadelerinin tamamı kullanılabildiği gibi istenirse bazıları veya tamamen kullanılmayıp boşta bırakılabilir. Her halükarda ;'ler mutlaka iki tane kullanılmalıdır. For döngüsü içerisinde tanımlanan değişkenin sadece for bloğu içerisinde kullanıldığını da unutmayalım.

Örnek:

5'ten küçük sayı girilmesi durumunda ekrana 5'ten küçük sayı girildi yazdırılsın. Eğer 5'ten büyük sayı girilirse döngü tekrar başa alınıp 5'ten küçük sayı giriniz uyarısı verilsin örneğini yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_bool_kullanimi
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            bool y = true;
            for(;y;)
            {
                Console.WriteLine("Lütfen 5'ten küçük sayı giriniz:");
                x = Convert.ToInt32(Console.ReadLine());

                if(x>5)
                {

                    Console.WriteLine("5'ten büyük sayı girildi.");
                }
                else
                {
                    y = false;
                    Console.WriteLine("5'ten küçük sayı girildi.");
                }
            }
            Console.ReadKey();
        }
    }
}
```

```
file:///C:/Users/mehmet/Desktop/prg/for_bool_kullanimi/for_bool_kullanimi/bi...
Lütfen 5'ten küçük sayı giriniz:
8
Lütfen 5'ten küçük sayı giriniz:
4
5'ten küçük sayı girildi.
```

Program kod satırlarından for yapısına baktığımızda sadece bir koşulun olduğu tanımı yapılmıştır. Koşula bağlı olan değişkenümüz bool yapısı ile eğer şart sağlanmaz ise if yapısının içerisinde y değerine true ifadesi atanıp for döngüsü tekrar işlemi başa alacaktr. Else yapısının içerisinde klavyeden girilen ifade 5'ten küçükse ekrana 5'ten küçük sayı girildi ifadesi yazdırılıp döngüden çıkış olacaktr. 5'ten büyük sayıların girilmesi durumunda bu döngü işlemi sürekli tekrar edecektir. Taki girilen ifadenin 5'ten küçük olmasına kadar.

```
bool y = true;
for(;y;)
```

Burada y değişkenin içerişi ilk true olarak atanmıştır. For döngüsünde y değişkeni true olduğu için döngümüz çalışacaktır. Eğer y değişkenine ilk değer olarak false atanmış olsaydı for döngüsü çalışmadan program sonlandırılmış olacaktı.

Örnek:

Klavyeden girilen 2 sayıyı toplayan ve bunu sürekli olarak tekrar eden programımızı for döngüsünü kullanarak yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_kullanimi
{
    class Program
    {
        static void Main(string[] args)
        {
            int x,y, toplam;

            for(); //Sürekli döngü işlemi yap
            {

                Console.WriteLine("1. Sayıyı giriniz:");
                x = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("2. Sayıyı giriniz:");
                y = Convert.ToInt32(Console.ReadLine());
                toplam = x + y;
                Console.WriteLine("Toplam: " + toplam);
            }
        }
    }
}
```

```

        y = Convert.ToInt32(Console.ReadLine());
        toplam = x + y;
        Console.WriteLine("Toplama Sonucu:" + toplam);
        Console.WriteLine("*****");
    }

}
}
}

```

```

file:///C:/Users/mehmet/Desktop/prg/for_bool_kullanimi/for_bool_kullanimi/bi...
1. Sayınızı giriniz:5899
2. Sayınızı giriniz:258746
Toplama Sonucu:264645
*****
1. Sayınızı giriniz:589666
2. Sayınızı giriniz:9854755
Toplama Sonucu:10444421
*****
1. Sayınızı giriniz:

```

Programdan çıkış yapılmadığı sürece ekranda sürekli olarak sayı isteyip bu istenen iki sayıyı toplayıp ekranda toplam sonucunu gösterecektir. Sürekli döngüyü sağlayan for yapısı içerisinde herhangi bir koşul sağlanmadığı için döngü bloğu sürekli işlemi bitirip başa alacaktır.

Örnek:

Klavyeden girilen bir sayının faktoriyelini bulan programı for döngüsünü kullanarak yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_faktoriyel
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            Console.Write("Faktoriyeli alınacak sayıyı giriniz:");
            x = Convert.ToInt32(Console.ReadLine());
            for(int i=1;x>1;x--)
            {
                i *= x;
                Console.WriteLine(i);

            }
            Console.ReadKey();
        }
    }
}

```

```
    }  
}
```

Program kod satırlarımızı incelemeye alalım:

```
for(int i=1;x>1;x--)  
{  
    i *= x;
```

Burada klavyeden girilen x değişkenin sayısı birer birer azaltılp i değişkeniyle çarpılı sona tekrar i değişkenin içeresine ataması yapılarak en son x'in değeri 1'e geldiğinde son sonuç ile çarpıp i değişkenin içeresine atama yaparak sayının faktöriyel sonucu bulunmuş olur.

Bir sayının faktöriyel hesabı yapılırken sayının geri tüm basamakları 1'e kadar olan kısımları bir biri ile çarpılır ve sonuç bulunmuş olur. Aynı işlem program yazarken de geçerli. For döngüsü ile faktöriyel hesabı yaptığımızda yapılan işlemi tablo ile açıklayalım. Örneğin klavyeden girilen sayıımız 4 olsun:

x(4) işlem x--	i	Sonuç (i)
4	1 * 4 = 4	4
3	4 * 3 = 12	12
2	12 * 2 = 24	24

$4!=24$ sonucunu program işleyişinde nasıl olduğunu tablomuzda göstermiş olduk. Program döngü içerisinde console.WriteLine kod satırımız kaldığı için işlem adı sonuçlarını tek tek göstermiş oluyor. Dilerseniz girilen sayıyı direk olarak sonucunu vermek istedğimizde kodlarımızı aşağıdaki gibi düzenleyebiliriz.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace for_faktoriyel  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int x, sonuc = 0;  
            Console.Write("Faktöriyeli alınacak sayıyı giriniz:");  
            x = Convert.ToInt32(Console.ReadLine());  
            for(int i=1;x>1;x--)  
            {  
                i *= x;  
  
                sonuc = i;  
            }  
            Console.WriteLine(sonuc);  
            Console.ReadKey();  
        }  
    }  
}
```

Girilen sayının direk olarak faktöriyeli göstermek için `console.WriteLine` ifadesini `for` döngümüzün dışına aldık. Ve ayrıca `int` sonuc diye bir değişken tanımladık. Bu değişkenin içerisinde ayrıca ilk değer olarak 0 değerini atadık. Bu değeri atamamızın sebebi sonuç değişkeni döngü haricinde herhangi bir yerde içerisinde atama yapılmadan `console.WriteLine` ile ekrana bastırılmıştır.

Örnek:

Klavyeden girilen harfe göre ekrana girilen harften sonraki harfleri alfabetik sıralamasını gösteren programımızı yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_yapisi
{
    class Program
    {
        static void Main(string[] args)
        {
            char harf;

            for (harf= Convert.ToChar(Console.ReadLine()); harf <= 'z'; harf++)
            {
                Console.Write(harf);
            }
            Console.ReadKey();
        }
    }
}
```

Programımızda `for` içerisinde okuma işlemi yapıp `char` değişken türü ile tanımlanmış olan harf değişkenine ataması yapılmıştır. `harf++` ile harfleri sırasıyla alfabetik sıraya göre getirme işlemi yapılp bu durum z harfine eşit olana kadar devam edilmiştir.

Örnek:

0'dan başlayıp klavyeden girilen sayıya kadar olan sayıların toplamını veren programı `for` döngüsü ile yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_yapisi
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayı,topla=0;
            sayı = int.Parse(Console.ReadLine());
            //Klavyeden okuma işlemi yapmak. Convert.ToInt32 gibi bir başka kullanımı
        }
    }
}
```

```

        for (int i = 0; i <= sayı; i++)
        {
            topla += i;
        }
        Console.WriteLine("Sayıların toplamı {0}", topla);
        Console.ReadKey();
    }
}
}

```

Örneğimizde klavyeden girilen sayı 5 olması durumunda 0'dan 5'e kadar olan sayıları toplayıp($0+1+2+3+4+5 = 15$) sonucunu ekranımıza yazdıracaktır. Örneğimizi biraz daha ayrıntılı olacak şekilde düzenleyelim. Çıkış işlemi yapana kadar sürekli klavyeden sayı isteyip buna göre toplama işlemi yapan programımızı aşağıdaki şekilde düzenleyebiliriz.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_dongusu_toplam
{
    class Program
    {
        static void Main(string[] args)
        {
            int toplam = 0;
            bool durum = true;
            char e;
            for (;durum;)
            {
                Console.Write("Lütfen bir sayı giriniz:");
                for(int x=Convert.ToInt32(Console.ReadLine());x>0;x--)
                {
                    toplam += x;
                }
                Console.WriteLine("Toplam Sonuç: " + toplam);
                Console.WriteLine("Çıkış Yapılsın mı?: Evet(e) Hayır(h)");
                e = Convert.ToChar(Console.ReadLine());
                if (e == 'e' || e == 'E')
                    durum = false;
                else if (e == 'h' || e == 'H')
                    continue;
                }
            }
        }
    }
}

```

Program ekran çıktısı aşağıdaki gibidir:

```
file:///C:/Users/mehmet/Desktop/prg/for_dongusu_toplam-20170327T160504Z... - □ ×
Lütfen bir sayı giriniz:10
Toplam Sonuç: 55
Çıkış Yapılsın mı?: Evet<e> Hayır<h>
h
Lütfen bir sayı giriniz:20
Toplam Sonuç: 265
Çıkış Yapılsın mı?: Evet<e> Hayır<h>
h
Lütfen bir sayı giriniz:70
Toplam Sonuç: 2750
Çıkış Yapılsın mı?: Evet<e> Hayır<h>
h
Lütfen bir sayı giriniz:
```

Örnek:

Klavyeden girilen sayıya kadar olan çift sayıların ve tek sayıların çarpımını veren programımızı for döngüsünü kullanarak yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_yapisi
{
    class Program
    {
        static void Main(string[] args)
        {
            int c = 1, t=1 , z;
            Console.WriteLine("\nSayı giriniz:");

            z = int.Parse(Console.ReadLine());

            for (int y = 1; y <= z; y++)
            {
                if (y % 2 == 1)//tek sayılar
                    c *= y;

                if (y % 2 == 0)// Çift Sayılar
                    t *= y;
            }
            Console.WriteLine("\nGirilen Sayıya kadar olan çift sayıların çarpımı=" +c);
            Console.WriteLine("\n*****\n");
            Console.WriteLine("Girilen Sayıya kadar olan çift sayıların çarpımı=" + t);

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

Örnek:

1'den 50'ye kadar olan tek ve çift sayılarını ekranaya yazdırın programı for döngüsünü kullanarak yapalım.

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace for_dongusu
{
    class Program
    {
        static void Main(string[] args)
        {
            int a=0;
            Console.WriteLine("Tek Sayılar :\t");
            for (int x=0;x<=50;x++)
            {

                if(a<50)
                {
                    for(int y=1;y<=50;y+=2)
                    {
                        Console.Write(" " + y);
                        a+=2;
                    }
                    Console.WriteLine("\nÇift Sayılar:\t");
                }

                if (x%2==0)
                    Console.Write(" " + x);
            }
            Console.ReadKey();
        }
    }
}

```

Programımızda iç içe for döngüsünü kullandık. Çalışma yapısı olarak ilk for döngüsü çalışlığında hemen altında bir kontrol işlemi yani a'nın 50'den küçük olup olmadığını kontrol ettik. a'nın ilk değeri 0 olarak atandı. Bu kontrol yapısında şart sağlandığı için hemen if yapısına bağlı olarak çalışan for yapısı devreyegirecektir. Bu for yapısında tek sayıları ekrana yazdırıldı. Döngülerin dışında ekrana yazmış olduğumuz tek sayılar açıklamasının yanında tek sayılar yazdırılacaktır. Böyle yapmamızın sebebi birden fazla aynı açıklama kısmının yazılmasının önüne geçmek. İçtek for yapısı tamamlandıktan sonra hemen alt satırındaki ekrana çift sayılar yazısı yazdırıldı. Tek sayılar işlemi yazdırılırken aynı zamanda a'nın değeride paralel olarak arttırıldı. Döngüden çıktılığında a'nın değeri 50'den büyük olacağı için bir sonraki dış döngü işleminde içteki for yapısı çalıştırılmayacaktır. Sonuçta artık dış döngü yapısı sürekli çalıştığı için mod yardımıyla alt satır çift sayıları yazdıracaktır. Aşağıda ekran çıktısı verilmiştir.

Örnek:

50'ye kadar olan tek sayıları ekrana yazdırın programımızı for döngüsüyle continue komutunu kullanarak yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace karar_kontrol_dongu
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int x = 0; x <= 50; x++)
            {
                if (x % 2 == 0)
                    continue;
                Console.WriteLine(x);
            }
            Console.ReadKey();
        }
    }
}
```

Programımız çalıştırıldığında 0'dan 50'ye kadar(50 dahil) olan tek sayıları continue komutu ile yazdıracaktır. Eğer continue komutunu kullanmamış olsaydık, ekranımıza 0'dan 50'ye kadar(50 dahil) çift sayıları çıkarmış olacaktı.

Continue komutu:

Bu komuttan sonra blok içerisinde kalan ifadeler çalıştırılmaz.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace karar_kontrol_dongu
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int x = 0; x <= 10; x++)
            {
                if (x < 5)
                {
                    continue;//bundan sonraki komutlara bakılmaz
                }
                Console.Write(x); // continue komutundan dolayı bu komut çalışmaz
                Console.Write(x);
            }
            Console.ReadKey();
        }
    }
}
```

```
        }  
    }  
}
```

Programımız çalıştırıldığında 5'ten küçük ifadeler if koşul yapısında sağlandığı için continue komutu ile 0..5 aralığındaki sayıları göstermeyecektir. Sayı 5'ten büyük olduğu zaman if koşulu sağlanmadığı için 5 ile 10 aralığındaki sayılar ekrana yazdırılacaktır.

while Döngüsü

Belirlenen koşul sağlandığı sürece while bloğu içerisinde kalan bloğun içerisinde çalışmasını sağlar. For bloğu ile tek farkı tek koşula bağlı olarak çalışmasıdır. for döngüsünde kullanılan örneğin bu kontrol for(;x>10;) while döngüsünde while(x>10) ile aynı yapıya sahiptir. While kullanım bloğu:

```
while (koşul)  
{  
    Komutlar;  
}
```

Kullanım bloğuna baktığımızda koşul sağlandığı sürece while komutu ile döngü sağlanmış olur. Burada döngüyü sonlandırmabilmek için koşula bağlı komut kümesinin sonlandırma işlemini yapması gereklidir.

Örnek:

0'dan 6'ya kadar olan sayıları ekrana yazdırın programımızı while döngüsünü kullanarak yapalım.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace while_dongusu  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int x = 0;  
            while (x < 6)  
            {  
                Console.WriteLine("x:{0}", x);  
                x++;  
            }  
            Console.ReadKey();  
        }  
    }  
}
```

Programımız çalıştığında ekrana alt alta gelecek şekilde 0'dan 6'ya kadar sayıları ekrana yazdıracaktır. Bu örneğimizi aşağıdaki gibide kullanabiliriz:

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace while_dongusu
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 0;

            while (x++ < 6)
            {
                Console.WriteLine("x:{0}", x);
            }
            Console.ReadKey();
        }
    }
}

```

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace while_dongusu
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 0, y;

            while (true) // for yapısında bu kullanım for(;;) şeklinde idi.
            /* sürekli döngüyü başlat. Eğer while(false) yazılırsa
            Döngüye girmeden sonlandırma işlemi yapılacaktır. */
            {
                y = ++x;

                if (y > 5)
                {
                    Console.WriteLine("Yüklandı.");
                    break;
                /* koşul burada sağlanlığında break komutu
                   kullanılarak while sürekli döngüsünden
                   Çıkılmış oldu. */
                }

                Console.WriteLine("Yükleniyor...");
            }
            Console.ReadKey();
        }
    }
}

```

Program çalıştırıldığında ekrana 5 kez Yükleniyor... yazısı çıķıp en son if bloğu arttırlan y değişkenin koşul içeriğini sağladığı için Yükleni yazısı çıķıp program sonlandırılacaktır.

Break komutu:

Break komutunu switch case kullanımında olduğu gibi program sonlandırılması olacaksa her yerde kullanılabilir. Burada da break komutunu kullanarak sonlandırma işlemi yapmış olduk. Eğer break komunu kullanmasaydık, program sonlandırma işlemi yapmayıp sürekli yükleniyor ve yükleni yazısın ekrana yazdıracaktı.

Örnek:

Daha önce for yapısında kullanmış olduğumuz örneği while ile yapalım. 5'ten küçük sayı girilmesi durumunda ekrana 5'ten küçük sayı girildi yazdırılsın. Eğer 5'ten büyük sayı girilirse döngü tekrar başa alınıp 5'ten küçük sayı giriniz uyarısı verilsin örneğini while döngüsünü kullanarak yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace while_dongusu
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            bool y = true;
            while(y) // for yapısındaki örnekte burası for(;y;) şeklindeydi.
            {
                Console.WriteLine("Lütfen 5'ten küçük sayı giriniz:");
                x = Convert.ToInt32(Console.ReadLine());

                if (x > 5)
                {

                    y = true;

                }
                else
                {
                    y = false;
                    Console.WriteLine("5'ten küçük sayı girildi.");
                }
            }
            Console.ReadKey();
        }
    }
}
```

Örnek:

Klavyeden girilen sayıya kadar olan çift sayıları gösteren programı while döngüsü ile yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace while_dongusu
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            Console.Write("Lütfen bir sayı giriniz:");
            x = Convert.ToInt32(Console.ReadLine());
            while (x>0)
            {
                if(x%2==0)//bir sayının 2 ile bölümünden kalan 0 ise o sayı çifttir
                Console.WriteLine(x+" ");

                --x;
            }
            Console.ReadKey();
        }
    }
}

```

do while Kullanımı

While döngüsünde koşul sağlanmadığı sürece komutlar çalışmayacaktır. do while döngüsünde ise koşul sağlanın veya sağlanmasın komutlar bir kere çalıştırılacaktır. Koşul sağlanırsa döngüye devam edilir. Eğer koşul sağlanmaz ise 1 kez çalıştırılır ve sonlandırma yapar. Komut yapısının kullanımı:

```

do
{
    komutlar;
}
while (koşul);

```

while(koşul) yapısından sonra ; bırakılmaz. Ama do while yapısında ; bırakılır.

Örnek:

Ekranımıza Programlama Teknikleri yazısını 5 kere yazdırın programı do while yapısını kullanarak yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace while_dongusu
{
    class Program
    {
        static void Main(string[] args)

```

```

    {
        int x = 0;

        do
        {
            Console.WriteLine("Programlama Teknikleri");
            x++;
        }
        while (x<=5);

        Console.ReadKey();
    }
}

```

Programımızı çalıştırıldığımızda ekranımıza 5 kere Programlama Teknikleri yazısını yazacaktır. Eğer kodumuzda ki while yapısını şu şekilde değiştirdiğiz:

```

while (x>5);

```

koşul sağlanmadığı halde bile Programlama Teknikleri yazısını ekrana 1 kere yazacaktır.

Karar Kontrol ve Döngülerle ilgili Uygulama Örnekleri

Örnek:

Klavyeden girilen bir sayının asal olup olmadığını bulan programı yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace karar_kontrol_dongu
{
    class Program
    {
        static void Main(string[] args)
        {
            bool x=false;
            int sayi = Convert.ToInt32(Console.ReadLine());

            for (int i = 1; i < sayi; i++)
            {
                if (sayi % i == 0 && i != 1)
                    x = true;

            }
            if(x==true)
                Console.WriteLine("Girilen sayı asal değildir.");
            else
                Console.WriteLine("Girilen sayı asaldır.");
            Console.ReadKey();
        }
}

```

```
    }  
}
```

Bir sayının asal olup olmadığı kendisinden ve 1'den başka böleni yoksa o sayı asal sayıdır. Örneğimizde sayının asal olup olmadığı kontrolünü sağlayan satırı bakalım:

```
if (sayi % i == 0 && i != 1)  
    x = true;
```

Klavyeden girilen sayı ile for döngüsünde o sayıya kadar üretilecek sayıların modu alınarak kalan ifade 0 ise ve for döngüsünde üretlen sayının 1'e eşit olmama durumunda girilen sayının asal olmadığı anlaşılacaktır. Daha önce tanımlanmış olan bool değişken türüyle bu ifadeye true değişkeni atanarak sayının asal olmadığını ekrana yazdırma işlemi yapılacaktır. Eğer koşulları sağlayan durum yoksa sayının asal olduğu bulunup ekrana girilen sayınız asaldır yazısı yazdırılacaktır.

Örnek:

Kullanıcı adı ve şifre kontrolü sağlayan programımızı, döngü ve karar kontrol yapılarını kullanarak yapalım. Programımızda en fazla 3 kere yanlış girmeye hakkı olsun. 3 yanlış girişten sonra hesabınız bloke edildi denip sonlandırma işlemi yapalım.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace kontrol_uygulama_ornegi  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            bool y = true;  
            int kalan = 0, a = 3;  
            string sifre=null, kuladi=null;  
  
            for (int x = 1; x < 4; x++)  
            {  
                Console.Write("Giriş yapmak için kullanıcı adını giriniz:");  
                kuladi = Console.ReadLine();  
                Console.Write("Şifrenizi giriniz:");  
                sifre = Console.ReadLine();  
  
                if (sifre == "123abc" && kuladi=="mehmet" && kalan < 4)  
                {  
                    Console.Clear();  
                    Console.WriteLine("Giriş Yapıldı");  
  
                }  
                else  
                {  
                    a--;  
                    kalan++;  
                    Console.WriteLine("Kalan şifre kullanım hakkınız:{0}", a);  
                    if (a == 0)  
                    {  
                        Console.Clear();  
                    }  
                }  
            }  
        }  
    }  
}
```

```

        Console.WriteLine("Hesabınız Bloke edildi");
        break;
    }

}

Console.ReadKey();
}

}

```

Örnek:

Bir bankanın atm'sine kullanıcı adı ve şifre girilerek giriş yapılsın. 3 kere yanlış girilmesinde hesabınız bloke edildi uyarısı verilsin. Eğer kullanıcı giriş'i doğru 4 tane banka seçimi için seçenek sunulsun. Seçeneklerden bir tanesinin seçimini yaptıktan sonra havale yapılacak hesap numarası istenilsin. Hesap numarası 10 ile sınırlı yapılarak giriş yapılsın. Hesap numarası doğru girdikten sonra havale tutarı 10TL altında olmayacağı şekilde kontrol sağlanıp eğer 10TL yukarı yapılmışsa Belirtilen hesaba havale işlemi tamamlanmıştır yazısı ekrana yazdırılsın.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace kontrol_uygulama_ornegi
{
    class Program
    {
        static void Main(string[] args)
        {
            bool y = true, surekli=true, banka=true;
            int kalan = 0, a = 3;
            string sifre=null, kuladi=null;//ilk değerler boş olarak atandı.

            for (int x = 1; x < 4; x++)
            {
                Console.Write("Giriş yapmak için kullanıcı adını giriniz:");
                kuladi = Console.ReadLine();
                Console.Write("Şifrenizi giriniz:");
                sifre = Console.ReadLine();

                surekli = true;
                y = true;
                banka = true;

                if (sifre == "123abc" && kuladi=="mehmet" && kalan < 4)
                {
                    Console.Clear();//console ekranı temizle
                    Console.WriteLine("Giriş Yapıldı");

                    while (surekli)
                    {

```

```

        for (;y;)
        {
            Console.WriteLine("Banka tercihinizi yapınız");
            Console.WriteLine("Akbank\t\t[1]");
            Console.WriteLine("Vakıfbank\t[2]");
            Console.WriteLine("Halkbankası\t[3]");
            Console.WriteLine("Ziraat Bankası\t[4]");

            int z = Convert.ToInt32(Console.ReadLine());

            if (z < 5 && z > 0)
            {
                banka = true;

                for (; banka;)
                {
                    Console.Write("Havale yapılacak hesap numarasını giriniz:");
                    string hesap = Console.ReadLine();

                    if (hesap.Length > 9 && hesap.Length < 11)// hesap uzunluğunun kontrolü
                    {
                        Console.WriteLine("Havale Tutarı:");
                        int havale = Convert.ToInt32(Console.ReadLine());

                        if (havale < 10)
                            Console.WriteLine("Havale tutarı 10tl'den düşük olamaz");

                        else
                    {
Console.WriteLine("Havale işlemi yapılmıştır. Tutar:{0} \nHesap No:{1}", havale, hesap);
                    Console.WriteLine("Çıkış yapılışın mı? Evet (e) Hayır (h)");
                        string cikis = Console.ReadLine();

                        if (cikis == "e" || cikis == "E")
                        {
                            surekli = false;
                            y = false;
                            banka = false;

                            break;
                        }

                        else if (cikis == "h" || cikis == "H")
                            banka = false;
                    }

                    else
                        Console.WriteLine("Hesap numarası yanlış");
                }

                else
                    Console.WriteLine("Lütfen 1 ile 4 arası seçiminizi giriniz");
            }
        }

        else
    {

```

```

        a--;
        kalan++;
        Console.WriteLine("Kalan şifre kullanım hakkınız:{0}", a);

        if (a == 0)
        {
            Console.Clear();
            Console.WriteLine("Hesabınız Bloke edildi");
            break;
        }

    }

Console.ReadKey();
}
}

```

Diziler

Bu araya kadar işledğimiz konularda değişken tanımlı yaparken değişken adını x,y...sayı.. v.s gibi tanımlamalar ile yaptık. Tabi bu tanımlamalar az sayıda oldu. Eğer 50, 200 yada daha fazla değişken tanımlı tek tek isim vererek tanımladığımızda çok zahmetli bir hal almış olacaktır. İşte bu gibi durumların önüne geçmek için dizi ile değişken türü ve ismi ne olacağı bir kere tanımlı yapıp zahmet çekmeden çok hızlı değişken isimleri istenildiği kadar verilebilir. Dizi ile değişken türü ve değişken adı nasıl kullanıldığına bakalım.

```
double [] x = new double[120];
```

double değişken türünden 120 tane değişken tanımladık. Bu kadar çok değişken türünü daha önce yaptığımz gibi kullanmak çok güç olacaktır.

```

double [] x = new double[120];
int[] y = new int[50];
float[] z = new float[48];
string[] t = new string[350];

```

Dizi elemanlarını kullanmak istedğimizde;

```
x[5] = 30;
```

Dizi elemanı olarak tanımlanan x'in 6. Elemanına 30 değeri atandı. Dizi elemanlarının ilk başlangıç noktası 0'dır. Yani eğer x'in 1. Elemanına 40 değeri atamak isteniyorsa o zaman kod satırımızı şu şekilde düzeltmek gerekir:

```
x[0] = 40;
```

Örnek:

Float türünden tanımlanmış dizinin, 17. elemanına 30.58 ondalıklı sayısı atanıp ekrana yazdırın programı yapalım.

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi
{
    class Program
    {
        static void Main(string[] args)
        {
            float[] x = new float[35];
            x[16] = 30.58f;
            Console.WriteLine(x[16]);
        }
    }
}
```

Örnek:

Bu örneğimizi klavyeden okunan değer ile dizinin 17. elemanına değer atamasını yapmak istedğimizde aşağıdaki şekilde kullanabiliriz:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi
{
    class Program
    {
        static void Main(string[] args)
        {
            float[] x = new float[35];
            x[16] = Convert.ToSingle(Console.ReadLine());/*klavyeden okunan değer 17.
dizi değişkenine atanmış oldu.*/
            Console.WriteLine(x[16]);
        }
    }
}
```

Örnek:

Dizimizin elemanını kendimiz klavyeden girilen kadar olmasını istersek aşağıdaki şekilde kullanabiliriz.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi
{
    class Program
```

```

{
    static void Main(string[] args)
    {
        int y = Convert.ToInt32(Console.ReadLine());
        float[] x = new float[y];
        x[12] = Convert.ToSingle(Console.ReadLine());/*klavyeden okunan değer 12.
dizi değişkenine atanmış oldu.*/
        Console.WriteLine(x[16]);
    }
}

```

Bu örneğimizde eğer dizi sayısı klavyeden 10 olarak girilmişse x[12] dizi değerine atama yapamayacaktır. Olmayan bir dizi sayısına atama yapılamadığı için hata verecektir.

Not: Dizi tanımlamalarında önceden veriler atanıp daha sonra kaçınıcı dizinin verisine erişilecekse dizi elemanın numarası verilerek kullanılabilir. Örneğin:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi_ornekleri
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] sayilar = { 10, 52, 36, 45, 1258, 23698 };
            string[] aylar = { "ocak", "şubat", "mart" };

            Console.WriteLine("Sayilar dizisinin 3. elemanı:" + sayilar[2]);
            Console.WriteLine("Aylar dizisinin 2. elemanı:" + aylar[1]);
            Console.ReadKey();
        }
    }
}

```

Yukarıdaki örnekte sayılar ve aylar dizileri tanımlanıp içerisindeki değerler atanmıştır. Program çalıştırıldığında sayılar dizisinin 3. Elemanı olarak 36, aylar dizisinin 2. Elemanı olarak şubat ayı gösterilecektir.

Örnek:

Bir dersin 2 sınav sonucun klavyeden girilen not yardımı ile ortalamasını alıp eğer ortalama sonucu 50 ve üzeri ise geçti yazıp not ortalaması sonucunu, değilse kaldı yazıp not ortalaması yazdırın programı dizi kullanara yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace dizi_ornekleri
{
    class Program
    {
        static void Main(string[] args)
        {
float[] not= new float[3];// float türünden dizi kullanarak 3 değişken tanımlı yapıldı
            Console.WriteLine("1. Notunuzu giriniz:");
            not[0] = Convert.ToSingle(Console.ReadLine());

            Console.WriteLine("2. Notunuzu giriniz:");
            not[1] = Convert.ToSingle(Console.ReadLine());

            not[2] = (not[0] + not[1]) / 2;

            if(not[2]>=50)
                Console.WriteLine("Geçtiniz. Not ortalamanız:" + not[2]);

            else
                Console.WriteLine("Kaldınız. Not ortalamanız:" + not[2]);
            Console.ReadKey();
        }
    }
}

```

foreach Kullanımı

Bir dizinin elemanlarının hepsini kullanmak istediğiniz zaman foreach yapısı kullanılır. Dizi elemanlarının değerini değiştirmez. Sadece ekrana yazdırma veya okuma işlemi için kullanılır.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi_uygulama_orn
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] isimler = { "Mehmet", "Ahmed", "Fatima", "Hatice", "Ömer" };
            foreach(string x in isimler)
            {
                Console.WriteLine(x);
            }
            Console.ReadKey();
        }
    }
}

```

Programımız çalıştırıldığında isimler dizindeki elemanları alt alta gelecek şekilde yazdıracaktır. Foreach yapısında yeni bir string x değişkeni tanımlayıp isimler dizisindeki elemanların x'in içerisine atamasını yapacaktır(x=isimler). Dizi değişkenimiz hangi türden tanımlama

yapılmışsa foreach yapısının içerisinde tanımlanacak olan değişken türüde dizide tanımlanınan tür ile aynı olması gereklidir. Örneğin int tanımlı bir dizi türünün elemanlarını ekrana yazdırduğumuzda foreach içeresine yazılacak değişken türüne int olması gereklidir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi_ugulama_orn
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] sayılar = {547, 258, 3, 68, 59};
            foreach(int a in sayılar)
            {
                Console.WriteLine(a);
            }
            Console.ReadKey();
        }
    }
}
```

Dizi tanımlamamızda kullanılan int sayılar dizisinin elemanlarını foreach yapısını kullanarak alt alta gelecek şekilde yazdırma işlemini yapmıştır.

Çok Boyutlu Diziler

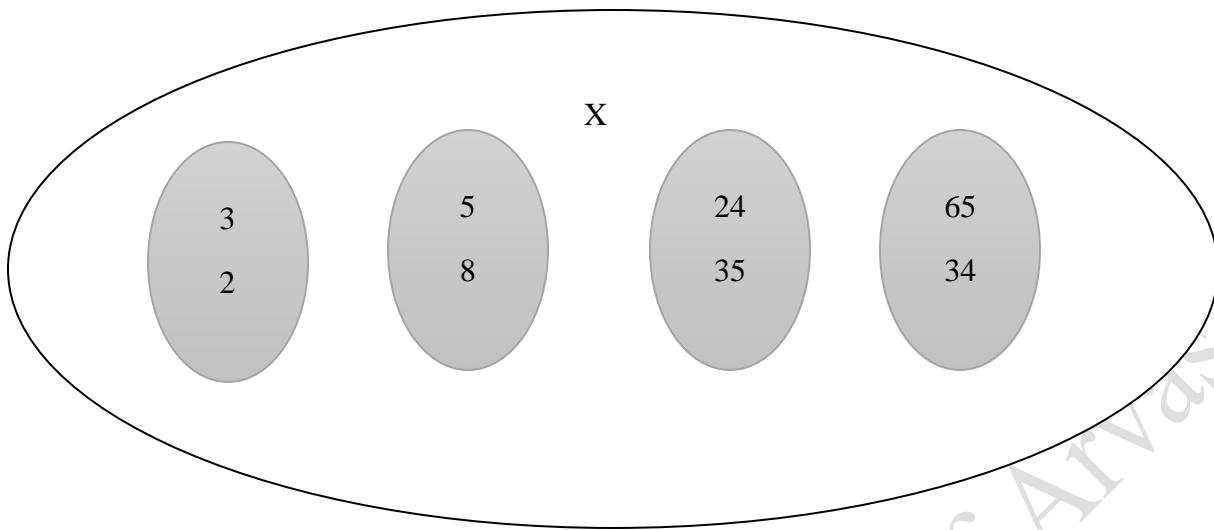
Şimdiye kadar oluşturduğumuz dizi elemanları tek satır tek sütun boyutunda idi. Çok boyutlu dizi elemanlarında bir dizinin matris şeklinde kaç tane eleman içereceğini belirtebiliriz. Örneğin 4x2'lik int türünden 2 boyutlu matris dizi elemanını gösterdiğimizde aşağıdaki gibi tanımlanır:

```
int[,] x = new int[4, 2];
```

4 küme elemanlı her bir kümeyi 2 elemanı olan x tamsayı kümesi olarak tanımlanabilir.

```
int[,] y = { { 3, 2 }, { 5, 8 }, { 24, 35 }, { 65, 34 } }; //4x2 elemanlı dizinin öne tanımlı elemanları.
```

Küme üzerinden gösterecek olursak;



Örnek:

İki boyutlu bir matris dizinin elemanlarını ekrana yazdırın programımızı yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi_matris
{
    class Program
    {
        static void Main(string[] args)
        {

            int[,] y = { { 3, 2 }, { 5, 8 }, { 24, 35 } };

            Console.WriteLine(y[0, 0] + "\n" + y[0, 1] + "\n" + y[1, 0] + "\n" + y[1, 1] + "\n" + y[2, 0] + "\n" + y[2, 1]);
            Console.ReadKey();
        }
    }
}
```

İkiden fazla dizi boyutu tanımlamasına örnek:

```
int[, , ] x = new int[4, 3, 2];
```

Yukardaki tanımlamada 3 boyutlu bir dizi tanımı yapılmıştır.

```
int[, , ] y = new int[1, 2, 4] { { { 2, 8, 9, 3 }, { 1, 1, 22, 125 } } };
```

Yukarıdaki tanımlamada 3 boyutlu bir matris dizi elemanın içeriği önceden ataması yapılan örneğimizde 1 kapsayıcı küme elemanı, aynı kümeye ait 2 eleman, her kümeye ait 4 eleman içermektedir. Tablo üzerinden 3 boyutlu dizi elemanlarına nasıl erişebileceğimizi görelim:

Dizi elemanı	İçeriği
y[0, 0, 0]	2
y[0, 0, 1]	8

y[0, 0, 2]	9
y[0, 0, 3]	3
y[0, 1, 0]	1
y[0, 1, 1]	1
y[0, 1, 2]	22
y[0, 1, 3]	125

```
int[,] y = new int[2, 2, 3] {{{66, 0, 3}, {785, 42, 6}}, {{2, 8, 9}, {1, 1, 22}}};
```

Yukarıdaki tanımlamada 3 boyutlu bir dizi matris elemanında 2 farklı küme, bu iki farklı kümenin 2 altkümesi, bu alt kümelerin eleman sayısı 3 olarak önceden tanımlama yapılmıştır. Tablo üzerinden 3 boyutlu dizi elemanlarına nasıl erişebileceğimizi görelim:

Dizi elemanı	İçeriği
y[0, 0, 0]	66
y[0, 0, 1]	0
y[0, 0, 2]	3
y[0, 1, 0]	785
y[0, 1, 1]	42
y[0, 1, 2]	6
y[1, 0, 0]	2
y[1, 0, 1]	8
y[1, 0, 2]	9
y[1, 1, 0]	1
y[1, 1, 1]	1
y[1, 1, 2]	22

Program içerisinde kullanımını gösterelim:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi_matris
{
    class Program
    {
        static void Main(string[] args)
        {

int[,] y = new int[2, 2, 3] {{{66, 0, 3}, {785, 42, 6}}, {{2, 8, 9}, {1, 1, 22}}};

            Console.WriteLine(y[0, 0, 0]);
            Console.WriteLine(y[0, 0, 1]);
            Console.WriteLine(y[0, 0, 2]);

            Console.WriteLine(y[0, 1, 0]);
            Console.WriteLine(y[0, 1, 1]);
            Console.WriteLine(y[0, 1, 2]);

            Console.WriteLine(y[1, 0, 0]);
            Console.WriteLine(y[1, 0, 1]);
```

```

        Console.WriteLine(y[1, 0, 2]);
        Console.WriteLine(y[1, 0, 0]);
        Console.WriteLine(y[1, 0, 1]);
        Console.WriteLine(y[1, 0, 2]);
        Console.WriteLine(y[1, 1, 0]);
        Console.WriteLine(y[1, 1, 1]);
        Console.WriteLine(y[1, 1, 2]);
    }

    Console.ReadKey();
}
}
}

```

Program çalıştırıldığında ekrana alt alta gelecek şekilde; 2,2,3,66,0,3,785,42,6,2,8,9,1,1,22 yazacaktır.

Yukarıda işlediğimiz matris dizileri düzenli bir yapıya sahip olduğu için aynı zamanda düzenli dizi matrisleride denir. Dizi eleman sayıları bazen düzenli olmayabilir her eleman sayısı farklı sayıda olacaksa düzensiz dizi elemanları kullanılmalıdır. Düzensiz dizi elemanlarını aşağıdaki gibi gösterebiliriz.

3 boyutlu 4 sütuna sahip düzensiz dizi örneğimiz:

```

float[][][] x = new float[4][][];
x[0] = new float[4][]; // 1. Satırda 4 sütun
x[1] = new float[1][]; // 2. Satırda 1 sütun
x[2] = new float[3][]; // 3. Satırda 3 sütun
x[3] = new float[3][]; // 3. Satırda 3 sütun

```

x[0, 0, 0];	x[0, 0, 1];	x[0, 0, 2];	x[0, 0, 3];
x[0, 1, 0];			
x[1, 0, 0];	x[1, 0, 1];	x[1, 0, 2];	
x[1, 1, 0];	x[1, 1, 1];	x[1, 1, 2];	

Dizilerde Kullanılan Metotlar

GetLength() Metodu

Bir dizinin eleman sayısının ne kadar olduğunu bulmak için kullanılır. Dizi türü farketmeksizin(int,float,string...) sayma işlemini yapar. Kullanımı:

Örnek:

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi_duzensiz
{
    class Program
    {
        static void Main(string[] args)
        {
            string [] dizi = {"C#","Java","Asp.Net","Php","Asp"};
            Console.WriteLine(dizi.GetLength(0));
            Console.ReadKey();
        }
    }
}

```

Program çalıştırıldığında ekrana 5 yazacaktır. Dizi elemanlarında 5 tane string türünden veri olduğu için dizi.GetLength(0) metodu ile sayma işlemini gerçekleştirdik.

Dizilerde Kopyalama Metodu

Örnek:

```

float[] x = { 55, 47, 88, 71, 19 };
float[] y = new float[20];
x.CopyTo(y,0);

```

x dizisinin bütün elemanları kopyalandı, y'nin 0. elemanından başlamak şartıyla yapıştırma yapılmıyor.

Örnek:

```

string[] x = {"Ahmet","Mehmet","Ali","Ömer","Osman","Ayşe"};
string[] y = new string[15];
Array.Copy(x, 3, y, 5, 2);

```

x dizisinin 3. elemanından itibaren yani Ali'den sonra 2 tane eleman kopyalandı (Ömer Osman), y dizisinin 5. Elemanından sonrası yapıştır.

Örnek:

```

char[] x = {'m','e','h','m','e','t'};
char[] y = new char[20];
Array.Copy(x, y, 5);

```

x dizisindeki 5 eleman kopyalandı, y dizisine, 0. elemandan itibaren yapıştırılır.

CreateInstance, Dizi Sıralama ve Dizi Arama Metotları

Dizi tanımlamanın bir başka yoluda bu metodu kullanarak yapabiliriz. Bu metot ile dizileri sıralamada sıkılıkla kullanılır.

```
Array x = Array.CreateInstance(typeof(string), 7);
```

Kod satırımızda string türünden 7 elemanlı bir dizi tanımladık. Dizi atamasını daha önce kullandığımız gibi(örnek: x[0]=""metin") atama yapılamaz. Bunun için bu metot kullanımında aşağıdaki maddelere dikkat ederek kullanmamız gerekmektedir.

- **GetValue:** Dizinin belirli bir indeksini tutar
- **SetValue:** Dizinin belirlenen değere atama yapmak için kullanılır
- **GetUpperBound:** Dizinin son indeksinin numarasını verir

Örnek:

CreateInstance metodu kullanarak dizileri sıralayan bir program yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi_kopyalama
{
    class Program
    {
        static void Main(string[] args)
        {
            Array x = Array.CreateInstance(typeof(string), 5);

            x.SetValue("Mehmet", 0);
            x.SetValue("Akif", 1);
            x.SetValue("Fatih", 2);
            x.SetValue("Emre", 3);
            x.SetValue("Yıldırım", 4);

            foreach (string y in x)
                Console.WriteLine(y);
            Console.ReadKey();
        }
    }
}
```

Programımız çalıştırıldığında ekrana dizi elemanları alt alta gelecek şekilde foreach döngüsü ile yazdırılır. Eğer dizi elemanları sıralı bir şekilde olmasını istersek, program kodlarını şöyle değiştirebiliriz:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi_kopyalama
{
    class Program
    {
        static void Main(string[] args)
        {
            Array x = Array.CreateInstance(typeof(string), 5);
```

```

        x.SetValue("Mehmet", 0);
        x.SetValue("Akif", 1);
        x.SetValue("Fatih", 2);
        x.SetValue("Emre", 3);
        x.SetValue("Yıldırım", 4);

        Array.Sort(x);// Dizileri alfabetik sırala

        foreach (string y in x)
            Console.WriteLine(y);

        Console.ReadKey();

    }
}

```

Not: Dizi sıralaması yapıldığında yalnızca tek boyutlu diziler sıralınır.

Örnek:

Dizelerde arama yapan programımızı yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace dizi_kopyalama
{
    class Program
    {
        static void Main(string[] args)
        {
            Array x = Array.CreateInstance(typeof(string), 5);

            x.SetValue("Mehmet", 0);
            x.SetValue("Akif", 1);
            x.SetValue("Fatih", 2);
            x.SetValue("Emre", 3);
            x.SetValue("Yıldırım", 4);

            Array.Sort(x);//dizileri alfabetik sırala
            Console.Write(Array.BinarySearch(x, "Mehmet"));

        }
    }
}

```

Programımız çalıştığında ekranımıza 3 sayısı gelecektir. BinarySearch metodu ile, mehmet elemanını dizi içinde aradık, bulursa array ile sıralanan şekliyle nesnenin indeksini tutar, eğer bulamazsa negatif bir sayı tutar. BinarySearch'ü kullanabilmek için arraySort ile sıralamalıyız.

ArrayList Add Metodu

Bu metot ile diziye değer ekleme yapılır. ArrayList ile değer eklemeye yapılrken farklı veri tiplerine sahip değerler atanabilir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections; //ArrayList için kütüphane dosyası eklendi.

namespace dizi_add_list
{
    class Program
    {
        static void Main(string[] args)
        {
            ArrayList x = new ArrayList();

            x.Add("Mehmet");
            x.Add(258);
            x.Add('?');
            x.Add(356.58);

            foreach (object y in x)
                /* diziye eklenen veriler her türden olduğu için
                   foreach yapısında object olarak tanımlandı*/
                Console.WriteLine(y);

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

ArrayList Insert Metodu

Insert metodu ile dizi içeresine araya değer vermek için kullanılan metottur. 2 parametre alır. Birinci parametrede verilen indeks numarasına, ikinci parametre olarak verilen değer eklenir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections; //ArrayList için kütüphane dosyası eklendi.

namespace dizi_add_list
{
    class Program
    {
        static void Main(string[] args)
        {
            ArrayList x = new ArrayList();
```

```

        x.Add("Mehmet");
        x.Add(258);
        x.Add('?');
        x.Add(356.58);

        x.Insert(3, "Ahmet");//? ile 356.58 arasına Ahmet eklenir.

        foreach (object y in x)
            /* diziye eklenen veriler her türden olduğu için
               foreach yapısında object olarak tanımlandı*/
            Console.WriteLine(y);

        Console.ReadKey();

    }
}

```

Not:

- Array.Sort(x);// Sıralama yapar
- Array.Clear(x,2,1);// 2. İndeksinden sonra 1 elemanını temizle
- Array.Reverse(x);// Dizi elemanlarını ters çevir

Hata Yakalama Mekanizması(try,catch,finally)

Programımız şuana kadar tanımladığımız ifadelerin doğru bir şekilde işledeğinde herhangi bir problem çıkartmıyordu. Ama problem çıkmayacak veya problemsiz veri girişi işleyisi olacak diye bir kaide olamaz. Elbette ki madde olan herşeyin bir hata yapma olasılığı vardır. Program tasarlarken de ne gibi hataların oluşabileceğini kestirebilmek gerekiyor. Nitekim bir yazılım ne kadar basit ve çok kişiye hitap edip sorunsuz bir hata yapma olasılığı var. Hatta bu hata yakalama ve buna göre nasıl işlem yürütüleceğinden bahsedeceğiz. Hatanın türüne bağlı olarak hata yakalama yöntemleri var. Bunları örnekler halinde inceleyelim.

Örnek:

Klavyeden giriş yapılan 2 sayının toplamını veren programı, try catch komutunu kullanarak sayının haricinde bir giriş yapılması durumunda hata mesajı veren programımızı yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace try_catch_kullanimi
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                int x,y,toplam;
                Console.Write("Birinci Sayı:");

```

```
        x = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("İkinci Sayı:");
        y = Convert.ToInt32(Console.ReadLine());
        toplam = x + y;
        Console.WriteLine("Toplam Sonucu: "+toplam);
    }
    catch
    {
        Console.WriteLine("Oops sayı giriniz!!!\"");
    }
    Console.ReadKey();
}
}
```

Programa normalde tamsayı girildiği durumda try bloku içerisinde hatasız bir şekilde çalışacaktır. Eğer klavyeden harf veya kesirli sayı veya tamsayı haricinde bir şey girildiğinde hata uyarısı için catch bloğu çalışacaktır.

Hatanın türünüde tespit edip, örneğin klavyeden girilen sayı int değişkenin boyutunu aşarsa veya sayı girişi yerine harf girişi yapılrsa veya tanımlanan dizi değişkenin tanımlanmayan bir dizisi ile işlem yapılacaksa programda try catch bloklarına aşağıdaki gibi ekleme yapmak gereklidir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace try_catch_kullanimi
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                int x,y,toplam;
                Console.Write("Birinci Sayı:");
                x = Convert.ToInt32(Console.ReadLine());
                Console.Write("İkinci Sayı:");
                y = Convert.ToInt32(Console.ReadLine());
                toplam = x * y;
                Console.WriteLine("Toplam Sonucu: "+toplam);
            }

            catch (FormatException hata) //Sayı yerine harf girişi yapılınrsa bu blok çalışacak
            {
                Console.WriteLine("Sayı girmelisiniz!" +hata.Message);
            }

            catch (OverflowException hata1)//int'in kapasitesinin üzerine çıkarsa burası çalışacak
            {
                Console.WriteLine("Giriş Sayısı çok büyük!" + hata1.Message);
            }

            catch (IndexOutOfRangeException hata2)//Dizi kapasitesi aşıldı
            {
                Console.WriteLine("Dizi indeksi geçerli değil!");
            }
        }
    }
}
```

```

        {
            Console.WriteLine("Tanımlanmayan bir diziye atama yapıldı. "+hata2.Message);
        }

        Console.ReadKey();
    }
}

```

Eğer programımızda hata olsada olmasada çalışacak bir yapı varsa buda finally bloğunu içerisine tanımlanır. Yukardaki örneğimizde toplam işlemi yapın veya yapmasın eğer toplam sonucunun 0'a eşit olmasını istiyorsak finally bloğunu içerisine aşağıdaki ifadeyi eklememiz gereklidir.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace try_catch_kullanimi
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, y, toplam;

            try
            {
                Console.Write("Birinci Sayı:");
                x = Convert.ToInt32(Console.ReadLine());
                Console.Write("İkinci Sayı:");
                y = Convert.ToInt32(Console.ReadLine());
                toplam = x * y;
                Console.WriteLine("Toplam Sonucu: "+toplam);
            }

            catch (FormatException hata) //Sayı yerine harf giriş yapılırsa bu blok çalışacak
            {
                Console.WriteLine("Sayı girmelisiniz!" +hata.Message);
            }

            catch (OverflowException hata1)//int kapasitesinin üzerine çıkarsa burası çalışacak
            {
                Console.WriteLine("Giriş Sayısı çok büyük!" + hata1.Message);
            }

            finally
            {
                toplam = 0;
                Console.WriteLine("Toplam sonucu 0'a eşit yapıldı.");
            }
        }
    }
}

```

Rastgele Sayı Üretme

Rastgele sayı üretebilmemiz için Random türünden nesne oluşturulması gereklidir. Daha sonra random ile tanımlanan nesne ile random sınıf metotları kullanılabilir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace random
{
    class Program
    {
        static void Main(string[] args)
        {
            Random x = new Random();

            int y1 = x.Next(5, 10);
            //1 ile 10 arası rastgele sayı üretir.(1 dahil ancak 10 dahil değildir)
            Console.WriteLine(y1);

            int y2 = x.Next(50);
            //0 ile 50 arası rastgele sayı üretir.(0 dahil ancak 50 dahil değildir)
            Console.WriteLine(y2);

            int y3 = x.Next();
            //Pozitif herhangi bir tamsayı üretir.
            Console.WriteLine(y3);

            double y4 = x.NextDouble();
            //0.0 ile 1 arası rastgele sayı üretir.
            Console.WriteLine(y4);

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

Metotlar

Bir kod ifadesini birden fazla yerde tanımladığımızda kodlarımız uzamakta sürekli olarak tekrara tekrar yazmak zorunda kaldığımız zamanlar olabilir. Program içerisinde misal aynı kod parçasını defalarca kullandık. Eğer bu kod parçasında bir yer değiştirmek istediğimizde defalarca aynı kod parçasını bulup değiştirmek zorunda kalabilirdik. Böyle bir durumda çok fazla iş yükü ortaya çıkacaktır. İşte bu gibi durumların önüne geçmek için metotlar kullanılabilir. 1 den fazla yerde kullanacağımız bir ifadeyi tek bir yerde tanımlayıp bunu istediğimiz zaman çağırıp işleme tabi tutabiliriz. Metotlar ile ilgili bazı yapısal farklılıklar var bunları başlıklar halinde inceleyelim.

- **private:** Bu yapı ile belirlenen bir metot yalnızca tanımlandığı sınıf içerisinde erişilebilir. Bu metodu programımız içinde kullanabilmemiz için metodun içinde bulunduğu sınıf türünden bir nesne oluşturmamız gereklidir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace metotlar2
{
    class Program //nesne oluşturmada kullanılacak ad
    {
        private void x()
        {
            Console.WriteLine("C# Programlama \n Java Programlama");
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Programlama Dilleri: ");
            Program metot1 = new Program(); //private ile tanımlanmış metod için nesne tanımlandı
            metot1.x();
            Console.ReadKey();
        }
    }
}
```

- **static:** Metotlar dahil oldukları sınıf adları ile birlikte çağrılabılırler. static olarak tanımlanan bir metot ana programdan çağrılrken sınıf adını yazmaya gerek yoktur.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace metotlar
{
    class Program
    {
        static void x()
        {
            Console.WriteLine("C# Programlama \n Java Programlama");

        }
        static void Main(string[] args)
        {
            Console.WriteLine("Programlama Dilleri: ");
            x();
            Console.ReadKey();
        }
    }
}
```

```
    }  
}
```

- **public:** Program içerisinde herhangi bir alanda metot'dun çağrılabilmesi için erişim şeklinin public olarak tanımlanması gerekmektedir. Aşağıda iki farklı sınıf tanımlanmış. Metot1 sınıfının içeriğindeki metod public olarak tanımlandığı için her tarafından erişilme imkanı vardır.

Örnek:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace metotlar2  
{  
    class Program  
    {  
        class metot1  
        {  
  
            public static void x()  
            {  
                Console.WriteLine("C# Programlama \n Java Programlama");  
  
            }  
        }  
        class metot2  
        {  
            static void Main(string[] args)  
            {  
                Console.WriteLine("Programlama Dilleri: ");  
                metot1.x();  
  
                Console.ReadKey();  
            }  
        }  
    }  
}
```

Main Metodu

Programın ilk çalışma noktasıdır. Her programda bulunması gereklidir. Main metodu kitaplara ve hizmetlere ihtiyaç duymaz. Burada program başlayıp ve biter. Main ana yapısı değiştirilebilir. Metotlar main bloğu içerisinde çağrıılır.

Metotlar İle İlgili Özellikler

Metotlar çağrırlarken beraberinde götürülen değişken türleri farklı türden de olabilir. Örneğin string değişken türü ile int değişken türü aynı metot içerisinde kullanılabilir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Metotlar_ilk
{
    class Program
    {
        static void metot(string x, double y)
        {
            Console.WriteLine(x);
            if (y < 10)
                Console.WriteLine("10'dan küçük sayı girdiniz");
        }
        static void Main(string[] args)
        {
            metot("Merhaba", 8);
            Console.ReadKey();
        }
    }
}
```

Metotda tanımlanan ifadeleri klavyeden atanan değerlere göre tanımladığımızda yukarıdaki örneğimizi aşağıdaki şekilde kullanabiliriz.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Metotlar_ilk
{
    class Program
    {
        static void metot(string x, double y)
        {
            Console.WriteLine(x);
            if (y < 10)
                Console.WriteLine("10'dan küçük sayı girdiniz");
        }
        static void Main(string[] args)
        {
            string x;
            double y;
            Console.Write("Metinsel bir ifade giriniz:");
            x = Console.ReadLine();
            Console.Write("Sayısal bir ifade giriniz:");
            y = Convert.ToDouble(Console.ReadLine());
            metot(x, y);
            Console.ReadKey();
        }
    }
}
```

Örnek:

Klavyeden girilen 2 sayıyı çarpıp sonucunu ekrana yazdırın programı metot kullanarak yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace metotlar2
{
    class Program
    {
        private static float carp(float a, float b)
        {
            float carpim = a * b;
            return carpim;
        }

        static void Main(string[] args)
        {
            float a, b, c;
            Console.Write("1. Sayı:");
            a = Convert.ToSingle(Console.ReadLine());
            Console.Write("2. Sayı:");
            b = Convert.ToSingle(Console.ReadLine());

            c = carp(a,b);
            Console.WriteLine(c);
            Console.ReadKey();
        }
    }
}
```

Örnek:

Klavyeden girilen sayının tek mi çift mi olduğunu bulan programı metotlar yardımıyla yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace metotlar2
{
    class Program
    {
        private static int cift(int a)
        {
            int x = a % 2;
            return x;
        }
    }
}
```

```

static void Main(string[] args)
{
    int a, c;
    Console.Write("Sayı:");
    a = Convert.ToInt32(Console.ReadLine());

    c = cift(a);
    if(c==0)
        Console.WriteLine("Girilen Sayı Çifttir");
    else
        Console.WriteLine("Girilen Sayı Tektir");
    Console.ReadKey();
}

}

```

Örnek:

Dizi elemanlarını metot kullanarak ekrana yazdıralım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace metotlar2
{

    class Program
    {
        static void dizi_metot(int[] a)
        {
            foreach (int b in a)
                Console.WriteLine(b);
        }
        static void Main(string[] args)
        {
            int[] x = { 1, 2, 4, 7, 9 };
            dizi_metot(x);

            Console.ReadKey();
        }
    }
}

```

Burada kullandığımız metotda dizi sayıları belli olan elemanları ekrana yazdırıldı.

Params Metodu

Şimdije kadar kullandığımız metodlara belirli sayıda değişken ataması yapalıydık. Metotlarda kullandığımız değişken tanımı sabit bir değer olarak tanımlandığı için bunu üzerine eklem ya da çıkarma yapılamıyordu. Params yardımıyla tanımlanmış olan metodda sabit değişken değer aralığı yok. Yani metodumuza 2 değer gönderip daha sonra 5 yada 10 yada daha

farklı sayıda değişken gönderip bunları gönderilecek sayılarla göre metodda işleme tabi tutabiliyoruz.

Örnek:

Main metodumuzun içinden class yapımızın içinde tanımlanmış olan metodumuza farklı sayıarda değer atamasını atayıp bunları foreach yardımıyla çarpma işlemine tabi tutup sonucunu ekrana yazan programımızı yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace metotlar_degisken_sayida
{
    class Program
    {
        public static int x(params int[] y)
        {
            int carp=1;
            foreach (int i in y)
                carp *= i;
            return carp;
        }
        static void Main(string[] args)
        {
            int a;
            a= x(2, 5, 5, 3);
            Console.WriteLine(a);
            a = x(20, 2);
            Console.WriteLine(a);

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

Program çalıştırıldığında ekranımıza alt alta 150 ve 40 değerlerini yazacaktır.

Not: params metodu kullanarak, şart ile birlikte belirlenen sayılar beraberinde metoda gönderim sağlanabilir. Aşağıdaki örnekte sayılar ile birlikte şartlı bir ifade de sayılar ile birlikte gönderimi sağlanmıştır.

Örnek:

Params değişken metodu kullanılarak 6 sayının ortalamasını hesaplayıp metotdan gelen ortalama sonucunun 5'ten büyük olması durumunda ekrana ortalama sonucu 5'ten büyük, değilse ortalama sonucu 5'ten küçük yazdırılsın.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace metotlar_params
{
```

```

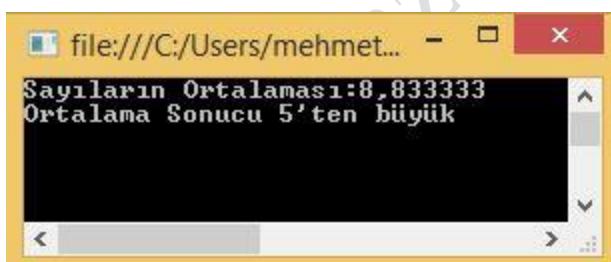
class Program
{
    public static float durum(string a, params float[] gelen)
    {
        if (a == "ortalama")
        {
            float hesap=0,ort=0;
            int sayi=0;
            foreach (float x in gelen)
            {
                sayi++;
                hesap += x;
                ort = hesap / sayi;

            }
            Console.WriteLine("Sayıların Ortalaması:"+ort);
            return ort;
        }
        else
            return 0;
    }
    static void Main(string[] args)
    {

        if (durum("ortalama", 5, 6, 9, 8, 10, 15) > 5)
            Console.WriteLine("Ortalama Sonucu 5'ten büyük");
        else
            Console.WriteLine("Ortalama Sonucu 5'ten küçük");

        Console.ReadKey();
    }
}

```



Programımız çalıştırıldığında yandaki sonuç verecektir.

`durum("ortalama", 5, 6, 9, 8, 10, 15)`

Metot tanımında şartlı bir ifade olarak ortalama kullanılıyor. Şartlı ifade metot içerisinde tanımlı olan blok içerisinde sağlandığında, beraberinde götürmiş olduğu sayıları işleme tabii tutuyor

Aynı Metot Adına Sahip, Değişken Türleri Farklı Olan Metotlar

Bir metot ismi içeriğindeki değişken türleri farklı olmak koşulu ile birden fazla aynı ada sahip metot yazılabilir. Aynı ada sahip metotlar çağrılrken çağrılan metoda gelen veri türlerini kıyaslar. Veri türü hangi metodda geçerli olacağsa o metodda işleme tabii tutulur. Çağırılan metot türlerinde metoda gelen veri türü eğer bu metotlardan herhangi biriyle uyum sağlanmıyorsa hata durumu oluşacaktır.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace metot_ayni_ad
{
    class Program
    {

        public static void b(double x, double y)
        {
            Console.WriteLine("Toplam Sonucu:{0}",(x+y));
        }
        public static void b(string x, char z, string y)
        {

            Console.WriteLine("Merhaba, {0}",(x+z+y));
        }
        static void Main(string[] args)
        {
            b(528.36, 63);
            b("Mehmet", '&', " Ahmed");

            Console.ReadKey();
        }
    }
}

```



Program çalıştırıldığında yandaki sonuç olacaktır. Aynı isimde iki farklı değişken türüne sahip b isimli metot tanımladık. Metotlar çağrılrken içeriğindeki değişken yapısına bağlı olarak işleme tabii tutuldu. Metotları çağrılrken metodlarda tanımlı olmayan bir değişken türü kullandığımızda programımız hata verecektir.

Not: Aynı ada sahip metotlar kullanıldığında, değişken türler yapılarında birbirine benzer ise metot çağrıımı yapılrken en az kapasiteli değişken türlü metot çağrırlır.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace metot_ayni_ad
{
    class Program
    {

        public static void b(double x, double y)

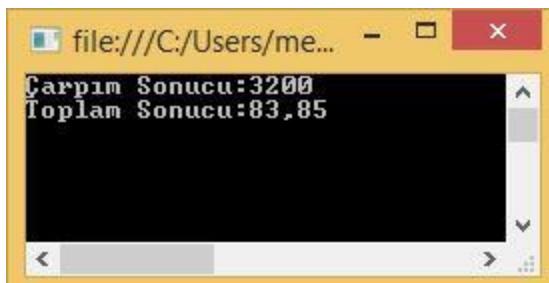
```

```

{
    Console.WriteLine("Toplam Sonucu:{0}",(x+y));
}
public static void b(float x, float y)
{
    Console.WriteLine("Çarpım Sonucu:{0}", (x*y));
}
static void Main(string[] args)
{
    b(25, 128);
    b(25.2, 58.65);

    Console.ReadKey();
}
}

```



b(25, 128);

Bu metod çağrılrken en az kapasiteli olan float tanımlı metod çağrılrır.

b(25.2, 58.65);

Bu metodda ise double tanımlı metod çağrılracaktır.

Özyineli (Rekürsif-Recursive) Metotlar

Bir metodun içerisinde yine kendisinden bir yapı bulunuyorsa bu metodlara özyineli metodlar denir. Burada metod yapısı kendisini tekrar eder. Örneğin faktöriyel hesabı yapan programı Özyineli ile kullanarak ve özyineli kullanmayarak yapalım.

Özyineli Kullanım ile faktöriyel hesabı:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Metotlar_ilk
{
    class Program
    {
        static int fakt(int x)
        {
            if (x == 0)
                return 1;
            return x * fakt(x - 1);
        }
        static void Main(string[] args)
        {

            int y;

```

```

        Console.WriteLine("Sayı giriniz:");
        y = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Girilen Sayının Faktoriyeli:"+fakt(y));

        Console.ReadKey();
    }

}

```

Özyineli metot kullanmayarak yapılan faktöriyel programı:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Metotlar_ilk
{
    class Program
    {
        static int fakt(int x)
        {
            int sonuc=1;
            if (x == 0)
                return 1;
            for (int i = 1; x > 1; x--)
            {
                i *= x;
                sonuc = i;

            }
            x = sonuc;
            return x;
        }
        static void Main(string[] args)
        {

            int y;

            Console.WriteLine("Sayı giriniz:");
            y = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Girilen Sayının Faktoriyeli:"+fakt(y));

            Console.ReadKey();
        }
    }
}

```

Her iki programdada sonuç aynı olacaktır. Rekursif kullandığımız zaman ayrıca for döngüsüne ihtiyaç kalmadan işlevini kendi içerisinde döngüye tabii tutarak faktöriyel hesabını yapmış oldu. Rekursif kullanarak az kodla aynı işi yaptırmış olduk.

Metinsel Metotlar

Compare() Metodu

Bu metot iki nesnenin karşılaştırılabilmesini sağlar. Compare metodu -1, 0 veya 1 değeri döndürür. Karşılaştırma işlemini büyülük, küçüklük ilişkisine göre yapar. Compare() metodunun kullanımı aşağıdaki gibidir:

```
int krs = String.Compare(x, y);
```

Örnek:

Klavyeden girilen 2 metni büyülük olarak birbirile karşılaştırmasını yapan programı compare metodunu kullanarak yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace hazir_metotlar
{
    class Program
    {
        public static void karsilastir(string metin1, string metin2)
        {
            int krs = String.Compare(metin1, metin2);
            if (krs == 1)
                Console.WriteLine("1. Metin 2. Metinden Büyüktür.");
            else if(krs== -1)
                Console.WriteLine("2. Metin 1. Metinden Büyüktür.");
            else
                Console.WriteLine("1. Metin ile 2. Metin birbirine eşittir.");
        }
        static void Main(string[] args)
        {
            string x, y;
            Console.WriteLine("Birinci metni giriniz:");
            x = Console.ReadLine();
            Console.WriteLine("İkinci metni giriniz:");
            y = Console.ReadLine();
            karsilastir(x, y);
            Console.ReadKey();
        }
    }
}
```

Concat() Metodu

Metinleri birleştirmek için kullanılır. Aşağıdaki şekilde metot kullanılabilir.

```
string x = String.Concat(Metin1, Metin2, Metin3, ....);
```

yada params metodu da kullanılarak istenildiği kadar değişkenleri birleştirip kullanılabilir.

```
static string Concat(params Array string[] x)
```

Örnek:

Klavyeden girilen 2 metni concat yardımıyla birlestirelim.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace string_metotlar
{
    class Program
    {

        static void birlestir(string x, string y)
        {
            string br = String.Concat(x,y);
            Console.WriteLine(br);
        }
        static void Main(string[] args)
        {
            string x, y;
            Console.WriteLine("Birinci metni giriniz:");
            x = Console.ReadLine();
            Console.WriteLine("İkinci metni giriniz:");
            y = Console.ReadLine();
            birlestir(x, y);
            Console.ReadKey();
        }
    }
}
```

Format() Metodu

Farklı formatlarda görüntüleme yapılımak istenen değerleri String.Format metodu ile gerçekleştirebilirsiniz.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

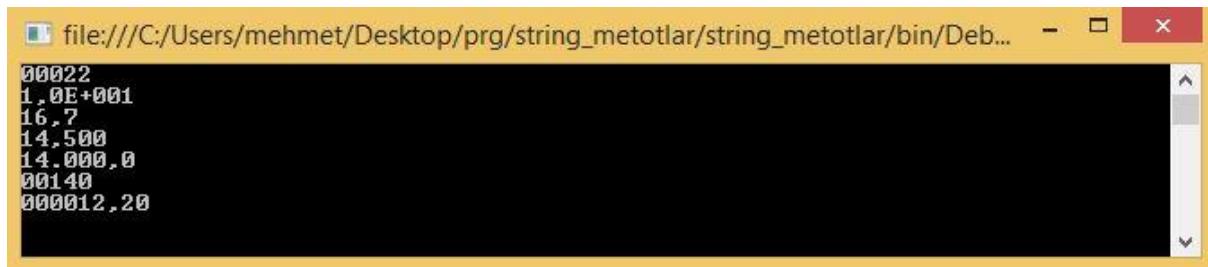
namespace string_metotlar
{
    class Program
    {

        static void Main(string[] args)
        {

            Console.WriteLine(String.Format("{0:D5}", 22));
            Console.WriteLine(String.Format("{0:E1}", 10));
            Console.WriteLine(String.Format("{0:F1}", 16.7));
            Console.WriteLine(String.Format("{0:F3}", 14.5));
            Console.WriteLine(String.Format("{0:N1}", 14000));
            Console.WriteLine(String.Format("{0:X5}", 320));
            Console.WriteLine(String.Format("{0:000000.00}", 12.2));

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

```
    }  
}
```



```
file:///C:/Users/mehmet/Desktop/prg/string_metotlar/string_metotlar/bin/Deb... - □ ×  
0.0022  
1.0E+001  
16.7  
14.500  
14.000,0  
00140  
000012.20
```

IsNullOrEmpty() Metodu

String türündeki değişkenin içeriğinin boş mu, dolu mu olduğunu kontrol eder. Değişkenin içeriği boş ise bool değişken türünden true değerini atamasını yapar. Değişken içeriği boş değilse false değerini atar.

Örnek:

Klavyeden veri girilip girilmediğini kontrol eden programımızı IsNullOrEmpty metodu ile yapalım.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace string_metotlar  
{  
    class Program  
    {  
        static void bosmu(string x)  
        {  
            bool durum = String.IsNullOrEmpty(x);  
            if (durum == true)  
                Console.WriteLine("Lütfen e-posta adresinizi yazınız!");  
            else  
                Console.WriteLine("e-posta adresi eklendi");  
        }  
  
        static void Main(string[] args)  
        {  
            string x = null;  
            Console.Write("e-posta adresini yazınız:");  
            x = Console.ReadLine();  
            bosmu(x);  
            Console.ReadKey();  
        }  
    }  
}
```

Contains() Metodu

String değişken türünden tanımlanmış olan ifadenin içerisinde arama yapar. Bool değişken türü ile kullanılabilir. Aranan değişken türünün içeriğinde aranan ifade varsa true değerini geri gönderirir, yoksa false değerini gönderir. Arama yaparken harf harf arama yapar. Arama yapılan ifadede herhangi bir harf yakaladığında true değişkenini geri gönderir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace string_metotlar
{
    class Program
    {
        static void Main(string[] args)
        {

            string cümle = "Mehmet Akif Arvas";
            bool kontrol = cümle.Contains("M");
            Console.WriteLine(kontrol);
            Console.ReadKey();

        }
    }
}
```

EndsWith() ve StartsWith () Metotları

EndsWith metodu ile tanımlanan bir string değişken türündeki ifadenin ne ile bittiğini kontrol etmek için, StartsWith metodu ile tanımlanan bir string değişken türündeki ifadenin ne ile başladığını kontrol etmek için kullanılır.

Örnek:

Önceden string değişken türüne atanan internet adresinin doğruluğunu kontrol eden yapıyı endswith ve startswith komutları ile yapalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace string_metotlar
{
    class Program
    {
        static void Main(string[] args)
        {
            String x = "www.gmail.com" ;

            if (x.EndsWith(".com"))
            {
                if (x.StartsWith("www."))
                    Console.WriteLine("Girilen internet adresi doğrulandı.");
                else
                    Console.WriteLine("Girilen internet adresi doğrulanamadı.");
            }
        }
    }
}
```

```

        }
        else
            Console.WriteLine("Girilen internet adresi doğrulanamadı.");
        Console.ReadKey();
    }
}

```

ToUpper() ve ToLower() Metotları

ToUpper metodu ile harfleri büyütmek için, ToLower metodu ile de harfleri küçültmek için kullanılır.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace string_metotlar
{
    class Program
    {
        static void Main(string[] args)
        {
            string x = "C# PROGRAMLAMA";

            String kucuk = x.ToLower();
            Console.WriteLine(kucuk);

            String buyuk = kucuk.ToUpper();
            Console.WriteLine(buyuk);
            Console.ReadKey();
        }
    }
}

```

IndexOf() ve LastIndexOf Metodu

IndexOf metodu string içerisinde verinin kaçinci karakter olduğunu index türünden int olarak çıktı verir, eğer bulamazsa -1 verir. LastIndexOf metodu en sondaki bulduğu aynı ifadenin indexini verir. İlk index değeri programlamada her zaman 0 olarak geçer.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace string_metotlar
{
    class Program
    {
        static void Main(string[] args)
        {
            string x = "Programlama Temelleri";

```

```

        int y = x.IndexOf("m");// Aranan ifadenin ilk indexini verir.
        int z = x.LastIndexOf("m"); //Aranan ifadenin son indexini verir.

        Console.WriteLine("Aranan harfin indexi: "+y);// 6 yazar
        Console.WriteLine("Aranan harfin son indexi: " + z); // 14 yazar

        Console.ReadKey();
    }
}
}

```

PadLeft () ve PadRight () Metodu

String değişken türüyle tanımlanmış ifadenin soluna ve sağına tek karakterlik herhangi bir ifade eklemek için kullanılır. PadLeft ile string ile tanımlanmış ifadenin soluna ekleme yapar, PadRight ile string ile tanımlanmış ifadenin sağına ekleme yapar. Eklenecek olan ifade char türünden olması gereklidir. Eklenen ifadenin başlangıç noktası 0. İndeksten başlar.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

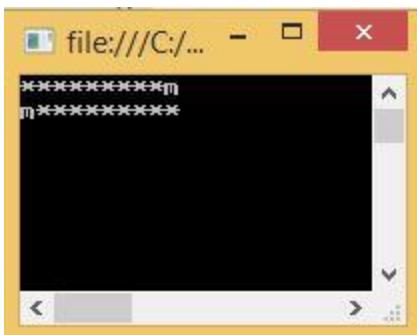
namespace string_metotlar
{
    class Program
    {
        static void Main(string[] args)
        {
            string x = "m";

            string xLeft = x.PadLeft(10, '*');
//m'nin Soluna index 0'dan başlayacak şekilde * bırak
            Console.WriteLine(xLeft);

            string xRight = x.PadRight(10, '*');
//m'nin Sağına index 0'dan başlayacak şekilde * bırak
            Console.WriteLine(xRight);

            Console.ReadKey();
        }
    }
}

```



Program çalıştırıldığında string türünden tanımlanmış m ifadesinin sağına ve soluna 9 tane * eklendi. 0. İndekste m olduğu için yıldız bırakılmadı. Eğer tanımladığımız ifade ile aynı sayıda karakter bırakılırsa bırakılan ifade gözükmeyecektir.

Remove() Metodu

Belirtilen ifadenin kaçinci indexinden sonra, kaç tane karakterin kaldırılacağını bu metodu kullanarak yapabiliriz. Kaldırılan karakterlere boşluklarda dahildir.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace string_metotlar
{
    class Program
    {
        static void Main(string[] args)
        {
            string x = "C# Programlama Dili";
            string y = "Asp.Net Web Programlama";
            string z = "Web Tasarım ve Programlama";

            Console.WriteLine(x.Remove(4, 3));
            // 4. indexten sonra 3 tane karakter sil.
            Console.WriteLine(y.Remove(1, 4));
            // 1. indexten sonra 4 tane karakter sil.
            Console.WriteLine(z.Remove(0, 3));
            // 0. indexten sonra 3 tane karakter sil.

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

Replace() Metodu

Metnin içerisinde değiştirilmesi istenen karakterin başka bir karakter ile değiştirmesini sağlar.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```

namespace string_metotlar
{
    class Program
    {
        static void Main(string[] args)
        {
            string x = "C# Programlama Dili";

            string degisen = x.Replace('ı','i');
            // ı harfleri i olarak değiştirildi

            Console.WriteLine("Son hali: "+ degisen);

            Console.ReadKey();
        }
    }
}

```

Split() Metodu

String olarak tanımlanan değişkenin içeriğini ayırcı dizisindeki tanımlanmış karakter yardımıyla parçalara ayırır. Sonuçlar string dizisinde tutulur.

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace string_metotlar
{
    class Program
    {
        static void Main(string[] args)
        {
            string x = "C#, Java, Asp.net";

            string[] y = x.Split(',');
            Console.WriteLine(y[1]);

            Console.ReadKey();
        }
    }
}

```

Tanımlanan ayırcı ile x değişkenin içerisindeki metin ifadesi parçalara ayrıldı. Her bir ayraçtan sonraki ifadeyi dizi değişken yardımıyla kullanılabilir. Yukarıdaki örnekte dizi[1] elemanı olarak ekrana java yazdırılacaktır.

Trim() Metodu

Bir string değişkenin içerisinde tanımlı olan ifadenin başındaki ve sonundaki boşlukları kaldırmak için kullanılır.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace string_metotlar
{
    class Program
    {
        static void Main(string[] args)
        {
            string x = " C#, Java, Asp.net ";
            string y= x.Trim();
            Console.WriteLine(y);
            Console.ReadKey();
        }
    }
}
```

Matematiksel Metotlar

Sqrt() Metodu

Değişken türü sayı olarak tanımlanmış bir ifadenin karekökünü almak için kullanılan matematiksel metottur.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace math_metotlari
{
    class Program
    {
        static void Main(string[] args)
        {
            double x;
            Console.WriteLine("Sayı Giriniz:");
            x = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Girilen sayının karekökü =" + Math.Sqrt(x));
            Console.ReadKey();
        }
    }
}
```

Sin(), Cos(), Tan(), Abs() Metotları

Sayıların sinüs, cosinüs, tanjant ve mutlak değerini almak için kullanılan metotlar aşağıdaki şekilde kullanılır.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace math_metotlari
{
    class Program
    {
        static void Main(string[] args)
        {
            double x;
            Console.WriteLine("Sayı Giriniz:");
            x = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Girilen sayının Sinüsü ={0}", Math.Sin(x));
            Console.WriteLine("Girilen Sayının Cosinüsü={0}", Math.Cos(x));

            Console.WriteLine("*****");
            Console.WriteLine("Girilen sayının Tanjantı ={0}", Math.Tan(x));
            Console.WriteLine("Girilen Sayının Mutlak Değeri={0}", Math.Abs(x));

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

Diger Matematiksel Metotlar

Matematiksel Metotlar	Açıklama
Math.Ceiling(x)	x sayıından büyük en küçük tam sayıyı yukarı yuvarla
Math.Floor(x)	x sayıından küçük en büyük tam sayıyı aşağı yuvarla
Math.Max(x,y)	x ve y sayılarının en büyüğü
Math.Min(x,y)	x ve y sayılarının en küçüğü
Math.Pow(x,y)	x üzeri y işlemini yapar
Math.Log(x)	x sayısının e tabanında logaritmasını alır
Math.Exp(x)	e üzeri x'in değerini hesaplar
Math.Log5(x)	x sayısının 10 tabanındaki logaritmasını hesaplar
Math.Round(x, 3)	Sayıının ondalık hassasiyetini artırır. 3 sayısı virgülinden sonra kaç basamak ondalık olacağını belirtir
Math.Sign(x)	x sayısının pozitif mi yada negatif mi olduğunu kontrol eder. Sayı pozitif ise 1 negatif ise -1 sıfıra eşitse 0 geriye gönderir
BigMul(x,y)	x ve y sayısını çarpıp sonucunu long türündeki ifadenin içerisinde tutar
Math.Acos(x)	x sayısının kosinüs değerinin radyan açı değerini verir
Math.Asin(x)	x sayısının sinüs değerinin radyan açı değerini verir
Math.Atan(x)	x sayısının tanjant değerinin radyan açı değerini verir

Tarih/Saat (DateTime) Metotları

Metot	Örnek Kullanımı	Açıklama
AddMilliseconds()	<code>DateTime bg = DateTime.Now; DateTime x = bg.AddMilliseconds(1);</code>	Bugünün zamanına ne kadar salise ekleneceği için kullanılır
AddSeconds()	<code>DateTime bugun = DateTime.Now; DateTime x = bugun.AddSeconds(1);</code>	Bugünün zamanına ne kadar saniye ekleneceği için kullanılır
AddMinutes()	<code>DateTime bugun = DateTime.Now; DateTime x = bugun.AddMinutes(1);</code>	Bugünün zamanına ne kadar dakika ekleneceği için kullanılır
AddHours()	<code>DateTime bugun = DateTime.Now; DateTime x = bugun.AddHours(1);</code>	Bugünün zamanına ne kadar saat ekleneceği için kullanılır
AddYears()	<code>DateTime bugun = DateTime.Now; DateTime x = bugun.AddYears(1);</code>	Bugünün tarihine ne kadar yıl ekleneceği için kullanılır
AddMonths()	<code>DateTime bugun = DateTime.Now; DateTime x = bugun.AddMonths(2);</code>	Bugünün tarihine ne kadar ay ekleneceği için kullanılır
AddDays()	<code>DateTime bugun = DateTime.Now; DateTime x = bugun.AddDays(10);</code>	Bugünün tarihine ne kadar gün ekleneceği için kullanılır
DateTimeParse()	<code>DateTime y = DateTime.Parse("26-03-2017");</code>	String türünden girilen tarihi kullanılabilecek tarih ve saat bilgisine dönüştür
DateTimeIsLeapYear()	<code>bool y = DateTime.IsLeapYear(1987);</code>	Belirtilen yılın 366'dan büyük olması durumunda true değerini, küçük olması durumunda false değerini gönderir
DateTimeDaysInMonth()	<code>int y=DateTime.DaysInMonth(2015, 5);</code>	Belirtilen yıldaki ayın kaç günden ibaret olduğunu bulur
DateTimeCompare()	<code>int x = DateTime.Compare(DateTime Tarih1, DateTime Tarih2);</code>	İki tarihi karşılaştırır. 1. Tarih büyükse 1, 2. Tarih büyükse -1, tarihler birbirine eşitse 0 gönderir.

Now	<code>DateTime y = DateTime.Now;</code>	Bugünün hem tarih hemde saatini verir
Today	<code>DateTime y = DateTime.Today;</code>	Bugünün tarihini verir
MaxValue	<code>DateTime y = DateTime.MaxValue;</code>	En yüksek tarih ve saatı verir
MinValue	<code>DateTime y = DateTime.MinValue;</code>	En düşük tarih ve saatı verir

Dosya Giriş Çıkış İşlemleri

En temel düzeyde bir dosyayı veritabanı olarak kullanabilmek için bir metin dosyası kullanılabilir. Veritabanına kayıt ekleme yaparken yine bu açılan dosya üzerine kayıt işlemi yapılabilmesi mümkün olmaktadır. Fakat unutulmaması gereken bir şey varki dosya ile bir veri tabanı yürütülmesi oldukça güç ve neredeyse imkansız olmaktadır. Bir okulun öğrencilerinin kimlik adres v.s bilgilerin tümünü bir metin belgesinde kaydetmek ve buradan kullanmak ne kadar güç olacağını tahmin edebilirsiniz. İşte dosya ile ilgili işlemler en temel düzeyde nasıl kullanılabileceğinden bahsedeceğiz. Verilerin net bir şekilde birbirinden ayrılması kategorize edilmesi kategorize edilen bu verileri düzgün ve basit bir yapıda alıp kullanabilmek işte bunların hepsini yapan bir sistem var oda veritabanı sistemleridir. Veritabanı konusunda bunu tafsılatıyla açıklayacağız.

Dosya kaydetme, yazdırma, okuma, arama v.s gibi dosya ve klasör ile ilgili işlemlerin nasıl yapılacağını bu konumuzda anlatmaya çalışacağımız. Dosya ile ilgili işlemlerin kullanılabilmesi için system kütüphane dosyasına aşağıdaki kaynak kütüphane, eklenmesi gereklidir.

```
using System.IO;
```

Temel giriş çıkışlarının kütüphane dosyasını bu şekilde aktif hale getirdik.

Dosya Açıma

C sürücüsünü hemen altında programlama ismi ile bir klasör nasıl açıldığını aşağıdaki örnekte gösterelim.

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO; // Temel Giriş Çıkış Kütüphane dosyası eklendi

namespace dosya_klasor
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("C sürücüsünün altında Programlama klasörü açıldı.");
            Directory.CreateDirectory(@"C:\Programlama");
            Console.ReadKey();
        }
    }
}
```

C sürücüsünün altında klasör açtık. Bu şekilde konum yolunu belirterek istediğimiz alana klasör açabiliyoruz. Eğer programın çalıştığı yere klasör açmak istiyorsak aşağıdaki şekilde direkt klasörün ismini tanımlayabiliriz:

```
Directory.CreateDirectory("Programlama");
```

Şekilde değiştirmek gereklidir. Eğer bir üst klasöre klasör açılacaksa aşağıdaki şekilde kullanılır:

```
Directory.CreateDirectory(@"..\Programlama");
```

Bu şekilde tanımlanabilir. Bir üst klasörü daha ekleme yapılacaksa aynı şekilde ..\ ifadeleri artırtılıp yapılabilir.

Dosya Silme

Dosya yolu belirtilen klasörü içindekileri ile beraber komple siler. Kullanımı:

```
Directory.Delete(Dosya konumu, bool);
```

Örnek:

Aşağıdaki örnekte dosya açılmak aynı dosya onay isteği ile birlikte silinmektedir.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO; //Dosya kütüphanesi eklendi

namespace try_catch_kullanimi
{
    class Program
    {

        static void Main(string[] args)
        {
            string x;
            Console.WriteLine("C sürücüsünün altında Programlama klasörü açıldı.");
            Directory.CreateDirectory(@"C:\Programlama");
            Console.WriteLine("Klasör silinsin mi?");
            x = Console.ReadLine();

            if (x=="sil")
            {
                string konum = @"C:\programlama";
                Directory.Delete(konum, true); //Klasör içindekileri ile birlikte silindi
                Console.WriteLine("Klasör silindi");
            }

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

FileStream ile Dosya Oluşturma

FileStream sınıfı ile yeni bir dosya açabileceğim gibi, var olan bir dosya üzerinde yazma işlemi gerçekleştirilebilir. Kullanımı:

```
FileStream dosya = File.Create(dosya_yolu);
```

veya

```
FileStream dosya = new FileStream(dosya_yolu, FileMode.OpenOrCreate, FileAccess.Write);
```

FileMode.OpenOrCreate

Dosya varsa açılacağını eğer dosya yoksa oluşturulacağını belirtir.

FileAccess.Write

Dosyaya veri yazılabileceğini belirtir.

Dosyaya Veri Yazma ve Güncelleme

Dosyaya veri yazmak için aşağıdaki StreamWriter nesne kullanılır.

```
StreamWriter yaz = new StreamWriter(dosya);
```

Örnek:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace dosya_islemleri
{
    class Program
    {
        public static void sil(String silinecek)
        {
            Directory.CreateDirectory(@"C:\Programlama");
            // Programlam isimli bir klasör açıldı.

            string belge_yolu = @"C:\Programlama\belge.txt";
            FileStream dosya = File.Create(belge_yolu);
            //belge.txt dosyasyı açıldı.

            StreamWriter yaz = new StreamWriter(dosya);

            yaz.WriteLine("C# Programlama dili Asp.net web programlama dili");// 1. satır
            yaz.Write("Java Programlama ");//2. Satırda yazdı
            //Dosyaya yazma işlemi yapıldı.

            yaz.Close();//dosya yazma işlemini kapat
            dosya.Close();//dosyayı kapat

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

Eğer aynı dosya üzerinde güncelleme yapılacaksa, aşağıdaki gibi flush() fonksiyonu ile birlikte kullanılması gereklidir.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace dosya_islemleri
{
    class Program
    {
        static void Main(string[] args)
        {

            Directory.CreateDirectory(@"C:\Programlama");
            // Programlama isimli bir klasör açıldı.

            string belge_yolu = @"C:\Programlama\belge.txt";
            FileStream dosya = new FileStream(belge_yolu, FileMode.OpenOrCreate, FileAccess.Write);
            //belge.txt dosyasyı açıldı.

            StreamWriter yaz = new StreamWriter(dosya);

            yaz.WriteLine("C# Programlama dili Asp.net web programlama dili");// 1. satır
            yaz.WriteLine("Java Programlama ");//2. Satırda yazdı
            //Dosyaya yazma işlemi yapıldı.

            yaz.Flush();
            //Dosyadaki bilgilerin boşaltılmasını ve güncellenmesini sağladık.

            yaz.Close(); //dosya yazma işlemini kapat
            dosya.Close(); //dosyayı kapat

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

Not: Dosyaya yazılan veriler, dosyaya hemen yazılmaz. Dosya yazımı belirli bir miktar veri yazılana kadar dosyayı güncellemez. FileStream sınıfının Flush() metodunu kullanarak istediğimiz anda dosyayı boşaltıp güncelleyebiliriz.

Dosyadan Veri Okuma

Dosyadan veri okuyabilmek için StreamReader sınıfı kullanılır. Örnek Kullanım:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO; //system kütüphane dosyası eklendi

namespace dosya_islemleri
```

```

{
    class Program
    {
        static void Main(string[] args)
        {

            string dosya_yolu = @"C:\Programlama\belge.txt";
            //belge.txt yolu tanımlandı

            StreamReader oku = new StreamReader(dosya_yolu);
            //belge.txt dosyasına yaz

            string bas= oku.ReadLine(); //baştan veriler okunmaya başlandı
            string son = oku.ReadToEnd(); //baştan sona veriler okunmaya başlandı.
            Console.WriteLine(bas+son); //Veriler ekrana yazdırıldı.

            oku.Close();

            Console.ReadKey();
        }
    }
}

```

Yukarıdaki okuma işlemini int ReadByte() metodunu kullanarak yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace dosya_islemleri
{
    class Program
    {
        static void Main(string[] args)
        {

            string adres = @"C:\Programlama\belge.txt";
            //belge.txt yolu tanımlandı

            FileStream dosya = new FileStream(adres, FileMode.Open);
            int x;

            while ((x = dosya.ReadByte()) != -1)
                //icerik boş ise -1 olacaktır. -1'e eşit değilse işlem yap.

                Console.Write((char)x); //aranan içeriği char tipinde çağırıp göster

            Console.ReadKey();
        }
    }
}

```

Dosyada Veri Arama

Dosyayı içerisinde veri aramak içinde StreamReader sınıfı kullanılır. Örnek Kullanım:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO; //system kütüphane dosyası eklendi

namespace dosya_islemeleri
{
    class Program
    {
        static void Main(string[] args)
        {

            string dosya_yolu = @"C:\Programlama\belge.txt", okunan_veri, kelime;
            //belge.txt dosyasına yaz

            StreamReader oku = new StreamReader(dosya_yolu);
            //belge.txt dosyasına yaz
            okunan_veri = oku.ReadLine();

            Console.WriteLine("Aranmak istenen kelime= ");
            kelime = Console.ReadLine();

            int bul = okunan_veri.IndexOf(kelime);
            /*Belirtilen ifadenin dosyanın içinde olup olmadığını kontrol ettik.
            Belirtilen değer var ise geriye 1 numarasını döndürür,
            yok ise -1 değerini döndürür. */

            if (bul != -1) // -1 eşit değilse
                Console.WriteLine("Aranan veri Bulundu.");
            else
                Console.WriteLine("Aranan veri bulunamadı.");
            oku.Close();

            Console.ReadKey();
        }
    }
}

```

Console Ekranında Renklendirme İşlemleri

ForegroundColor ile Yazıyı Renklendirme

Yazıyı istenilen renkte yapmak için kullanılacak kod satırı:

```
Console.ForegroundColor = ConsoleColor.Yellow;
```

BackgroundColor ile Arka Plan Renklendirme

Yazının arka plan rengini istenilen renk ile boyamak için kullanılacak kod satırı:

```
Console.BackgroundColor = ConsoleColor.Green;
```

Örnek:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```

using System.Threading.Tasks;
using System.IO;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {

            string yazi = "C# programlama";

            Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
            Console.ForegroundColor = ConsoleColor.White;//yazı rengi beyaz
            Console.WriteLine(yazi);

            Console.ForegroundColor = ConsoleColor.Blue;//yazı rengi beyaz
            Console.WriteLine(yazi);//Yukarıda arka plan kırmızı tanımı burada da geçerli

            Console.ResetColor(); //renklendirme sıfırlandı
            Console.WriteLine(yazi);

            Console.ReadLine();
        }
    }
}

```

Veri Tabanı İşlemleri

Veritabanı demek en basit ifade ile bilgi depoların düzenli bir şekilde ayrılp tutulduğu alanlardır. Dosya işlemleri konusunda ufakta olsa bir veritabanını olarak metin dosyasına veri eklemenin nasıl yapıldığını önceki konumuzda anlatmıştık. Bir verinin, veri tabanına kaydedildirken konum belirterek kaydetme, verileri okurken aynı şekilde konumunu belirterek okumak yada verileri silerken belirtilen adresdeki konumunda bulunan verileri silerek nasıl kolay bir şekilde yapıldığını veritabanı sistemlerini kullanarak anlatmaya çalışacağız.

Bir veritabanı sistemini hazırlamak için çeşitli veritabanı programları vardır. Programlama yazılımı veritabanlarının hemen hemen tamamıyla iletişim kurabilmektedir. Şimdi C# kullanabileceğimiz bazı veritabanı sistemlerinden ve kullanılan bağlantı parametrelerinden bahsedeceğiz.

- Microsoft Access
- MSSQL
- Microsoft Sql Server veritabanında çok hızlı bir veritabanı sistemidir. Genelde c#, asp.net ile hazırlanmış yazılımların, veritabanı sistemlerinde kullanılır.
- Oracle database (En güçlü Veri Tabanı Sistemi)
- Oracle genelde bankacılık sistemlerinde kullanılan bir veri tabanı sistemidir. Çok güçlü hızlı bir veritabanıdır.
- IBM DB/2
- Adaptive Server Enterprise
- Informix
- Microsoft Visual FoxPro
- MySQL

- Genelde Php gibi web programlama dillerinde kullanılan bir veritabanı sistemidir.
- PostgreSQL
- Progress
- SQLite

Android işletim sistemlerinde özellikle telefonlarda kullanılan, yazılımın kurulum yaptığı andan itibaren devreye giren bir veritabanı sistemidir.
- Teradata

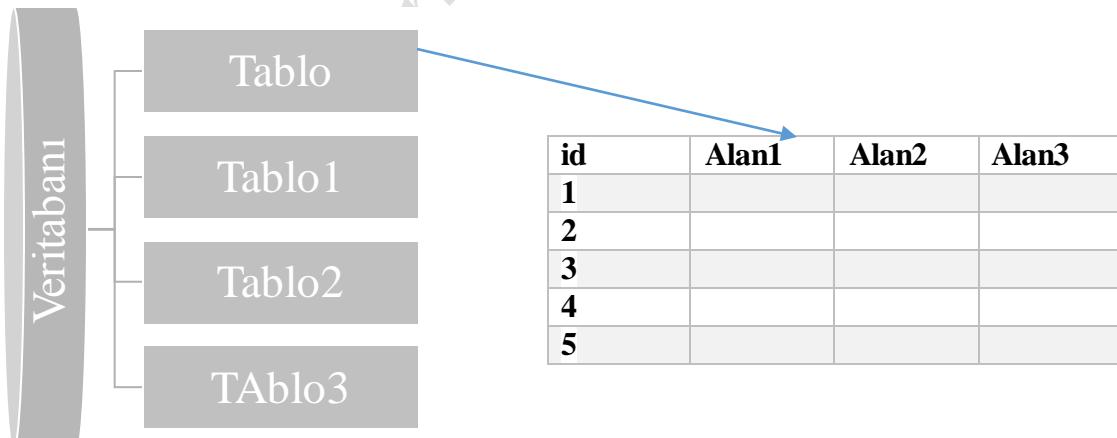
Bu kitabımızda C#’da en çok kullanılan veritabanı sistemi olarak **Access ve Mssql** veritabanı sistemlerini kullanacağız. Genel olarak yapının nasıl olduğunu anlatabilmek için Access veritabanını kullanacağız. Mssql ve Access veritabanında kullanılan erişim metodları:

- MsSql veritabanına direk erişim sağlanabilir
- Access veritabanına erişim için OleDb erişim protokolü ile erişim sağlanır.

Kullanacığımız yapıya bağlı olarak hız ve güvenirlik söz konusu ise MsSql veritabanını kullanmak daha mantıklı bir yol olacaktır. Access veritabanı MsSql veritabanına göre daha yavaştır. Yaptığımız yazılım eğer küçük bir kesime ve bilgi deposuna sahipse MsSql veritabanını yerine Access veritabanı kullanılabilir.

Veritabanı sistemleri çoğunlukla bir metin depolama yeridir. Ve zaten olması gerekende budur. Bir resmi yada bir videoyu veritabanına direkt kaydetmek mümkün değildir. Yalnız unutulma masası gereken bir nokta var veritabanı ne kadar az bir boyuta sahipse kadar verimli ve hızlı çalışır. Metin haricindeki dosyalarımızı veritabanına kaydetmeyeceğiz. Bunların bulunduğu konumun adreslerini veritabanına kaydetmemiz gereklidir.

Veritabanının yapısını şekil üzerinden gösterelim:



Bir veritabanı tablolardan oluşur. Bu tablolarda kendi içerisinde veri girişi yapılacak tablo alanlarından meydana gelir. Yukarıdaki şekilde baktığımızda 4 farklı tablo açıldı. İlk tabloda verilerin kaydedileceği ayrıca tablo alanları açıldı. Veritabanı mantığına baktığımızda aslında verilerin kaydedildiği yer tablolardır. Her tablo birbirinden bağımsız veri içerikleri ile doludur. Ve bu tablolara bağlı veri alan tabloları mevcut olmaktadır. Veriler bu şekilde numarası ve hangi alanda olduğu kolaylıkla bulunmuş ve kaydedilmiş olur.

Bir Access veritabanında tablo ve alanların oluşumunu aşağıdaki şekilde gösterelim.

The screenshot shows the Microsoft Access ribbon with the 'TABLO' tab selected. In the 'ALANLAR' (Fields) section, there are buttons for 'Görünüm' (View), 'Kısa Metin' (Short Text), 'Sayı' (Number), 'Para Birimi' (Currency), 'Tarih ve Saat' (Date/Time), and 'Evet/Hayır' (Yes/No). Below these are buttons for 'Sil' (Delete), 'Ad ve Resim Yazısı' (Text with Picture), 'Varsayılan Değer' (Default Value), and 'Alan Boyutu' (Field Size). On the right, there are buttons for 'Aramaları' (Search), 'İfadeyi Değiştir' (Change Placeholder), and 'Değiştir Aya' (Change Year). The 'Özellikler' (Properties) button is also visible. The main area displays a table named 'Tablo1' with columns 'Kimlik', 'Numara', 'AdSoyad', and 'Sınıf'. The data rows are: 1 34 Mehmet Akif Arvas 12/C, 2 48 Ahmed Arvas 11/A, 3 78 Fatima Sultan 10/B. A sidebar on the left shows 'Tablolar' (Tables) with 'Tablo1', 'Tablo2' (selected), and 'Tablo3'.

Access veritabanında 3 farklı tablo hazırladık. Ve her bir tablonun içerisinde ayrıca tablo alanları açtık. Tablo1'e ait alanlar açılıp veri giriş parametreleri kategorize edildi.

Not: Tablo adı ve alanları boşluk içermemesi, ve kullanılacak isimlerin sadece İngilizce harflerden oluşmuş bir ad olması gereklidir.

Veritabanımız açtıktı. Tablo ve alanlarımızı da oluşturduk. Sıra geldi bu veritabanına nasıl kaydetme, okuma, silme gibi işlemlerin yapılacağına. Veritabanına bu işlemleri yapabileceğimiz bir programlama dili var buna SQL programlama dili denir. Şimdi Sql programlama dili ile sırası nasıl yapıldığını gösterelim.

SQL Programlama Dili

Structured Query Language (SQL) yapısal sorgulama dilidir. Veritabanları üzerinde, karmaşık sorgular yapmak için tasarlanmış bir programlama dildir.

SQL Komutları

1. Select
Veritabanından kayıt çekmek için kullanılır.
2. Insert
Veritabanına kayıt eklemek için kullanılır.
3. Update
Veritabanında güncelleştirme yapmak için kullanılır.
4. Delete
Veritabanında silme işlemini gerçekleştirmek için kullanılır.

Select Komutu

- SELECT
- ORDER BY
- ASC ve DESC
- WHERE
- AND ve OR

Bir tablodan veri çekmek için kullanılır. Tablo adını yazarken birebir aynı olması gerekir. Kullanımı:

```
SELECT * FROM Tablo1
```

Tablo1'deki bütün veriler çekildi. Eğer tablonun içerisinde sadece bellli alanların çekilmesi isteniyorsa aşağıdaki gibi kullanılabilir;

```
SELECT Alan1, Alan2 FROM Tablo1
```

Tablo1'deki sadece Alan1 ve Alan2'de olan veriler çekildi. Verileri çekerken belirli sıralı düzen içerisinde listelenmesi isteniyorsa aşağıdaki gibi kullanılabilir;

```
SELECT *FROM Tablo1 ORDER BY Alan1
```

Tablo1'deki bütün veriler listelenirken Alan1'de olan verilere göre sıralama yap anlamına geliyor. Sorgumuzu aşağıdaki şekilde de kullanmak mümkün;

```
SELECT *FROM Tablo1 ORDER BY Alan1 ASC
```

Eğer sıralama yaparken en son eklenen verilere göre tersten listelemeye yapmak isteniyorsa aşağıdaki gibi kullanılabilir;

```
SELECT *FROM Tablo1 ORDER BY Alan1 DESC
```

Tablo1'deki bütün verileri listelerken Alan1'de olan verilere göre tersten sıralama yap demek anlamına geliyor. Bir tablonu verileri birden fazla kriterde göre listelenebilir. Örneğin;

```
SELECT *FROM Tablo1 ORDER BY Alan1 ASC, Alan2 DESC
```

Tablo1 listelenmesi yapılrken Alan1'i normal sıralama, Alan2'de tersten sıralama yapacak şekilde düzenleyip listeler.

Tabloda istenilen bir koşula bağlı olarak arama yapıp listeleyip yapmak istediğimizde Where komutunun nasıl kullanıldığını gösterelim;

```
SELECT * FROM Tablo1 Where Alan1 = 'aranmak istenen ifade'
```

Not: Aranan değer metin ise tırnak içerisinde, sayısal bir değer ise tırnak içerisinde yazmaya gerek yoktur.

Tablomuzun içerisinde birden fazla koşula bağlı alanda arama yapmak istediğimizde AND komutu aşağıdaki gibi kullanılabilir;

```
SELECT * FROM Tablo1 Where Alan1 = 'metin' AND Alan2 = sayı
```

Tablodaki iki koşuldan birisinin sağlanması durumunda arama yapmak istediğimizde AND yerine OR aşağıdaki gibi kullanılabilir;

```
SELECT * FROM Tablo1 Where Alan1 = 'metin' OR Alan2 = sayı
```

Tabloda koşulların farklı olması durumunda yani hem AND komutu, hemde OR komutu kullanılmak istendiğinde aşağıdaki gibi yazılabılır;

```
SELECT * FROM Tablo1 Where (Alan1 = 'metin' OR Alan2 = sayı) AND Alan3 = 'metin'
```

Evet sıra geldi yukarıdaki anlattığımız komutları programımız içerisinde nasıl ve nerde kullanıldığına. Yukarıdaki örneklerimiz bir Access veri tabanı açıp uygulamalı olarak gösterelim.

Veritabanımızı açtıktan sonra aşağıdaki gibi tablo ve alan isimlerini düzenleyelim.

Kimlik	AdSoyad	Numara
1	Mehmet Akif Arvas	34
2	Fatima Sultan	120
3	Ahmed Işık	65
*	Yeni	0

Veritabanı Adı: datveri.mdb

Tablo Adı: Tablo1

Tablo Alan İsimleri: Kimlik, AdSoyad (Veri Türü= Kısa Metin) ve Numara (Veri Türü= Sayı)

Veritabanımız açık tablo ve alan isimlerinde verdikten sonra veritabanımızı program dosyaların bulunduğu **debug** klasörün içerişine kopyalayalım. Daha sonra veritabanımıza bağlanıp tablomuzun içindekilerini listeleyelim.

Örnek:

Veritabanımızdaki AdSoyad ve Numara bölgelerindeki kayıtları listeleyelim.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        static void Main(string[] args)
        {

string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb; Persist Security Info=False;";

/*Access veritabanı bağlantı cümlesi. Tek bir satırda yazılır. Access veritabanı sürümü 2003'den sonraki sürümlerde veritabanının uzantısı .accdb olarak yazılmaktadır. Buna bağlı olarak bağlantı cümleseide değişmektedir. Fakat biz veritabanı sürümü olarak 2003 kullandık. Yeni sürümlerde Veritabanına dışardan veritabanına kayıt işlemini yapabilmek için güvenlik ayarlarından izin düzenlenmesi gereklidir. Aksi takdirde her ne kadar bağlantı cümlesi doğru yazılısa da veritabanına bağlanamayacaktır. */

string liste = "SELECT * FROM Tablo1";//Listeleme için komut oluşturduk

OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
```

```

//Veritabanına Bağlandık.
baglanti.Open();

OleDbCommand komut = new OleDbCommand(listele,baglanti);
//Veritabanındaki verileri listeledik

OleDbDataReader okud;
okud = komut.ExecuteReader();

while (okud.Read())
{
Console.WriteLine(okud["AdSoyad"].ToString()+" "+okud["Numara"].ToString());
}
baglanti.Close();
Console.ReadKey();
}
}
}

```



Program çalıştırıldığında veritabanımızda alan isimleri AdSoyad ve Numara bölmelerindeki kayıtlar yandaki gibi listelendi.

Örnek:

Veritabanımızdaki verileri sıralı bir düzen içerisinde listeleyelim.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        static void Main(string[] args)
        {

string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb;
Persist Security Info=False;";//Access veritabanı bağlantı cümlesi

string listele = "SELECT * FROM Tablo1 Order By AdSoyad";
// Sıralı Listeleme için komut oluşturduk

OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
//Veritabanına Bağlandık.
baglanti.Open();

OleDbCommand komut = new OleDbCommand(listele,baglanti);
//Veritabanındaki verileri listeledik

```

```

        OleDbDataReader okud;
        okud = komut.ExecuteReader();

        while (okud.Read())
        {
Console.WriteLine(okud["AdSoyad"].ToString() + " " + okud["Numara"].ToString());
        }
        baglanti.Close();
        Console.ReadKey();
    }
}
}

```

Veritabanındaki kayıtlar sıralı bir şekilde listelendi.

Örnek:

Veritabanımızdaki verileri ters sıralı bir düzen içerisinde listeleyelim.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        static void Main(string[] args)
        {

string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb;
Persist Security Info=False;";//Access veritabanı bağlantı cümlesi

string liste = "SELECT * FROM Tablo1 Order By AdSoyad Desc";
// Ters Sıralı Listeleme için komut oluşturduk

OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
//Veritabanına Bağlandık.
baglanti.Open();

OleDbCommand komut = new OleDbCommand(liste,baglanti);
//Veritabanındaki verileri listededik

        OleDbDataReader okud;
        okud = komut.ExecuteReader();
    }
}
}

```

```

        while (okud.Read())
    {
Console.WriteLine(okud["AdSoyad"].ToString()+" "+okud["Numara"].ToString());
    }
    baglanti.Close();
    Console.ReadKey();
}
}
}

```

Listelemeyi ters olacak şekilde listeledik.

Örnek:

Veritabanında ismi Mehmet Akif Arvas olan kişilere göre arama işlemini yapalım.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        static void Main(string[] args)
        {

            string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb;
Persist Security Info=False;";//Access veritabanı bağlantı cümlesi

string liste = "SELECT * FROM Tablo1 Where AdSoyad='Mehmet Akif Arvas'";
// AdSoyad Alanı içinde arama yapıldı.

OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
//Veritabanına Bağlandık.
            baglanti.Open();

OleDbCommand komut = new OleDbCommand(liste,baglanti);
//Veritabanındaki verileri listeledik

            OleDbDataReader okud;
            okud = komut.ExecuteReader();

            while (okud.Read())
            {
Console.WriteLine(okud["AdSoyad"].ToString()+" "+okud["Numara"].ToString());
            }
            baglanti.Close();
            Console.ReadKey();
        }
    }
}

```

Program çalıştırıldığında ekranda sadece Mehmet Akif Arvas yazılacaktır. Where arama ile istenilen değer eğer veritabanında varsa listelenir yoksa listelenmez. Arama yapılacak ifade birebir aynı olması gereklidir. Dosya işlemleri konusunda arama yapılan ifade birebir olmasa bile

listeleme yapılabiliyordu bu durumda istenilen sonucu bulmakta veya listelemekte problem teşkil ediyordu. Sql komutları ile bu problem ortadan kaldırılmış olmaktadır.

INSERT Komutu

Veritabanına kayıt eklemek için kullanılan sql komutudur. Kullanımı;

INSERT INTO Tablo Adı(Tablo Alanları) values (Tablo alanlarına eklenecek veriler)

Örnek:

Veritabanı Adı: datveri.mdb

Tablo Adı: Tablo1

Tablo Alan İsimleri: Kimlik, AdSoyad (Veri Türü= Kısa Metin) ve Numara (Veri Türü= Sayı)

Veritabanımızı açtık. Tablo ve alan isimlerinde verdikten sonra veritabanın program dosyalarının bulunduğu **debug** klasörün içerisine kopyalayalım. Daha sonra veritabanımıza bağlanıp tablomuza isim ve numara ekleyelim.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        static void Main(string[] args)
        {

string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb; Persist Security Info=False;";//Access veritabanı bağlantı cümlesi

        OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
        //Veritabanına Bağlandık.
        baglanti.Open();

        string kayit = "INSERT INTO Tablo1(AdSoyad, Numara) Values('Ali Yıldırım', 458)";
        // Ali Yıldırım 458 numaralı kayıt cümlesi oluşturuldu.

        OleDbCommand komut_kayit = new OleDbCommand();
        komut_kayit.Connection = baglanti;
        komut_kayit.CommandText = kayit;//veritabanına kaydetme işlemi yapıldı.
        komut_kayit.ExecuteNonQuery();
        baglanti.Close();

        Console.ReadKey();

    }
}
```

Veritabanına Ali Yıldırım isimli ve 458 numarada kayıt yapıldı. Eğer aynı zamanda listeleme yapacaksak aşağıdaki gibi kullanılabilir;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        static void Main(string[] args)
        {

string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb;
Persist Security Info=False;"//Access veritabanı bağlantı cümlesi

            string liste = "SELECT * FROM Tablo1";
            // Sıralı Listeleme için komut oluşturduk

            OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
            //Veritabanına Bağlandık.
            baglanti.Open();

            string kayit = "INSERT INTO Tablo1(AdSoyad, Numara) Values('Ali Yıldırım', 458) ";
            // Ali Yıldırım 458 numaralı kayıt cümlesi oluşturuldu.

            OleDbCommand komut_kayit = new OleDbCommand();
            komut_kayit.Connection = baglanti;
            komut_kayit.CommandText = kayit;
            komut_kayit.ExecuteNonQuery();
            baglanti.Close();

            baglanti.Open();

            OleDbCommand komut = new OleDbCommand(liste, baglanti);
            //Veritabanındaki verileri listeledik

            OleDbDataReader okud;
            okud = komut.ExecuteReader();

            while (okud.Read())
            {
                Console.WriteLine(okud["AdSoyad"].ToString() + " " + okud["Numara"].ToString());
            }
            baglanti.Close();
            Console.ReadKey();
        }
    }
}
```

Veritabanına kayıt yaptıktı ve kaydedilen veriyi aynı zamanda listeledik. Veritabanına kayıt yaparken bağlantıyı açtık. Kayıt yaptıktan sonra veritabanına bağlantıyı kestik. Sonra tekrar listeleme yapmak için veritabanına bağlantıyı açtık. En son veritabanına bağlantı kapatılarak işlem tamamlandı.

Veritabanımıza verileri eklerken klavyeden istenen bilgilere göre kayıt yapmak istiyorsak aşağıdaki gibi kullanılabilir;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        static void Main(string[] args)
        {

            string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb;
Persist Security Info=False;";//Access veritabanı bağlantı cümlesi

            string liste = "SELECT * FROM Tablo1";
            // Sıralı Listeleme için komut oluşturduk

            OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
            //Veritabanına Bağlandık.
            baglanti.Open();

            Console.WriteLine("Kayıt yapılacak Ad ve Soyad:");
            string isim = Console.ReadLine();
            Console.WriteLine("Kayıt yapılacak Numara:");
            string no = Console.ReadLine();

            string kayit = "INSERT INTO Tablo1(AdSoyad, Numara)
Values('"+isim.ToString()+"','"+no.ToString()+"')";
            //Tek satırda yazılması gereklidir. Klavyeden yazılan değerler, alanlara kaydedildi.

            OleDbCommand komut_kayit = new OleDbCommand();
            komut_kayit.Connection = baglanti;
            komut_kayit.CommandText = kayit;
            komut_kayit.ExecuteNonQuery();
            baglanti.Close();

            baglanti.Open();

            OleDbCommand komut = new OleDbCommand(liste, baglanti);
            //Veritabanındaki kayıtları listeledik

            OleDbDataReader okud;
            okud = komut.ExecuteReader();

            while (okud.Read())
            {
                Console.WriteLine(okud["AdSoyad"].ToString() + " " + okud["Numara"].ToString());
            }
            baglanti.Close();
            Console.ReadKey();

        }
    }
}
```

UPDATE Kullanımı

Veritabanındaki alanları güncelleştirmek için kullanılan komuttur. Kullanımı;

UPDATE Tablo SET Alan1 = 'Değiştirilecek ifade' Where Alan1='Değişmesi istenen ifade'

Örnek:

Veritabanı Adı: datveri.mdb

Tablo Adı: Tablo1

Tablo Alan İsimleri: Kimlik, AdSoyad (Veri Türü= Kısa Metin) ve Numara (Veri Türü= Sayı)

Veritabanımızı açtık. Tablo ve alan isimlerinde verdikten sonra veritabanını program dosyalarının bulunduğu **debug** klasörün içerisinde kopyalayalım. Daha sonra veritabanımıza bağlanıp tablomuzdaki değişmesi istenen ifadeyi klavyeden girilen veriye göre veritabanında arayıp yeni değer ile değiştirelim. Aynı zamanda veritabanındaki kayıtları listeleyelim.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        static void Main(string[] args)
        {

string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb;
Persist Security Info=False;";
//Access veritabanı bağlantı cümlesi

        string liste = "SELECT * FROM Tablo1";
        // Sıralı Listeleme için komut oluşturduk

        OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
        //Veritabanına Bağlandık.
        baglanti.Open();

        Console.WriteLine("Update yapılacak İsim:");
        string isim = Console.ReadLine();

        string guncel ="UPDATE Tablo1 SET AdSoyad = 'Fatih' Where AdSoyad='"+isim.ToString()+"'";
        // Klavyeden girilen veri eğer veritabanında varsa, yeni değer olarak fatih olarak
        //güncelleştir.

        OleDbCommand komut_kayit = new OleDbCommand();
        komut_kayit.Connection = baglanti;
        komut_kayit.CommandText = guncel;
        komut_kayit.ExecuteNonQuery();
        baglanti.Close();

        baglanti.Open();

        OleDbCommand komut = new OleDbCommand(listele, baglanti);
```

```

//Veritabanındaki verileri listeledik

OleDbDataReader okud;
okud = komut.ExecuteReader();

while (okud.Read())
{
    Console.WriteLine(okud["AdSoyad"].ToString() + " " + okud["Numara"].ToString());
}
baglanti.Close();
Console.ReadKey();

}
}
}

```

DELETE Komutu

Veritabanında bulunan tablolarda veya tablo alanlarındaki kayıtları silmek için kullanılır.
Delete komutunun kullanımı;

DELETE FROM Tablo1 WHERE Alan='Mehmet'

Tablo1'de Alan'ın içerisinde arama yapıp eğer aranan ifade varsa siler.

Örnek:

Veritabanımızda kayıtlı olan alanlarda klavyeden girilen veriye göre tablo alanlarımızda silme işlemini gerçekleştiren programımızı yapalım.

Veritabanı Adı: datveri.mdb

Tablo Adı: Tablo1

Tablo Alan İsimleri: Kimlik, AdSoyad (Veri Türü= Kısa Metin) ve Numara (Veri Türü= Sayı)

Veritabanımızı açtık. Tablo ve alan isimlerinde verdikten sonra veritabanını program dosyalarının bulunduğu **debug** klasörün içerisinde kopyalayalım. Daha sonra veritabanımıza bağlanıp tablomuzdaki değişmesi istenen ifadeyi klavyeden girilen veriye göre veritabanında arayıp silelim. Aynı zamanda veritabanındaki kayıtları listeleyelim.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        static void Main(string[] args)
        {

string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb; Persist Security Info=False;"//Access veritabanı bağlantı cümlesi

        string listele = "SELECT * FROM Tablo1";
    }
}

```

```
// Sıralı Listeleme için komut oluşturduk

OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
//Veritabanına Bağlandık.
baglanti.Open();

Console.WriteLine("Silinecek İsmi giriniz:");
string isim = Console.ReadLine();

string silinen = "DELETE FROM Tablo1 WHERE AdSoyad='"+isim.ToString()+"'";

OleDbCommand komut_sil = new OleDbCommand();
komut_sil.Connection = baglanti;
komut_sil.CommandText = silinen;
if (komut_sil.ExecuteNonQuery() == 1)
//silinen değer için 1 geri dönderir. silme işlemi gerçekleşmediyse 0 gönderir

Console.WriteLine("Silindi");
baglanti.Close();

baglanti.Open();

OleDbCommand komut = new OleDbCommand(listele, baglanti);
//Veritabanındaki verileri listeledik

OleDbDataReader okud;
okud = komut.ExecuteReader();

while (okud.Read())
{
Console.WriteLine(okud["AdSoyad"].ToString() + " " + okud["Numara"].ToString());
}
baglanti.Close();
Console.ReadKey();

}

}
```

Veritabanı Uygulama Örneği

Bir firmada bulunan ürünler veritabanımızdan çekip ekranımıza aynı zamanda alt menüler oluşturarak silmek, güncellemek, yeni ürün eklemek için seçim yapılacak liste ekranda oluşturulup seçilen ifadeye göre işlem yapan programımızı yapalım.

Veritabani Adı: datveri.mdb

Tablo Adı: Tablo1

Tablo Alan İsimleri: id, urunadi (Veri Türü= Kısa Metin) ve fiyat (Veri Türü= Sayı)

	id	urunadi	fiyat	Ek
*	1	Buzdolabı	1500	
	3	Çamaşır Makinesi	1200	
	6	Televizyon	1000	
	8	Bulaşık Makinesi	1850	
	9	Fırın	800	
		Yeni	0	

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        public static void metot()
        {
            try
            {

                bool dongu = true;//döngü için koşulla yapı tanımlandı
                string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=datveri.mdb; Persist Security Info=False;";//Access veritabanı bağlantı cümlesi
//veritabanı cümlesi tek satırda yazılması gereklidir. Burada sayfamıza sığmadığı için
alt satıra yazmak zorunda kaldık.

                string liste = "SELECT * FROM Tablo1";
                // Sıralı Listeleme için komut oluşturduk

                OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
                //Veritabanına Bağlandık.

                while (dongu)// dongu true olduğu için while çalışacaktır.
                {
                    Console.ResetColor();//Renkleri sıfırladık.
                    baglanti.Open();//Veritabanına bağlantıyı açtık.

                    OleDbCommand komut = new OleDbCommand(liste, baglanti);
                    //Veritabanındaki verileri listededik

                    OleDbDataReader okud;
                    okud = komut.ExecuteReader();
                    Console.WriteLine("id\tFiyat\t\türün Adı\n");
                    while (okud.Read())
                    {
Console.WriteLine(okud["id"].ToString()+"\t"+okud["fiyat"].ToString()+" TL "+ "\t\t\t"
+ okud["urunadi"].ToString());
                    }

                    baglanti.Close();

                    Console.WriteLine("\n*****\n");
                }
            }
        }
    }
}

```

```

Console.WriteLine("Yeni Ürün Ekle \t[0]");
Console.WriteLine("Ürün Sil \t[1]");
Console.WriteLine("Ürün Güncelle \t[2]");
Console.WriteLine("\n*****\n");

Console.Write("Seçiminiz:");
int secim = Convert.ToInt32(Console.ReadLine());

switch (secim)
{
    case 0:
    {
        baglanti.Open();

        Console.WriteLine("Eklenecek Ürün Adını Giriniz:");
        string ekleadi = Console.ReadLine();
        Console.WriteLine("Ürün Fiyatını Giriniz:");
        double eklefiyat = Convert.ToDouble(Console.ReadLine());

string kayit = "INSERT INTO Tablo1(urunadi, fiyat) Values('" + ekleadi.ToString() +
", '" + eklefiyat.ToString() + "') ";
//Tek satırda yazılması gereklidir. Sayfaya sıyrılmadığı için 2 satır oldu.

        OleDbCommand komut_ekle = new OleDbCommand();
        komut_ekle.Connection = baglanti;
        komut_ekle.CommandText = kayit;
        if (komut_ekle.ExecuteNonQuery() == 1)
//eklenen değer için 1 geri dönerdir. ekleme işlemi gerçekleşmediyse 0 gönderir
        {
            Console.Clear();
            Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
            Console.ForegroundColor = ConsoleColor.White;

            Console.WriteLine("Eklendi.");
        }
        else
        {
            Console.Clear();
            Console.BackgroundColor = ConsoleColor.Red; //arka plan kırmızı
            Console.ForegroundColor = ConsoleColor.White;

            Console.WriteLine("Eklenemedi!");
        }
        baglanti.Close();
    }
    break;
    case 1:
    {
        baglanti.Open();

        Console.WriteLine("Silinecek Ürün Adını Giriniz:");
        string isim = Console.ReadLine();

string silinen = "DELETE FROM Tablo1 WHERE urunadi='" + isim.ToString() + "'";

        OleDbCommand komut_sil = new OleDbCommand();
        komut_sil.Connection = baglanti;
    }
}

```

```

        komut_sil.CommandText = silinen;

        if (komut_sil.ExecuteNonQuery() == 1)
//silinen değer için 1 geri dönderir. silme işlemi gerçekleşmediyse 0 gönderir
        {
            Console.Clear();
            Console.BackgroundColor = ConsoleColor.Red;
//arka plan kırmızı
            Console.ForegroundColor = ConsoleColor.White;

            Console.WriteLine("Silindi");
        }

        else
        {
            Console.Clear();
            Console.BackgroundColor = ConsoleColor.Red;
//arka plan kırmızı
            Console.ForegroundColor = ConsoleColor.White;

            Console.WriteLine("Bulunamadı!");
        }
        baglanti.Close();
    }
    break;
case 2:
{
    Console.WriteLine("Update yapılacak Ürün:");
    string isim = Console.ReadLine();
    Console.WriteLine("Update yapılacak Ürünün yeni ismi:");
    string isimyeni = Console.ReadLine();
    Console.WriteLine("Update yapılacak Ürünün yeni fiyatı:");
    double fiyatyeni = Convert.ToDouble(Console.ReadLine());

string guncel = "UPDATE Tablo1 SET urunadi = '" + isimyeni.ToString() + "', fiyat ='"
+ fiyatyeni.ToString() + "' Where urunadi=''" + isim.ToString() + "'';

        OleDbCommand komut_kayit = new OleDbCommand();
        komut_kayit.Connection = baglanti;
        komut_kayit.CommandText = guncel;
        if (komut_kayit.ExecuteNonQuery() == 1)
        {
            Console.Clear();
Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
            Console.ForegroundColor = ConsoleColor.White;

            Console.WriteLine("Güncellendi.");
        }
        else
        {
            Console.Clear();
Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
            Console.ForegroundColor = ConsoleColor.White;

            Console.WriteLine("Bulunamadı!");
        }
        baglanti.Close();
    }
    break;
default:

```

```

        Console.Clear();
        Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
        Console.ForegroundColor = ConsoleColor.White;

        Console.WriteLine("Seçim Yapmadınız.");
        break;
    }

}

catch (FormatException hata)//sayı yerine başka bir şey girilirse burası çalışacak
{
    Console.Clear();
    Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
    Console.ForegroundColor = ConsoleColor.White;
    Console.WriteLine("Sayı girmelisiniz!" + hata.Message);

}
catch (OverflowException hata1)
//int'in kapasitesinin üzerine çıkarsa burası çalışacak
{
    Console.Clear();
    Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
    Console.ForegroundColor = ConsoleColor.White;
    Console.WriteLine("Giriş Sayısı çok büyük!" + hata1.Message);
}

finally
{
    metot();
}

}

static void Main(string[] args)
{
    try
    {
        metot(); //metot çağrııldı.
    }
    catch(FormatException hata)
    {
        Console.Clear();
        Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
        Console.ForegroundColor = ConsoleColor.White;
        Console.WriteLine("Sayı girmelisiniz!" + hata.Message);

    }
    catch (OverflowException hata1)//int'in kapasitesinin üzerine çıkarsa burası çalışacak
    {
        Console.Clear();
        Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
        Console.ForegroundColor = ConsoleColor.White;
        Console.WriteLine("Giriş Sayısı çok büyük!" + hata1.Message);
    }
    finally
    {

        metot();
    }
}

```

```
        }  
    }  
}
```

Program çalışmasının ekran çıktıları;

The screenshot shows a Windows command-line window with a yellow title bar. The title bar text is "file:///C:/Users/mehmet/Desktop/prg/veritabani/veritabani/bin/Debug/veritab...". The window contains a table of products:

id	Fiyat	Ürün Adı
1	1500TL	Buzdolabı
3	1200TL	Çamaşır Makinesi
8	1850TL	Bulaşık Makinesi
18	150TL	Sehpa

Below the table, there are several menu options:

Yeni Ürün Ekle [0]
Ürün Sil [1]
Ürün Güncelle [2]

Seçiminiz:0
Eklenecek Ürün Adını Giriniz:
TU
Ürün Fiyatını Giriniz:
1890

The screenshot shows a Windows command-line window with a yellow title bar. The title bar text is "file:///C:/Users/mehmet/Desktop/prg/veritabani/veritabani/bin/Debug/veritab...". The window displays the message "Eklendi." (Added) in red text.

The table of products now includes the new entry:

id	Fiyat	Ürün Adı
1	1500TL	Buzdolabı
3	1200TL	Çamaşır Makinesi
8	1850TL	Bulaşık Makinesi
18	150TL	Sehpa
19	1890TL	TU

Below the table, there are several menu options:

Yeni Ürün Ekle [0]
Ürün Sil [1]
Ürün Güncelle [2]

Seçiminiz:

```
file:///C:/Users/mehmet/Desktop/prg/veritabani/veritabani/bin/Debug/veritab... □ X
Silindi
id      Fiyat          Ürün Adı
1       1500TL         Buzdolabı
3       1200TL         Çamaşır Makinesi
18      150TL          Sehpası
19      1890TL         TV
*****
Yeni Ürün Ekle [0]
Ürün Sil   [1]
Ürün Güncelle [2]
*****
Seçiminiz:1
Silinecek Ürün Adını Giriniz:
Sehpası
```

```
file:///C:/Users/mehmet/Desktop/prg/veritabani/veritabani/bin/Debug/veritab... □ X
Silindi
id      Fiyat          Ürün Adı
1       1500TL         Buzdolabı
3       1200TL         Çamaşır Makinesi
19      1890TL         TV
*****
Yeni Ürün Ekle [0]
Ürün Sil   [1]
Ürün Güncelle [2]
*****
Seçiminiz:2
Update yapılacak Ürün:
TV
Update yapılacak Ürünün yeni ismi:
LCD Televizyon
Update yapılacak Ürünün yeni fiyatı:
2590
```

```
file:///C:/Users/mehmet/Desktop/prg/veritabani/veritabani/bin/Debug/veritab... □ X
Güncellendi.
id      Fiyat          Ürün Adı
1       1500TL         Buzdolabı
3       1200TL         Çamaşır Makinesi
19      2590TL         LCD Televizyon
*****
Yeni Ürün Ekle [0]
Ürün Sil   [1]
Ürün Güncelle [2]
*****
Seçiminiz:
```

Ürünlerimizin ismi uzun olursa peki ne yapacağız? Diğer bir problem ürünün adını birebir aynen girmediyim zaman ne olacak? İşte bu gibi durumların önüne geçmek için veritabanımızda tabloda ürün kodu ekleyip bu tür sorunların önündede geçmiş olacağız. Veritabanımıza Ürün Kodu Alanını ekleyip, yukarıdaki; Ekleme, Silme, Güncelleme, işlemlerini eklenen ürün koduna göre yapalım.

Not: Sayılarla göre yapılan işlemlerde ifadeler tek tırnak sonra çift tırnak yerine sadece çift tırnak bırakmak yeterli olacaktır.

Tablo1				
id	urunadi	fiyat	urunkodu	
1	Buzdolabı	1.500,00 ₺	123456	
3	Çamaşır Makinesi	1.200,00 ₺	789456	
19	LCD Televizyon	2.590,00 ₺	456123	
*	Yeni	0,00 ₺	0	

Program kodlarımız;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;//Access veritabanına bağlantı için kütüphane dosyası ekledik.

namespace veritabani
{
    class Program
    {
        public static void metot()
        {
            try
            {

                bool dongu = true;
                string baglanti_yolu = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=datveri.mdb;
                Persist Security Info=False;";//Access veritabanı bağlantısı cümlesi

                string liste = "SELECT * FROM Tablo1";
                // Sıralı Listeleme için komut oluşturduk

                OleDbConnection baglanti = new OleDbConnection(baglanti_yolu);
                //Veritabanına Bağlandık.
                while (dongu)
                {
                    Console.ResetColor();
                    baglanti.Open();

                    OleDbCommand komut = new OleDbCommand(liste, baglanti);
                    //Veritabanındaki verileri listeledik

                    OleDbDataReader okud;
                    okud = komut.ExecuteReader();
                    Console.WriteLine("Ürün Kodu\tFiyat\t\tÜrün Adı\n");
                    while (okud.Read())
                    {

```

```

Console.WriteLine(okud["urunkodu"].ToString()+"\t\t"+okud["fiyat"].ToString()+" TL"
+" \t\t\t " + okud["urunadi"].ToString());
        }
        baglanti.Close();
        Console.WriteLine("\n*****\n");
        Console.WriteLine("Yeni Ürün Ekle \t[0]");
        Console.WriteLine("Ürün Sil \t[1]");
        Console.WriteLine("Ürün Güncelle \t[2]");
        Console.WriteLine("\n*****\n");
        Console.Write("Seçiminiz:");
        int secim = Convert.ToInt32(Console.ReadLine());

        switch (secim)
        {
            case 0:
            {
                baglanti.Open();

                Console.WriteLine("Eklenenek Ürün Adını Giriniz:");
                string ekleadi = Console.ReadLine();
                Console.WriteLine("Ürün Fiyatını Giriniz:");
                double eklefiyat = Convert.ToDouble(Console.ReadLine());
                Console.WriteLine("Ürün Kodunu Giriniz:");
                int kod = Convert.ToInt32(Console.ReadLine());

                string kayit = "INSERT INTO Tablo1(urunadi, fiyat, urunkodu) Values('" +
ekleadi.ToString() + "', " + eklefiyat + ", " + kod+ ")";
//Sayıların olduğu bölgümlerde sadece tek tırnak kullandık. Ayrıca .tostring()
//bırakmaya gerek yok.

                OleDbCommand komut_ekle = new OleDbCommand();
                komut_ekle.Connection = baglanti;
                komut_ekle.CommandText = kayit;
                if (komut_ekle.ExecuteNonQuery() == 1)
//eklenen değer için 1 geri dönderir. ekleme işlemi gerçekleşmediyse 0 gönderir
                {
                    Console.Clear();
                    Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
                    Console.ForegroundColor = ConsoleColor.White;

                    Console.WriteLine("\nEklendi.\n");
                }
                else
                {
                    Console.Clear();
                    Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
                    Console.ForegroundColor = ConsoleColor.White;

                    Console.WriteLine("\nEklenemedi!\n");
                }
                baglanti.Close();
            }
            break;
        case 1:
        {
            baglanti.Open();

            Console.WriteLine("Silinecek Ürün Kodunu Giriniz:");

```

```

        int kod = Convert.ToInt32(Console.ReadLine());

string silinen = "DELETE FROM Tablo1 WHERE urunkodu=" + kod.ToString() + "";

        OleDbCommand komut_sil = new OleDbCommand();
        komut_sil.Connection = baglanti;
        komut_sil.CommandText = silinen;
        if (komut_sil.ExecuteNonQuery() == 1)
//silinen değer için 1 geri dönderir. silme işlemi gerçekleşmediyse 0 gönderir
{
            Console.Clear();
            Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
            Console.ForegroundColor = ConsoleColor.White;

            Console.WriteLine("\nSilindi\n");
}

else
{
    Console.Clear();
    Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
    Console.ForegroundColor = ConsoleColor.White;

    Console.WriteLine("\nBulunamadı!\n");
}
baglanti.Close();
}
break;
case 2:
{
    baglanti.Open();
    Console.WriteLine("Update yapılacak Ürün Kodu:");
    int kod = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Update yapılacak Ürünün yeni ismi:");
    string isimyeni = Console.ReadLine();
    Console.WriteLine("Update yapılacak Ürünün yeni fiyatı:");
    double fiyatyeni = Convert.ToDouble(Console.ReadLine());

string guncel = "UPDATE Tablo1 SET urunadi = '" + isimyeni.ToString() + "', fiyat ='"
+ fiyatyeni.ToString() + "' WHERE urunkodu=" + kod + "";
// Klavyeden girilen veri eğer veritabanında varsa, yeni değer olarak fatih olarak
//güncelleştir.

        OleDbCommand komut_kayit = new OleDbCommand();
        komut_kayit.Connection = baglanti;
        komut_kayit.CommandText = guncel;
        if (komut_kayit.ExecuteNonQuery() == 1)
{
            Console.Clear();
            Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
            Console.ForegroundColor = ConsoleColor.White;

            Console.WriteLine("\nGüncellendi.\n");
}
else
{
    Console.Clear();
    Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
    Console.ForegroundColor = ConsoleColor.White;
}

```

```

                Console.WriteLine("\nBulunamadı!\n");
            }

            baglanti.Close();

        }
        break;
    default:
        Console.Clear();
        Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
        Console.ForegroundColor = ConsoleColor.White;

        Console.WriteLine("Seçim Yapmadınız.");
        break;
    }

}
catch (FormatException hata)
{
    Console.Clear();
    Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
    Console.ForegroundColor = ConsoleColor.White;
    Console.WriteLine("Sayı girmelisiniz!" + hata.Message);

}
catch (OverflowException hata1)//int'in kapasitesinin üzerine çıkarsa burası çalışacak
{
    Console.Clear();
    Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
    Console.ForegroundColor = ConsoleColor.White;
    Console.WriteLine("Giriş Sayısı çok büyük!" + hata1.Message);
}
finally
{
    metot();
}

}

static void Main(string[] args)
{
    try
    {
        metot();
    }
    catch(FormatException hata)
    {
        Console.Clear();
        Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
        Console.ForegroundColor = ConsoleColor.White;
        Console.WriteLine("Sayı girmelisiniz!" + hata.Message);

    }
    catch (OverflowException hata1)//int'in kapasitesinin üzerine çıkarsa burası çalışacak
    {
        Console.Clear();
        Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
        Console.ForegroundColor = ConsoleColor.White;
        Console.WriteLine("Giriş Sayısı çok büyük!" + hata1.Message);
    }
}

```

```

        finally
        {
            metot();
        }
    }
}
}

```

Program ekran çıktısı;

Ürün Kodu	Fiyat	Ürün Adı
789456	1200TL	Çamaşır Makinesi
456123	2590TL	LCD Televizyon

Yeni Ürün Ekle [0]
Ürün Sil [1]
Ürün Güncelle [2]

Seçiminiz:0
Eklenecek Ürün Adını Giriniz:
Tv
Ürün Fiyatını Giriniz:
1854
Ürün Kodunu Giriniz:
4589958

Eklendi.

Ürün Kodu	Fiyat	Ürün Adı
789456	1200TL	Çamaşır Makinesi
456123	2590TL	LCD Televizyon
4589958	1854TL	Tv

Yeni Ürün Ekle [0]
Ürün Sil [1]
Ürün Güncelle [2]

Seçiminiz: