TEMEL VERİ TÜRLERİ VE TANIMLAMALARI

Ekrana veri yazdırma ve klavyeden veri girişi

C# dilinde kod yazarken, sonuçları ekrana yazdırma için **Console.WriteLine,** klavdeyen girilen verileri okuma için de **Console.ReadLine** fonksiyonlarını kullanırız.

Örnek: Ekrana *Merhaba OMÜ!* yazdıran bir C# programı oluşturalım.

```
Console.WriteLine("Merhaba OMÜ!");
Console.ReadKey();
```

NOT: Yazdığımız kodları çalıştırmak için Console.ReadKey() komutunu da programın sonuna eklememiz gerekir.

Veri türleri

Algoritmaları C# dilinde yazarken, verileri tanımlamak üzere değişkenler aracılığı ile programlar oluştururuz. Her verinin değer aralığına ve türüne göre, seçeceğimiz değişken tipi de farklılık göstermektedir.

Tablo 1: Değişken tipleri, alan ve değer aralıkları

Değişken tipi	Alan	Açıklama
byte	8 bit	[0, 255]
sbyte	8 bit	[-128, 127]
short	16 bit	[-32768, 32767]
ushort	16 bit	[0, 65535]
int	32 bit	[-2.147.483.648, 2.147.483.648,]
uint	32 bit	[0, 4.294.967.295]
long	64 bit	
ulong	64 bit	[0, 18.446.744.073.709.551.615]

double	64 bit	Artı Eksi 5x10 üzeri 324
decimal	128 bit	Artı Eksi 10 üzeri -28
bool		true false
char	16 bit	Tek karakter

Tablo 1'de C# üzerinde veri tanımlamak için kullanılan değişken tiplerine ilişkin özellikler gösterilmektedir. Genellikle algoritma tasarlarken, tamsayıları için **int**, sürekli sayılar için de **double** tiplerini kullanacağız. Mantıksal ifadeler için de **bool** tipinden yararlanacağız. Bu değişken tiplerinin yanı sıra sabit ifadeler için **const** tipini kullanıyoruz.

Örnek: Yaş, isim, boy ve karar şeklinde dört farklı değişken tanımlayalım.

```
int yas=32;
double boy = 1.76;
string isim="Emre";
bool karar=true;
Console.WriteLine(yas);
Console.WriteLine(boy);
Console.WriteLine(isim);
Console.WriteLine(karar);
Console.ReadKey();
```

Renk Değiştirme

Yazdığımız kodların içerisinde metnin ve yazının renklerini **Console.ForegroundColor** ve **Console.BackgroundColor** metotları ile değiştirebiliriz.

Console.ForegroundColor: Bir metnin yazı rengini belirler. *Console.ForegroundColor = ConsoleColor.*_____ şeklinde boşluk ifadesine istenilen renk tanımlandığında yazının rengi belirlenir.

Console.ForegroundColor = ConsoleColor.Yellow;	Metin sarı renkle gösterilir.
Console.ForegroundColor = ConsoleColor.Blue;	Metin mavi renkle gösterilir.

Console.BackgroundColor: Konsolun arka plan rengini belirler. *Console.BackgroundColor = ConsoleColor.*_____ şeklinde boşluk ifadesine istenilen renk tanımlandığında arka plan rengi belirlenir.

Console.ForegroundColor = ConsoleColor.Green;	Arka plan yeşil renkle gösterilir.
Console.ForegroundColor = ConsoleColor.Red;	Arka plan kırmızı renkle gösterilir.

Örnek: OMU ve Istatistik ifadelerini metnin yazı renklerini sırasıyla sarı ve mavi; arka plan renklerini de lacivert ve koyu kırmızı olacak şekilde tanımlayan bir program oluşturalım.

```
string unv = "OMU";
string bolum = "Istatistik";
Console.ForegroundColor = ConsoleColor.Yellow;
Console.BackgroundColor = ConsoleColor.DarkBlue;
Console.WriteLine(unv);
Console.ForegroundColor = ConsoleColor.Blue;
Console.BackgroundColor = ConsoleColor.DarkRed;
Console.WriteLine(bolum);
Console.ReadKey();
```

object ve dynamic türleri

object ve dynamic türü, genel veri türlerini temsil eder. Object türünde verilerin dönüşümleri kesin olarak tanımlanmalıdır. Ancak dynamic türü için tür tanımlama gerekli değildir. İşlemler esnasında değişken türü otomatik olarak belirlenir.

Örnek: object ve dynamic türlerine dair değişkenler tanımlayıp, veri türlerini gösteren programlar oluşturalım.

```
object yas = 56;
dynamic yol = 6.2;
Console.WriteLine(yas.GetType());
Console.WriteLine(yol.GetType());
Console.ReadKey();
```

```
object takim = "ABC";
string sampiyon = (string)takim;
object sayim = 525;
 int esassayim = (int)sayim;
 Console.WriteLine(sampiyon);
Console.WriteLine(esassayim);
Console.ReadKey();
```

NOT: GetType() metodu, bir değişkenin türünü tespit etmek için kullanılır.

Örnek: Bir adet int ve bir adet double şeklinde sayı tanımlarak değişken türünü gösteren bir program oluşturalım.

```
int para=500;
Console.WriteLine(para.GetType());
double ph=8.7;
Console.WriteLine(ph.GetType());
Console.ReadKey();
```

NOT: object türünde veri dönüşümü gerçekleştirmediğimiz takdirde, programlarımız hata verir. Ancak dynamic türünde, veri türü kendiliğinden belirlenir.

```
int sayi1 = 10;
double sayi2 = 30;
double toplam = sayi1 + sayi2;
dynamic toplam = sayi1 + sayi2;   /// Dynamic türünde, son türü kendi belirler.
Console.WriteLine(toplam);
Console.WriteLine(toplam.GetType());
Console.ReadKey();
```

```
object yas = 56;
dynamic yol = 6.2;
int sonyas = (int)yas + 1;     /// (int dönüşümü yapmazsak hata veriyor.
double sonyol = yol + 10;
Console.WriteLine(sonyas);
Console.WriteLine(sonyol);
Console.ReadKey();
```

Veri dönüşümleri

C# üzerinde veri dönüşümü yapılırken temel olarak iki yol izlenir. İlk yol; değişkenin önüne doğrudan türünü tanımlamaktır. Diğer yol ise, Convert.To___ şeklinde değişken türünü boşluk kısmına yazarak veri dönüştürmektir.

Tablo 2: Convert.To metodlarına dair veri dönüşümleri

Dönüşüm fonksiyonu	Açıklama
Convert.ToBoolean()	Boolean türüne dönüştürür.
Convert.ToByte()	Byte türüne dönüştürür.
Convert.ToInt16()	Int16 türüne dönüştürür.
Convert.ToInt32()	Int32 türüne dönüştürür.
Convert.ToInt64()	Int64 türüne dönüştürür.
Convert.ToString()	String türüne dönüştürür.
Convert.ToDouble()	String türüne dönüştürür.

Tablo 2'de C# üzerinde veri dönüşümü için kullanılan Convert.To metodlarına ait tanımlamalar gösterilmektedir.

Örnek: İki farklı sözce (string) biçiminde ve bir adet de object türünde değişken tanımlayarak veri dönüşümleri gerçekleştiren program oluşturalım.

```
string sayi1 = "45.5";
string sayi2 = "800";
object sayim = 525;
int esassayim = (int)sayim;
Console.WriteLine(Convert.ToDouble(sayi1));
Console.WriteLine(Convert.ToInt32(sayi2));
Console.ReadKey();
```