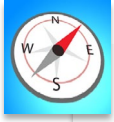


# DEĞİŞKENLER VE VERİ TİPLERİ



## İÇİNDEKİLER

- Değişkenler
  - Değişken Tanımlama Kuralları
  - Değişken İsimlendirme Standartları
  - Değişken Tanımlama
  - Değişkene Değer Atama
- Değişken Tipleri
  - Değer Tipleri
  - Referans Tipleri
  - Değişken Tiplerini Öğrenmek
- Değişkenlerin Geçerlilik Bölgesi
- Tip Dönüşümleri (Casting)
  - Örtülü/Otmatik Dönüşüm
  - Açık/Manuel Dönüşüm



## HEDEFLER

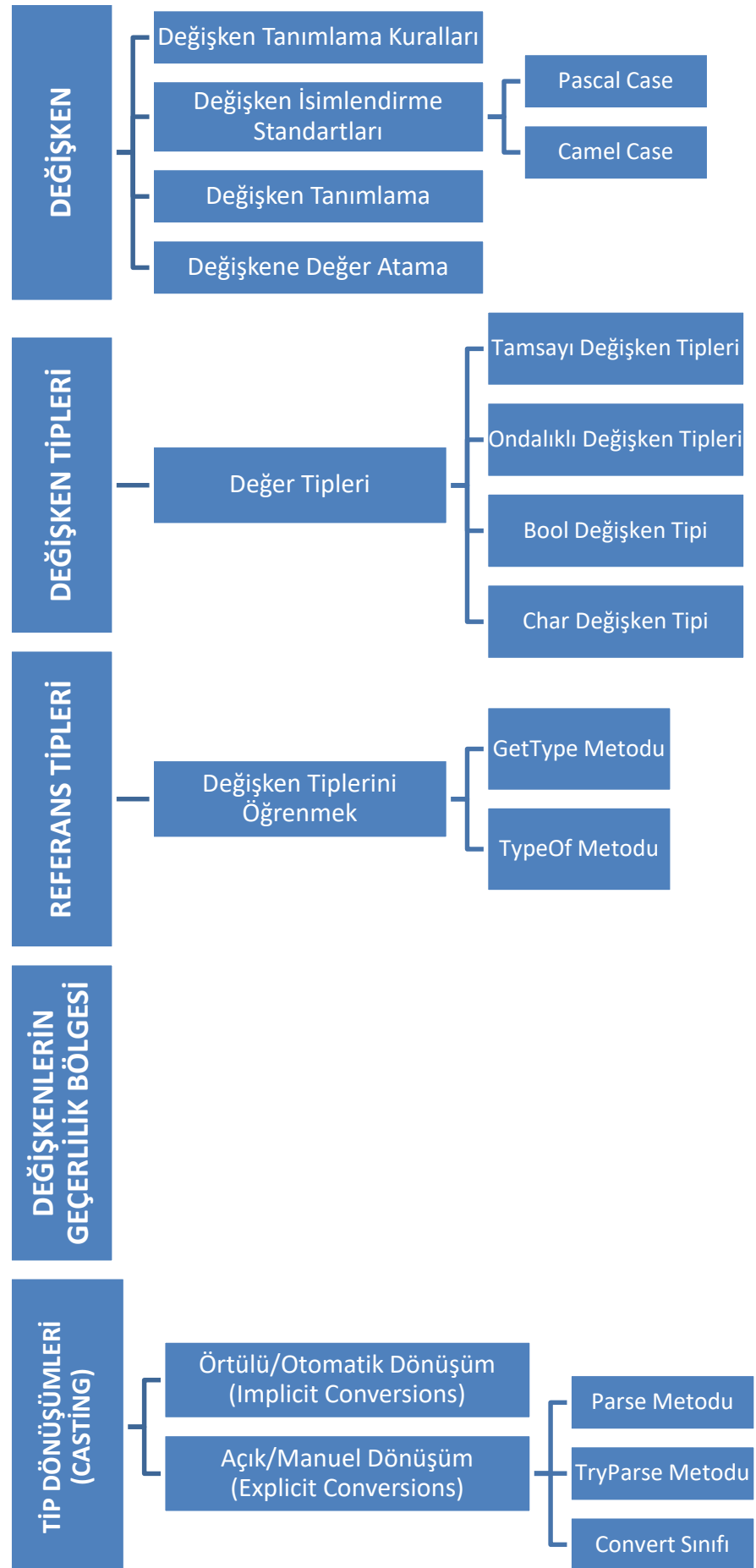
- Bu üniteyi çalıştıktan sonra;
- C# programlama dilinde değişkenler hakkında bilgi sahibi olacak,
- Değişken tanımlama kurallarını ve standartlarını öğrendikten sonra,
- Değişken tanımlamayı ve değişkene değer atama işlemlerini yapabilecek,
- Değişken tipleri ve tip dönüşümlerini de öğrenerek değişkenler üzerinde çalışabilmek için gerekli bilgiye sahip olacaksınız.



**Atatürk Üniversitesi**  
Açıköğretim Fakültesi

## GÖRSEL PROGRAMLAMA I Emir Sultan GÖKKAYA

# ÜNİTE 4



## GİRİŞ

Program yazarken her zaman sabit verilerle çalışılmaz. Çalışma zamanı içerisinde herhangi bir işlemin sonucuna ya da kullanıcı etkileşimine bağlı olarak ortaya çıkan bir verinin daha sonra kullanılmak üzere saklanması gerekebilir.

*Geçici olarak saklanan ve belirli tipte veri içeren bu yapılara değişken adı verilmektedir.*

Değişkenler her programlama dilinde var olan programlamanın en önemli unsurlarından bir tanesidir. Bu yüzden içerisinde değişkenlerin bulunmadığı bir program yapısı düşünülemez. Diğer programlama dillerinde olduğu gibi C# dilinde de birçok veri tipi ve bu veri tiplerini muhafaza eden değişken yapıları bulunmaktadır. *Her değişken tipinin özelliği ve kullanım amacı aynı olmadığından her değişken tipi farklı değişken yapıları tarafından temsil edilmektedir. Örneğin* metinsel ifadeler “string”, sayısal ifadeler ise “int, decimal, float veya double” gibi değişken tipleri tarafından temsil edilmektedirler.

*Program yazarken kodların sade ve başkaları tarafından kolay anlaşılır olması çok önemlidir. Bu yüzden değişkenler isimlendirilirken kullanılması gereken ifadelerin sade, kolay anlaşılabilir, değişken tanımlama kural ve standartlarına uygun olması gerekmektedir.* Aksi takdirde yazılan kodlar ve yapılan tanımlamalar; karmaşık, okuması ve anlaması oldukça güç ve zaman alan bir şekle dönüşecektir.

Tüm bu ön bilgiler ışığında kitabımızın bu bölümünde C# programlama dilinde değişkenleri, değişkenlerin tanımlanmasındaki önemli unsurları ve değişken tiplerini öğrenecek, daha sonra bu değişken tipleri üzerinde yapılabilecek işlemler hakkında detaylı bilgiye sahip olacaksınız.

## DEĞİŞKEN



Değişken için bellek üzerinde yapılan tüm işlemler değişkenin adresi olan adı üzerinden gerçekleştirilir.

Program yaşam döngüsü içerisinde belirli bir veri türünün değerini içeren yapılara değişken adı verilir (<https://www.azkod.com/> 2020). Değişkenler bellek üzerinde geçici olarak saklanırlar ve program yaşam döngüsünün sona ermesiyle birlikte bellekten düşerler.

Program içerisinde bir değişken tanımlı yapıldığında ilgili değişken için bellek üzerinde uygun bir alan ayrılır ve değişken sahip olduğu değeri kendisine verilen isimle birlikte bu alanda saklar.

Her değişkene, bellek üzerinde o değişkenin tutacağı veri tipine yetecek büyüklükte bir yer ayrılır. Bu yere *değişkenin adresi* adı verilir. Her değişkene bir isim verilir ve bu isim kullanılarak değişkene değer atanabilir, atanan değer okunabilir ve değiştirilebilir (Karaçay ve Karaçay 2008).

### Değişken Tanımlama Kuralları

C# programlama dilinde değişken tanımlanırken uyulması gereken birtakım kurallar bulunmaktadır. Bu kurallar aşağıdaki listede yer almaktadır.

- *Değişken ismi yalnızca harf, rakam ve \_ (alt çizgi) içerebilir.*

- *Değişken isimi harf ile başlamak zorundadır.*
- *Değişken ismi 255 karakterden fazla olamaz.*
- *Değişken ismi tanımlanırken karakterler arasında boşluk bırakılamaz.*
- *C# büyük küçük harf duyarlı bir programlama dili olduğundan; Değişken hangi isimle tanımlanmış ise o isimle kullanılabilir.*
- *Komut ifadeleri değişken adı olarak kullanılamaz. Örneğin; AND, OR ve NOT gibi.*

## Değişken İsimlendirme Standartları

Kurallara uygun şekilde tanımlanmış bir değişkene verilen isim; Her ne kadar programın çalışması için bir sorun teşkil etmese de tutarlı bir görünüm, kod bütünlüğü ve kodların rahat okunup anlaşılabilmesi açısından önem arz etmektedir. Bu sebeple resmi olarak belirlenmiş 2 (iki) adet isimlendirme standardı bulunmaktadır (Wanger vd. 2020). Bunlar; Pascal Case ve Camel Case'dir.

### Pascal Case

Değişken adını oluşturan her kelimenin baş harfinin büyük yazılarak değişkenin isimlendirildiği standarttır.



İsimlendirme standartlarının kullanım zorunluluğu bulunmamaktadır.



Örnek

- EnDusukFiyat
- OrtalamaFiyat
- EnYuksekFiyat

### Camel Case

Değişken adını oluşturan ilk kelimenin baş harfi küçük, diğer kelimelerin baş harfleri büyük yazılarak değişkenin isimlendirildiği standarttır.



Örnek

- enDusukFiyat
- ortalamaFiyat
- enYuksekFiyat

## Değişken Tanımlama

Her programlama dilinin bir syntax (sözdizimi) yapısı bulunmaktadır. Yazılan kodlar bu sözdizimi yapısına uygun olarak yazılmak zorundadır. C# dilinde ise değişken tanımlamak için kullanılan syntax (sözdizimi) yapısı;

`[Değişken Tipi] [Değişken Adı];`

şeklinde. İlk olarak değişken tipini tanımlayan ifade yazılır ardından değişken adı belirlenerek tanımlama işlemi gerçekleştirilir.



Örnek

•string Ad;

## Değişkene Değer Atama

Değer atama işlemi, değişkeni tanımlama esnasında yapılabileceği gibi daha sonrada gerçekleştirilebilir. *Bir değişkene değer atamak için C# da atama operatörü olan “=” eşittir sembolü kullanılır.* Daha sonra değişkenin tipine göre değer ataması gerçekleştirilir. Örnek 4.1.’de örnek değişken tanımlama işlemi gösterilmiştir.



Her değişken tipi sadece kendi türünden bir değer atanmasını kabul eder.

```
static void Main(string[] args)
{
    string Ad = "Emir Sultan";
}
```

Örnek 4.1. Değişken Tanımlama İşlemi

Örnek 4.2.’de ise daha önceden tanımlanmış bir değişkene sonradan değer atama işlemi görüntülenmektedir.

```
static void Main(string[] args)
{
    string Ad;
    Ad = "Emir Sultan";
}
```

Örnek 4.2. Değişkene Değer Atama İşlemi

## DEĞİŞKEN TİPLERİ

Değişkenler değer tipi ve referans tipi değişkenler olmak üzere 2 (iki) gruba ayrılırlar. Değer tipi değişkenler bellekte *stack* adı verilen alanlarda saklanırken referans tipi değişkenler *heap* adı verilen alanlarda saklanır (Aktaş 2021).

### Değer Tipleri

Değer tipleri; veriyi taşıyan ve taşıdığı veriye göre bellek üzerinde yer dolduran değişken tipleridir. Bellekte az yer kaplarlar ve hızlı bir şekilde erişilebilirler (<https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/06/c-'-ta-de%C4%9Fer-ve-referans-tipleri> 2013). Değer tipi değişkenler,

- *Tamsayı Değişken Tipleri*
- *Ondalıklı Sayı Değişken Tipleri*
- *Bool Değişken Tipi*
- *Char Değişken Tipi*

şeklinde listelenebilir.



Değer tipi değişkenler bellekte “stack” adı verilen alanlarda tutulurlar.

## Tamsayı değişken tipleri

Tam sayıları saklamak ve üzerlerinde işlem yapabilmek için kullanılan değişken tipleridir. Tablo 4.1.'de tam sayı değişken tipleri özellikleri ile birlikte liste halinde verilmiştir.



Sayı değişken tiplerinin varsayılan değerleri 0 (sıfır)'dır.

**Tablo 4.1.** Tamsayı Değişken Tipleri Listesi

Değişken Tipi	İsim Uzayı	Uzunluk (Byte)	Değer Aralığı
<b>byte</b>	System.Byte	1 byte	0, ..., 255 (tam sayı)
<b>sbyte</b>	System.Byte	1 byte	-128, ..., 127 (tam sayı)
<b>short</b>	System.Int16	2 byte	-32.768, ..., 32.767 (tam sayı)
<b>ushort</b>	System.UInt16	2 byte	0, ..., 65.535 (tam sayı)
<b>int</b>	System.Int32	4 byte	-2.147.483.648, ..., 2.147.483.647 (tam sayı)
<b>uint</b>	System.UInt32	4 byte	0, ..., 4.294.967.295 (tam sayı)
<b>long</b>	System.Int64	8 byte	-9.223.372.036.854.775.808,... 9.223.372.036.854.775.807 (tam sayı)
<b>ulong</b>	System.UInt64	8 byte	0, ..., 18.446.744.073.709.551.615 (tam sayı)
<b>nint</b>	System.IntPtr	4 byte	Platforma bağlıdır.
<b>nuint</b>	System.UIntPtr	8 byte	Platforma bağlıdır.

Örnek 4.3.'te tamsayı değişken tiplerinin tanımlanma şekilleri görüntülenmektedir.

```
static void Main(string[] args)
{
    byte byteDegisken = 255;
    sbyte sbyteDegisken = -128;
    short shortDegisken = 32_767;
    ushort ushortDegisken = 65535;
    int intDegisken = 1;
    uint uintDegisken = 1000;
    long longDegisken = 9223372036854775807;
    ulong ulongDegisken = 571;
}
```

**Örnek 4.3.** Tamsayı Değişken Tiplerinin Tanımlanma Şekilleri

## Ondalıklı sayı değişken tipleri

Ondalıklı sayıları saklamak ve üzerlerinde işlem yapabilmek için kullanılan değişken tipleridir. Tablo 4.2.'de ondalıklı sayı değişken tipleri özellikleri ile birlikte liste halinde verilmiştir.

**Tablo 4.2.** Kesirli Sayı Değişken Tipleri Listesi

Değişken Tipi	İsim Uzayı	Uzunluk (Byte)	Değer Aralığı
<b>float</b>	System.Double	4 byte	$\pm 1.5 \cdot 10^{-45}$ , ..., $\pm 3.4 \cdot 10^{38}$ (reel sayı)
<b>double</b>	System.Double	8 byte	$\pm 5.0 \cdot 10^{-324}$ , ..., $\pm 1.7 \cdot 10^{308}$ (reel sayı)
<b>decimal</b>	System.Decimal	16 byte	$\pm 1.5 \cdot 10^{-28}$ , ..., $\pm 7.9 \cdot 10^{28}$ (reel sayı)

Örnek 4.4.'te ondalıklı sayı değişken tiplerinin tanımlanma şekilleri görüntülenmektedir.

```
static void Main(string[] args)
{
    float floatDegisken = 3000.5F;
    double doubleDegisken = 1.0;
    decimal decimalDegisken = 2.1m;
}
```

**Örnek 4.4.** Ondalıklı Sayı Değişken Tiplerinin Tanımlama Şekilleri

### Bool değişken tipi

Bool veri tipi yalnızca iki değer alabilir. Bunlar “true” ve “false”dur. Bellekte 1 byte yer kaplar. Bool değişken tipi, programın akışının yönlendirilmesinde ve döngü yapılarında sıkça kullanılmaktadır. Tablo 4.3.'te bool değişken tipine ait özellikler yer almaktadır.

**Tablo 4.3.** Bool Değişken Tipi Özellikleri

Değişken Tipi	İsim Uzayı	Uzunluk (Byte)	Değer Aralığı
<b>bool</b>	System.Boolean	1 byte	true, false

Örnek 4.5.'te bool değişken tipinin tanımlanma şekli görüntülenmektedir.

```
static void Main(string[] args)
{
    bool boolDegisken = true;
}
```

**Örnek 4.5.** Bool Değişken Tipinin Tanımlanma Şekli

### Char değişken tipi

C#’da karakterler, “Unicode” adı verilen ve uzunluğu 2 byte olan bir sistemle belirlenir. Bu sistem içerisinde tüm alfabeler, sayılar, özel karakter ve semboller yer almaktadır. “Char” veri tipi de bu “Unicode” karakterler içerisinde sadece tek bir değer alabilen bir değişken tipidir. Tablo 4.4.'te char değişken tipine ait özellikler verilmiştir.



“bool” değişken tipinin varsayılan değerleri bulunmamaktadır. “true” ya da “false” türünden bir değer kullanıcı tarafından atanması zorunludur.

Tablo 4.4. Char değişken tipi özellikleri.

	İsim Uzayı	Uzunluk (Byte)	Değer Aralığı
char	System.Char	2 byte	Tüm Unicode karakterler.

Örnek 4.6.'da "char" değişken tipinin tanımlanma şekli görüntülenmektedir.



Referans tipi değişkenler bellekte "heap" adı verilen alanlarda tutulurlar.

```
static void Main(string[] args)
{
    char charDegisken = '#';
}
```

Örnek 4.6. Char Değişken Tipinin Tanımlama Şekli

## Referans Tipleri

Referans tipleri; bellek bölgesinde veri yerine adresi tutarlar ve o adresin gösterdiği yerde de veri tutulur (<https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/06/c-/-ta-de%C4%9Fer-ve-referans-tipleri> 2013). Tablo 4.5.'de referans tipi değişkenler açıklanmıştır.

Tablo 4.5. Referans Tipi Değişkenler

Değişken Tipi	Açıklama	Örnek Kullanım
string	Unicode karakter dizisi. Metinsel ifadeleri saklar.	string OrnekDegisken = "ATA";
object	Genel veri tipidir. Tip dönüşümü zorunludur.	object OrnekDegisken = 1881;
dynamic	Genel veri tipidir. Tip dönüşümü zorunlu değildir.	dynamic OrnekDegisken = "∞";

Örnek 4.7.'de referans tipi değişkenlerin tanımlanma şekilleri görüntülenmektedir.

```
static void Main(string[] args)
{
    string stringDegisken = "Değişkenler ve Veri Yapıları";
    object objectDegisken = 29 + 11;
    dynamic dynamicDegisken = false;
}
```

Örnek 4.7. Referans Tipi Değişken Tiplerinin Tanımlama Şekilleri

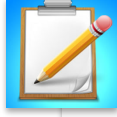
Ayrıca "Var" anahtarı, değişken tanımlı yaparken tip belirtmeksizin tanım yapmamızı sağlamaktadır. Var ile tanımlanan bir değişkene atanan ilk değer program derlendiği anda değişkenin veri türünü belirlemektedir(Okumuş 2012).



Örnek

```
•var Ad = "Atatürk Üniversitesi";
•var Donem = 2021;
```





Bireysel Etkinlik

- Visual Studio 2019 geliştirme ortamında size anlatılan adımları takip ederek, değişken tanımlama kurallarını ve isimlendirme standartlarına dikkat ederek bir değişken tanımlayın ve tanımladığınız değişkene değer ataması gerçekleştirin.

## Değişken Tiplerini Öğrenmek



“var” anahtar sözcüğü ile yapılan tanımlamalarda değişkene atanan değere göre tipi otomatik olarak belirlenir.

Genellikle “var” anahtar sözcüğü ile yapılan tanımlamalarda; değişkenin sahip olduğu veri tipini öğrenmek, doğruluğunu kontrol etmek ve program akışını yönlendirmek gibi nedenlerden dolayı değişkenin tipinin çalışma zamanında öğrenilmesine ihtiyaç duyulur. Bunun için kullanılacak 2 (iki) adet yardımcı metod bulunmaktadır. *Bunlar; GetType ve TypeOf metotlarıdır.*

### GetType metodu

Bir değişkenin tipinin öğrenilmesinde kullanılır. “Type” türünden sonuç üretmektedir. Örnek 4.8.’de “GetType” metodunun kullanımı için hazırlanan örnek bir program görüntüsü yer almaktadır.

```
static void Main(string[] args)
{
    var KurumAdi = "Atatürk Üniversitesi";
    Console.WriteLine("KurumAdi değişkenin tipi: {0}", KurumAdi.GetType());

    var Yil = 1957;
    Console.WriteLine("Yil değişkenin tipi: {0}", KurumAdi.GetType());
}
```

**Örnek 4.8.** GetType Metodu Kullanımı İçin Hazırlanan Örnek Program

Örnek 4.8’de GetType metodunun kullanımı için hazırlanan örnek programın çıktısı yer almaktadır.

```
C:\WINDOWS\system32\cmd.exe
KurumAdi değişkenin tipi: System.String
Yil değişkenin tipi: System.Int32
Press any key to continue . . .
```

**Resim 4.1.** GetType Metodu Kullanımı İçin Hazırlanan Örnek Program Çıktısı



Örnek 4.9.'da kullanılan "if else" komutu Ünite 5'te detaylı olarak açıklanacaktır.

## TypeOf metodu

Bir değişken tipinin başka bir değişken tipi ile karşılaştırılması ya da kontrol edilmesinde kullanılır. *Örnek 4.9.'da "TypeOf" metodunun kullanımı için hazırlanan örnek bir program görüntüsü yer almaktadır. Programda "Sayi1" isimli değişkene "int" türünden, "Sayi2" isimli değişkene ise "string" türünden değer atamaları yapılmıştır. Değişkenlerin türü "GetType" metodu ile alınarak "typeof(int)" ile karşılaştırılmıştır.*

```
static void Main(string[] args)
{
    var Sayi1 = 12345;
    if (Sayi1.GetType() == typeof(int))
        Console.WriteLine("Girilen Sayi1 bir sayı.");
    else
        Console.WriteLine("Girilen Sayi1 bir sayı değil.");

    var Sayi2 = "Bir İki Üç Dört Beş";
    if (Sayi2.GetType() == typeof(int))
        Console.WriteLine("Girilen Sayi2 bir sayı.");
    else
        Console.WriteLine("Girilen Sayi2 bir sayı değil.");
}
```

**Örnek 4.9.** Typeof Metodu Kullanımı İçin Hazırlanan Örnek Program

*Örnek 4.9.'da yapılmış olan karşılaştırma sonucunda "Sayi1" isimli değişkenin "int" türünden olduğu, "Sayi2" isimli değişkenin ise "int" türüne ait olmadığı görülmüş ve program akışı bu doğrultuda yönlendirilmiştir.*

**Resim 4.2.** Typeof Metodu Kullanımı İçin Hazırlanan Örnek Program Çıktısı

## DEĞİŞKENLERİN GEÇERLİLİK BÖLGESİ

Değişkenler tanımlanmış oldukları {...} (küme parantezi) bloğu içerisinde geçerlidirler. *Değişkenlere tanımlı oldukları blok içerisinde ve alt bloklardan ulaşılabilirken üst bloklardan alt bloktaki değişkenlere erişilemez.*

Örnek 4.10.'de görüldüğü üzere "Blok 3" içerisinde "Blok 2"de tanımlı "Sayi1" ve "Sayi2" isimli değişkenlere erişilerek bu sayılar toplanmış ve sonuç "Toplam" isimli değişkene aktarılmıştır. "Toplam" değişkenine tanımlı olduğu "Blok 3" içerisinde erişilebilmiş ancak "Blok 2"den erişilmek istendiğinde hata ile karşılaşmıştır.



Bir kod bloğunun işlevi sona erdiğinde o blok için bellekte ayrılan alan silinir.

```
class Program
{ //Blok 1; Program bloğu başlangıcı
    static void Main(string[] args)
    { //Blok 2; Main bloğu başlangıcı

        int Sayi1 = 10;
        int Sayi2 = 20;

        if (Sayi1>Sayi2)
        { //Blok 3; if bloğu başlangıcı

            int Toplam = Sayi1 + Sayi2; //Sayi1 ve Sayi2 değişkeni
            toplanarak sonuç Toplam isimli değişkene atanıyor.
            Console.WriteLine(Toplam); //Toplam değişkeninin sonucu ekrana
            yazdırılıyor.

        } //Blok 3; if bloğu bitiş

        Console.WriteLine(Toplam); //Toplam değişkenine tanımlı olduğu blok
        dışından ulaşamaz.

    } //Blok 2; Main bloğu bitiş
} //Blok 1; Program bloğu bitiş
```

Örnek 4.10. Değişkenlerin Geçerlilik Bölgesi

## TİP DÖNÜŞÜMLERİ (Casting)

Farklı tipteki veriler üzerinde çalışılırken karşılaştırma, mantıksal ve aritmetiksel işlemler için türler arası dönüşüm yapılması gerekir. Bu işleme Tip Dönüşümü (casting) adı verilir. Başka bir deyişle; Tip dönüşümü, bir veri tipinin bir başka veri tipine dönüştürülmesidir.

C# programlama dilinde 4 (dört) adet tip dönüşümü bulunmaktadır. Bunlar;

- Örtülü/Otomatik Dönüşüm (Implicit Conversions)
- Açık/Manuel Dönüşüm (Explicit Conversions)
- Kullanıcı Tanımlı Dönüşüm (User-defined conversions)
- Yardımcı Sınıflarla Dönüşüm (Conversions with helper classes)

şeklinde listelenebilirler (Wanger 2020).

*Kullanıcı Tanımlı Dönüşüm (User-defined conversions) ve Yardımcı Sınıflarla Dönüşüm işlemleri daha ileri seviye bir C# bilgisi gerektirdiğinden dolayı kitabımızın bu bölümünde sadece Örtülü/Otomatik Dönüşüm (Implicit Conversions) ve Açık/Manuel Dönüşüm (Explicit Conversions) işlemlerine değinilecektir.*

### Örtülü/Otomatik Dönüşüm (Implicit Conversions)

Dönüşecek değişkenin bellekte kapladığı alan, dönüştürülmek istenen değişken tipinin kapsadığı alandan daha küçük ise bu dönüşüm işlemi C# tarafından otomatik olarak gerçekleştirilir.

Bir örnek ile bu işlemi açıklamak için “int” veri tipinden “long” veri tipine dönüşüm sürecini ele alalım.



Dönüşüm işlemi sonrasında ilgili değişkenin bellekteki adresi de değişmektedir.

- “int” veri tipinde tanımlanmış “IntSayı” isimli değişkenin bellekte kapladığı alan 4 byte’dır.
- Long veri tipinde tanımlanmış olan “LongSayı” isimli değişkenin bellekte kapladığı alan 8 byte’dır.
- “IntSayı” değişkeninin değeri “LongSayı” isimli değişkene aktarıldığında, “int” tipindeki değişken bellekte 4 byte’lık alandan 8 byte’lık alana taşınır. Yani değişken küçük alandan daha büyük bir alana taşınır.

Bu dönüşüm tipine Örtülü/Otomatik Dönüşüm (Implicit Conversions) adı verilir. Tablo 4.6.’da Örtülü/Otomatik Dönüşüm (Implicit Conversions) işlemini destekleyen veri tipleri verilmiştir.

**Tablo 4.6.** Örtülü/Otomatik Dönüşüm (Implicit Conversions) İşlemini Destekleyen Veri Tipleri

Veri Tipi (Küçük olan)	Dönüştürüldüğü Veri Tipi (Büyük olan)
byte	short, int, long, float, double
short	int, long, float, double
int	long, float, double
long	float, double
float	double

Örnek 4.11.’te Örtülü/Otomatik Dönüşüm (Implicit Conversions) işlemi için hazırlanmış programa bakıldığında 2 (iki) adet dönüşüm işlemi görülmektedir. Birinci dönüşüm işleminde “int” türündeki değişkenin değeri “long” tipine dönüştürülmüştür. İkinci de ise “long” türündeki değişkenin değeri “float” tipine dönüştürülmüştür.

```
//Örtülü/Otomatik Dönüşüm (Implicit Conversions) İşlemi
static void Main(string[] args)
{
    // int türünde bir değişken tanımlanıyor.
    int intDegisken = 57;

    // int türündeki değişkenin değeri long türünde bir değişkene atanıyor.
    long longDegisken = intDegisken;

    // long türündeki değişkenin değeri float türünde bir değişkene atanıyor.
    float floatDegisken = longDegisken;

    Console.WriteLine("Int değeri " + intDegisken);
    Console.WriteLine("Long değeri " + longDegisken);
    Console.WriteLine("Float değeri " + floatDegisken);
}
```

**Örnek 4.11.** Örtülü/Otomatik Dönüşüm (Implicit Conversions) İşlemi



Dönüşüm işlemi sonrasında ilgili değişkenin bellekteki adresi değişmektedir.

Resim 4.3.'teki program çıktısına bakıldığında dönüşüm işlemi sonrası değişkenlerin değerlerinin aynı olduğu görülmektedir.

```
C:\WINDOWS\system32\cmd.exe
Int değeri 57
Long değeri 57
Float değeri 57
Press any key to continue . . .
```

Resim 4.3. Örtülü/Otomatik Dönüşüm (Implicit Conversions) İşlemi Program Çıktısı

### Açık/Manuel Dönüşüm (Explicit Conversions)

Dönüşecek değişkenin bellekte kapladığı alan, dönüştürülmek istenen değişken tipinin kapsadığı alandan daha büyük ise bu dönüşüm işlemi C# tarafından otomatik olarak gerçekleştirilemez. Ayrıca bu durum programda derleme hatasına sebep olur.

Dönüşecek değişkenin değerinin, dönüştürülmek istenen hedef veri tipinin değer aralığında olduğu biliniyor ise manuel dönüşüm işlemi gerçekleştirilerek derleyicinin bu işlemi hata olarak algılamaması sağlanabilir. Bunun için atama işlemi esnasında dönüştürülmek istenen değişkenin önüne parantez sembolü içerisinde değişken tipi ifadesi eklenir.

Bir örnek ile bu işlemi açıklamak için “double” veri tipinden “int” veri tipine dönüşüm sürecini ele alalım.

- “double” veri tipinde tanımlanmış “DoubleSayı” isimli değişkenin bellekte kapladığı alan 8 byte’dır.
- “int” veri tipinde tanımlanmış olan “IntSayı” isimli değişkenin bellekte kapladığı alan 4 byte’dır.
- “DoubleSayı” değişkeninin değeri “IntSayı” isimli değişkene aktarılmak istendiğinde; “double” tipindeki değişken bellekte 8 byte’lık alandan 4 byte’lık alana taşınacağından dolayı veri kaybı söz konusu olacaktır. Çünkü değişken kendisinden daha küçük bir alana taşınmak istenmektedir.
- Bu sebeple C# bu işlemi gerçekleştirilmeyecek ve bir derleme hatası olarak değerlendirilecektir.

Dönüşüm (casting) ifadesi kullanılarak gerçekleştirilen bu dönüşüm işlemine Açık/Manuel Dönüşüm (Explicit Conversions) adı verilmektedir.

Örnek 4.12.’de “double” türünde tanımlanmış olan değişkenin değerinin “int” tipinde tanımlanmış olan değişkene aktarılması esnasında alınan derleme hatası görülmektedir.



Açık/Manuel Dönüşüm (Explicit Conversions) işleminde dönüşüm sırasında veri tipleri arasında uyumsuzluk olması halinde çalışma zamanında “InvalidCastException” hatası meydana gelecektir.

```
//Açık/Manuel Dönüşüm (Explicit Conversions) İşlemi
static void Main(string[] args)
{
    double doubleDegisken = 765.12;

    int intDegisken = doubleDegisken;

    Console.WriteLine("intDegisken değeri " + intDegisken);
}
```

**Örnek 4.12.** Açık/Manuel Dönüşüm (Explicit Conversions) Derleme Hatası

Örnek 4.13.'te "double" türünde tanımlanmış olan değişkenin değerinin "int" tipinde dönüştürülebilmesi için atama işlemi esnasında "doubleDegisken" isimli değişkenin önüne "(int)" ifadesi eklenmiştir.

```
//Açık/Manuel Dönüşüm (Explicit Conversions) İşlemi
static void Main(string[] args)
{
    double doubleDegisken = 765.12;

    int intDegisken = (int)doubleDegisken;

    Console.WriteLine("intDegisken değeri " + intDegisken);
}
```

**Örnek 4.13.** Açık/Manuel Dönüşüm (Explicit Conversions) İşlemi

Resim 4.4.'de "double" değişkeninin kendisinden daha küçük bir değişken tipine dönüşümü sonrasındaki program çıktısı yer almaktadır.

**Resim 4.4.** Açık/Manuel Dönüşüm (Explicit Conversions) İşlemi Program Çıktısı

Ayrıca Açık/Manuel Dönüşüm (Explicit Conversions) işlemleri için "Parse", "TryParse" ve "Convert" yöntemleri de kullanılmaktadır.

### Parse metodu

"String" veri tipindeki bir değişkenin değerini farklı bir değişken türüne dönüştürmek için kullanılır. "Parse" metodu kullanılırken dönüşecek değişkenin değerinin "null" olmaması ve değerinin hedef değişkenin değer aralığında olması gerekmektedir.

Örnek 4.14.'de "Parse" metodunun "int" veri tipi üzerinde kullanılmasına yönelik örnek program görüntüsü yer almaktadır.



Referans tipi  
değişkenlerde "Parse"  
metodu  
bulunmamaktadır.

```
static void Main(string[] args)
{
    string StringSayi = "110";

    int IntSayi = int.Parse(StringSayi); //StringSayi değişkeninin değerini
    IntSayi değişkenine aktarabilmek için manuel dönüşüm işlemi
    yapılmıştır.
}
```

**Örnek 4.14.** “Parse” Metodunun “int” Veri Tipi Üzerinde Kullanılması

### TryParse metodu

İşlevi “Parse” komutu ile aynıdır. Tek farkı; Dönüşecek değişkenin değerinin “null” veya değer aralığı dışında olması ya da değişkenin farklı bir değişken tipine ait olması durumunda meydana gelecek hatada geriye 0 (sıfır) değeri döndürmesidir. *“out” eki ile birlikte kullanılır ve sonuç “out” referanslı değişkene aktarılır.*

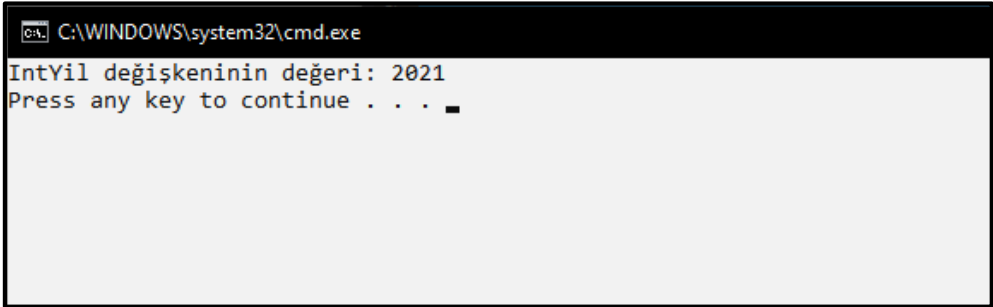
Örnek 4.15.’de “TryParse” metodunun kullanımına yönelik örnek program görüntüsü yer almaktadır.

```
static void Main(string[] args)
{
    string StringYil = DateTime.Now.Year.ToString(); //İçerisinde bulunan
    yıl değeri string türüne çevirilerek değişkene aktarılıyor.

    if (int.TryParse(StringYil, out int IntYil))
    {
        Console.WriteLine("IntYil değişkeninin değeri: {0}", IntYil);
    }
}
```

**Örnek 4.15.** “Tryparse” Metodunun “int” Veri Tipi Üzerinde Kullanılması

Resim 4.5.’de “TryParse” metodunun kullanımına yönelik örnek program ekran çıktısı almaktadır.



**Resim 4.5.** “Tryparse” Metodunun “int” Veri Tipi Üzerinde Kullanılmasına Yönelik Program Çıktısı

### Convert sınıfı

Değişken tipleri arasında dönüşüm yapmak için kullanılan bir sınıftır. “Convert” sınıfının metotları kullanılarak yapılabilecek tip dönüşümleri Tablo 4.7.’de listelenmiştir.



TryParse metodu “if” komutu ile birlikte kullanıldığında; Veri tipi uyumsuzluğu nedeniyle dönüşüm işlemi gerçekleştirilemez ise geriye dönen 0 (sıfır) değeri “false” olarak değerlendirilir.

Tablo 4.7. "Convert" Sınıfının Metotları ve Açıklamaları

Metot	Açıklama
<b>ToBase64String()</b>	Verilen bir string değeri Base64 şifreleme tipine dönüştürür.
<b>FromBase64String()</b>	Base64 şifreleme tipinden "string" tipine dönüştürme yapar.
<b>ToBoolean()</b>	Boolean tipine dönüştürme yapar.
<b>ToByte()</b>	Byte tipine dönüştürme yapar.
<b>ToChar()</b>	Char tipine dönüştürme yapar.
<b>ToDateTime()</b>	Tarih ve zaman tipine dönüştürme yapar.
<b>ToDecimal()</b>	Decimal tipine dönüştürme yapar.
<b>ToDouble()</b>	Double tipine dönüştürme yapar.
<b>ToInt16()</b>	Int16 tipine dönüştürme yapar.
<b>ToInt32()</b>	Int32 tipine dönüştürme yapar.
<b>ToInt64()</b>	Int64 tipine dönüştürme yapar.
<b>ToSByte()</b>	SByte tipine dönüştürme yapar.
<b>ToSingle()</b>	Single tipine dönüştürme yapar.
<b>ToString()</b>	String tipine dönüştürme yapar.
<b>ToUInt16()</b>	UInt16 tipine dönüştürme yapar.
<b>ToUInt32()</b>	UInt32 tipine dönüştürme yapar.
<b>ToUInt64()</b>	UInt64 tipine dönüştürme yapar.



Convert sınıfındaki her metot farklı türde veriler kabul edebilir.





## Özet

### • DEĞİŞKEN

Program yaşam döngüsü içerisinde belirli bir veri türünün değerini içeren yapılara değişken adı verilir.

Her değişkene, bellek üzerinde o değişkenin tutacağı veri tipine yetecek büyüklükte bir yer ayrılır. Bu yere *değişkenin adresi* adı verilir. Her değişkene bir isim verilir ve bu isim kullanılarak değişkene değer atanabilir, atanan değer okunabilir ve değiştirilebilir.

### • Değişken Tanımlama Kuralları

Değişken tanımlanırken aşağıdaki listede belirtilen kurallara uyulması gerekmektedir.

- Değişken ismi yalnızca harf, rakam ve \_ (alt çizgi) içerebilir.
- Değişken ismi harf ile başlamak zorundadır.
- Değişken ismi 255 karakterden fazla olamaz.
- Değişken ismi tanımlanırken karakterler arasında boşluk bırakılamaz.
- C# büyük küçük harf duyarlı bir programlama dili olduğundan; Değişken hangi isimle tanımlanmış ise o isimle kullanılabilir.
- Komut ifadeleri değişken adı olarak kullanılamaz. Örneğin; AND, OR ve NOT gibi.

### • Değişken İsimlendirme Standartları

Kurallara uygun şekilde tanımlanmış bir değişkene verilen isim; Her ne kadar programın çalışması için bir sorun teşkil etmese de tutarlı bir görünüm, kod bütünlüğü ve kodların rahat okunup anlaşılabilmesi açısından önem arz etmektedir. Bu sebeple resmi olarak belirlenmiş 2 (iki) adet isimlendirme standardı bulunmaktadır.

### • Pascal Case

Değişken adını oluşturan her kelimenin baş harfinin büyük yazılarak değişkenin isimlendirildiği standarttır.

### • Camel Case

Değişken adını oluşturan ilk kelimenin baş harfi küçük, diğer kelimelerin baş harfleri büyük yazılarak değişkenin isimlendirildiği standarttır.

### • Değişken Tanımlama

C# programlama dilinde değişken tanımlamak için;  
[Değişken Tipi] [Değişken Adı]; şeklinde tanımlama yapılmalıdır.

### • Değişkene Değer Atama

Değer atama işlemi, değişkeni tanımlama esnasında yapılabileceği gibi daha sonrada gerçekleştirilebilir. Bir değişkene değer atamak için C# da atama operatörü olan "=" eşittir sembolü kullanılır.

### • Değişken Tipleri

Değişkenler değer tipi ve referans tipi değişkenler olmak üzere 2 (iki) gruba ayrılırlar. Değer tipi değişkenler bellekte *stack* adı verilen alanlarda saklanırken referans tipi değişkenler *heap* adı verilen alanlarda saklanır.

### • Değer Tipleri

Değer tipleri; veriyi taşıyan ve taşıdığı veriye göre bellek üzerinde yer dolduran değişken tipleridir. Bellekte az yer kaplarlar ve hızlı bir şekilde erişilebilirler. Değer tipi değişkenler, Tam Sayı Değişken Tipleri, Ondalık Sayı Değişken Tipleri, Bool Değişken Tipi ve Char Değişken tipidir.

### • Tamsayı Değişken Tipleri

Tam sayıları saklamak ve üzerlerinde işlem yapabilmek için kullanılan değişken tipleridir. Tamsayı değişken tipleri byte, sbyte, short, ushort, int, uint, long, ulong, nint ve nuint'dir.

### • Ondalık Sayı Değişken Tipleri

Ondalık sayıları saklamak ve üzerlerinde işlem yapabilmek için kullanılan değişken tipleridir. Ondalık sayı değişken tipleri float, double ve decimal'dir.



## Özet (devamı)

- **Bool Değişken Tipi**  
Bool veri tipi yalnızca iki değer alabilir. Bunlar “true” ve “false”dur. Bellekte 1 byte yer kaplar. Bool değişken tipi, programın akışının yönlendirilmesinde ve döngü yapılarında sıkça kullanılmaktadır.
- **Char Değişken Tipi**  
C#’da karakterler, “Unicode” adı verilen ve uzunluğu 2 byte olan bir sistemle belirlenir. Char veri tipi de bu “Unicode” karakterler içerisinde sadece tek bir değer alabilen bir değişken tipidir.
- **Referans Tipleri**  
Referans tipleri; bellek bölgesinde veri yerine adresi tutarlar ve o adresin gösterdiği yerde de veri tutulur. Referans tipi değişkenler string, object ve dynamic’dir.
- **Değişken Tiplerini Öğrenmek**  
Değişkenin sahip olduğu veri tipini öğrenmek, doğruluğunu kontrol etmek ve program akışını yönlendirmek gibi nedenlerden dolayı değişkenin tipinin çalışma zamanında öğrenilmesine ihtiyaç duyulur. Bunun için kullanılabilecek 2 (iki) adet yardımcı metod bulunmaktadır. Bunlar; GetType ve TypeOf metotlarıdır.
- **GetType Metodu**  
Bir değişkenin tipinin öğrenilmesinde kullanılır. “Type” türünden sonuç üretmektedir.
- **typeof Metodu**  
Bir değişken tipinin başka bir değişken tipi ile karşılaştırılması ya da kontrol edilmesinde kullanılır.
- **Değişkenlerin Geçerlilik Bölgesi**  
Değişkenler tanımlanmış oldukları {...} (küme parantezi) bloğu içerisinde geçerlidirler. Değişkenlere tanımlı oldukları blok içerisinde ve alt bloklardan ulaşılabilirken üst bloklardan alt bloktaki değişkenlere erişilemez.
- **Tip Dönüşümleri (Casting)**  
Farklı tipteki veriler üzerinde çalışılırken karşılaştırma, mantıksal ve aritmetiksel işlemler için türler arası dönüşüm yapılması gerekir. Bu işleme Tip Dönüşümü (casting) adı verilir. Dört adet tip dönüşümü bulunmaktadır. Bunlar; Örtülü/Otomatik Dönüşüm (Implicit Conversions), Açık/Manuel Dönüşüm (Explicit Conversions), Kullanıcı Tanımlı Dönüşüm (User-defined conversions) ve Yardımcı Sınıflarla Dönüşüm (Conversions with helper classes)’dir.
- **Örtülü/Otomatik Dönüşüm (Implicit Conversions)**  
Dönüşecek değişkenin bellekte kapladığı alan, dönüştürülmek istenen değişken tipinin kapsadığı alandan daha küçük ise bu dönüşüm işlemi C# tarafından otomatik olarak gerçekleştirilir.
- **Açık/Manuel Dönüşüm (Explicit Conversions)**  
Dönüşecek değişkenin bellekte kapladığı alan, dönüştürülmek istenen değişken tipinin kapsadığı alandan daha büyük ise bu dönüşüm işlemi C# tarafından otomatik olarak gerçekleştirilemez. Dönüşecek değişkenin değerinin, dönüştürülmek istenen hedef veri tipinin değer aralığında olduğu biliniyor ise manuel dönüşüm işlemi gerçekleştirilerek bu dönüşüm sağlanır. Bunun için atama işlemi esnasında dönüştürülmek istenen değişkenin önüne parantez sembolü içerisinde değişken tipi ifadesi eklenir.
- **Parse Metodu**  
“String” veri tipindeki bir değişkenin değerini farklı bir değişken türüne dönüştürmek için kullanılır. “Parse” metodu kullanılırken dönüşecek değişkenin değerinin “null” olmaması ve değerinin hedef değişkenin değer aralığında olması gerekmektedir.



## Özet (devamı)

### •TryParse Metodu

İşlevi "Parse" komutu ile aynıdır. Tek farkı; Dönüşecek değişkenin değerinin "null" veya değer aralığı dışında olması ya da değişkenin farklı bir değişken tipine ait olması durumunda meydana gelecek hatada geriye 0 (sıfır) değeri döndürmesidir. "out" eki ile birlikte kullanılır ve sonuç "out" referanslı değişkene aktarılır.

### •Convert Sınıfı

Değişken tipleri arasında dönüşüm yapmak için kullanılan bir sınıftır. Convert sınıfının metotları ise şu şekildedir;

ToBase64String()	Verilen bir string değeri Base64 şifreleme tipine dönüştürür.
FromBase64String()	Base64 şifreleme tipinden "string" tipine dönüştürme yapar.
ToBoolean()	Boolean tipine dönüştürme yapar.
ToByte()	Byte tipine dönüştürme yapar.
ToChar()	Char tipine dönüştürme yapar.
ToDateTime()	Tarih ve zaman tipine dönüştürme yapar.
ToDecimal()	Decimal tipine dönüştürme yapar.
ToDouble()	Double tipine dönüştürme yapar.
ToInt16()	Int16 tipine dönüştürme yapar.
ToInt32()	Int32 tipine dönüştürme yapar.
ToInt64()	Int64 tipine dönüştürme yapar.
ToSByte()	SByte tipine dönüştürme yapar.
ToSingle()	Single tipine dönüştürme yapar.
ToString()	String tipine dönüştürme yapar.
ToUInt16()	UInt16 tipine dönüştürme yapar.
ToUInt32()	UInt32 tipine dönüştürme yapar.
ToUInt64()	UInt64 tipine dönüştürme yapar.

## DEĞERLENDİRME SORULARI

1. Değişkenlerle ilgili aşağıda yer alan ifadelerden hangisi yanlıştır?
  - a) Bellek üzerinde geçici olarak saklanırlar
  - b) Bellek üzerinde kalıcı olarak saklanırlar
  - c) Değeri değiştirilebilir
  - d) Belirli bir veri tipine ait değer içerir
  - e) Her değişken bir isme sahiptir
2. Aşağıdakilerden hangisi bir değişken tanımlama kuralı değildir?
  - a) Değişken ismi harf ile başlamak zorundadır
  - b) Değişken ismi 255 karakterden fazla olamaz
  - c) Değişken ismi tanımlanırken karakterler arasında boşluk bırakılamaz
  - d) Değişken ismi yalnızca harf, rakam ve \_ (alt çizgi) içerebilir.
  - e) Değişken isminin ilk karakteri büyük harfle başlamak zorundadır.
3. Aşağıdakilerden hangisi Pascal Case standardı ile tanımlanmış bir değişken adıdır?
  - a) adSoyad
  - b) \_adSoyad
  - c) AdSoyad
  - d) ad\_soyad
  - e) \_adsoyad\_
4. Aşağıdakilerden hangisi değişkenlerin bellekte tutulduğu noktalardan birisidir?
  - a) heap
  - b) heat
  - c) slack
  - d) sRAM
  - e) DRAM
5. Aşağıdakilerden hangisi değer tipi değişkenlerden birisi değildir?
  - a) Tamsayı Değişken Tipleri
  - b) Ondalıklı Sayı Değişken Tipleri
  - c) Devirli Sayı Değişken Tipleri
  - d) Bool Değişken Tipi
  - e) Char Değişken Tipi

6.

```
... . = ..  
int y = 10;  
Console.WriteLine(x + y);
```

Yukarıdaki program çıktısında ekranda “15” görülebilmesi için boş bırakılan yere yazılması gereken en uygun değişken tanımı aşağıdakilerden hangisidir?

- a) int x = 5;
  - b) int x = 5
  - c) int x = -5;
  - d) var x = 5
  - e) object x = 5;
7. Aşağıdakilerden hangisi tam sayı değişken tiplerinden biri değildir?
- a) int
  - b) long
  - c) byte
  - d) nint
  - e) double
8. Aşağıdakilerden hangisi C# programlama dilinde bulunan tip dönüşümlerinden biri değildir?
- a) Örtülü Dönüşüm
  - b) Açık Dönüşüm
  - c) Kullanıcı Tanımlı Dönüşüm
  - d) Tanımsız Dönüşüm
  - e) Yardımcı Sınıflarla Dönüşüm
9. Aşağıdakilerden hangisi Açık Dönüşüm işleminde dönüşüm sırasında veri tipleri arasında uyumsuzluk olması halinde çalışma zamanında ortaya çıkan hatadır?
- a) ArgumentException
  - b) InvalidCastException
  - c) FormatException
  - d) InvalidOperationException
  - e) NotSupportedException
10. Aşağıdakilerden hangisi Açık Dönüşüm işleminde kullanılan manuel yöntemlerden biri değildir?
- a) Parse
  - b) TryParse
  - c) Convert
  - d) Casting
  - e) CompareTo

**Cevap Anahtarı**

1.b, 2.e, 3.c, 4.a, 5.c, 6.a, 7.e, 8.d, 9.b, 10.e

## YARARLANILAN KAYNAKLAR

Aktaş, Volkan. 2021. *Her Yönüyle C# 9.0*.

<https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/06/c'-ta-de%C4%9Fer-ve-referans-tipleri>. 2013. "C# ta Değer ve Referans Tipleri". Tarihinde 10 Eylül 2021 (<https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/06/c'-ta-değer-ve-referans-tipleri>).

<https://www.azkod.com/>. 2020. "C# Değişkenler". Tarihinde 09 Ağustos 2021 (<https://www.azkod.com/csharp/c-degiskenler>).

Karaçay, Timur, ve Aybar Karaçay. 2008. *C # ile Nesne Programlama*. Seçkin Yayıncılık.

Okumuş, Hikmet. 2012. "C# İle Var Anahtar Kullanımı". Tarihinde 11 Eylül 2021 (<https://www.hikmetokumus.com/makale/40/csharp-ile-var-anahtar-kullanimi>).

Wanger, Bill. 2020. "Casting and type conversions (C# Programming Guide)". Tarihinde 11 Eylül 2021 (<https://docs.microsoft.com/en-gb/dotnet/csharp/programming-guide/types/casting-and-type-conversions>).

Wanger, Bill, Rick Anderson, Petr Kulikov, Gene Milener, ve Andy De George. 2020. "Value types (C# reference)". Tarihinde 12 Eylül 2021 (<https://docs.microsoft.com/en-gb/dotnet/csharp/language-reference/builtin-types/value-types>).