

Bu dokümanda yazılımın temeline giriş yapılmıştır. Değişkenler ve operatörler her programlama dilinin temelidir. Ayrıca bu dokümanda her karanlık nokta aydınlatılmamıştır. Sizlerden bu karanlık noktaların araştırılıp aydınlatılması beklenmektedir.

Değişkenler ve Operatörler

İlkel ve değer tipli değişkenler

Oğuzhan Karagüzel 24.11.2024

İçindekiler Tablosu

| | |
|--|----|
| ŞEKİLLER DİZİNİ..... | 2 |
| TABLolar DİZİNİ..... | 3 |
| DEĞİŞKENLER VE OPERATÖRLER..... | 4 |
| DEĞİŞKENLER..... | 5 |
| C#'ta Değişken Tipleri, Boyutları ve Nitelikleri | 6 |
| Değişken Adları (Adres Takma Adı) | 8 |
| OPERATÖRLER..... | 13 |
| C# Programlama Dilinde Operatörler | 13 |
| OPERATÖRLER İLE İLGİLİ BİLİNMESİ GEREKENLER | 18 |
| DEĞİKEN ADLANDIRMA, ATAMA VE FORMATLAMA..... | 20 |
| Adlandırma | 21 |
| Veriyi Formatlama | 21 |
| ADRESLERİN SIRALI OLMASI ÜZERİNE AÇIKLAMA | 23 |
| ÖDEV..... | 26 |
| Kaynakça | 27 |
| TEST | 28 |
| Cevap Anahtarı | 31 |

ŞEKİLLER DİZİNİ

| | |
|--|----|
| Şekil 1 Değişken tanımlama ve değer atama | 5 |
| Şekil 2 Veri tipleri ve kapladığı alanlar için oluşturulan kod bloğu | 8 |
| Şekil 3 Veri tipleri ve kapladığı alanlar için oluşturulan kod bloğu çıktısı | 8 |
| Şekil 4 Değişken Tanımlama ve adlandırma..... | 9 |
| Şekil 5 Değişkenin adresini bulma | 9 |
| Şekil 6 Değişken adresini bulma programının çıktısı..... | 11 |
| Şekil 7 Değişken adresinin 2. çalıştırmada çıktısı..... | 12 |
| Şekil 8 Değişkene değer atama operatörü ve değişkeni çağırma işlemi | 19 |
| Şekil 9 İşlem sırası örneği..... | 19 |
| Şekil 10 İşlem Sırası Örneği 2 | 20 |
| Şekil 11 İşlem sırası örneği 2 programın çıktısı | 20 |
| Şekil 12 Verinin çeşitli formatları | 22 |
| Şekil 13 Veri formatla programı çıktısı | 23 |
| Şekil 14 İnt Tipi Adresi ile her hücreye teker teker ulaşma programı çıktısı..... | 25 |

TABLolar DİZİNİ

| | |
|---|---|
| Tablo 1C#’ta ilkel veriler ve nitelikleri | 6 |
|---|---|

Öğuzhan KARAGÜZEL

DEĞİŞKENLER VE OPERATÖRLER

Eğer yazılımın yanında geçtiyseniz bu iki terimi kesinlikle duymuşsunuzdur. Kısaca şöyle tanımlanır;

DEĞİŞKEN : programlama dillerinde bir veri değerini depolayan isimlendirilmiş bir bellek konumudur. Değişkenler, veri türüne göre farklı değerleri tutabilirler ve bu değerler programın çalışması sırasında değiştirilebilir.

OPERATÖR : programlama dillerinde, değişkenler ve sabitler üzerinde işlemler gerçekleştiren sembollerdir. Operatörler, matematiksel, karşılaştırma, mantıksal ve bit düzeyinde işlemleri ifade ederler.

Hadi bu konuyu biraz daha detaylandıralım. Bir önceki bölümde bilgisayarda hafızadan bahsetmiştik (KARAGÜZEL, Bilgisayar ve Hafıza, 2024). o dokümanda da anlatıldığı üzere; Bizim yazdığımız programda ram'i kullanacak. Ram'e veri kaydedeceğiz ve ram'den veri okuyacağız. Peki nasıl kaydedeceğiz ve okuyacağız?

Kısaca hatırlayalım. Ram'de 8 bit'lik (1 byte) hafıza hücrelere bulunmakta. Bu bizim en küçük hafıza birimimizdir. Her bir hücrenin adresi bulunmaktadır. İşlemci bu adres bilgisi ile bu hücreye ulaşır. Buradan bilgiyi okur ya da buraya bilgi yazar. Buradaki bilgiler, her zaman, fiziksel olarak 1 volt ya da 0 volt olarak tutulmaktadır. Bu bilgiyi kullanarak bunları ikilik sistemde 1 ve 0 olarak kullanmaktayız. İşlemci içerisinde karmaşık yapıya sahip elektronik devreler ile bu bilgileri kullanarak çeşitli işlemler yapmaktayız. Herhangi birbirinden farklı iki ayrı hücreye ulaşmanın, işlemci için bir farkı olmadığından, bu yapıya Random Access memory (rastgele erişilebilir bellek) denilmektedir.

Buradan anlaşılıyor ki bizde adresleri kullanıp Ram'e veri yazacağız. Ardından yine adresleri kullanıp Ram'den veri okuyacağız. Adresleri anladık. Peki her zaman 1 ya da 0 olan bu verilerin ne anlama geldiğini nasıl anlayacağız. Ya da bu verinin 28 gibi bir sayı ya da "c" gibi bir harf olup olmadığını nasıl anlayacağız. Bu seferde ram'e kaydedilen verinin ya da ram'den okunmak istenen verinin ne olduğunu bilmemiz gerekiyor. Eğer sayı tabanlarını biliyorsanız ya da hazırlamış olduğum ek içeriği (KARAGÜZEL, Sayı Tabanları, 2024) incelediyseniz ikilik tabanda "1001" sayısının 9'a denk geldiğini biliyorsunuzdur. Ram içerisinde bir hücrede bu bilgiyi "0v-0v-0v-0v-1v-0v-0v-1v" şeklinde saklayabiliriz. Bu bilgiye ikilik sistemdeki bir sayı gibi de (işlemci içerisindeki karmaşık elektronik devreler sayesinde) davranabiliriz. Toplama veya çarpma gibi aritmetik işlemler yapabiliriz. Ancak belirli bir an belirli bir hücreye kaydedilmiş verini aslında ne anlama geldiğini (sayı - karakter) nasıl bilebiliriz. "00001001" 9 sayısı mı? Yoksa bambaşka bir şey mi? Demek ki ram'e kaydettiğimiz bilginin niteliğini de elimizde tutmamız ya da bu bilgiyi de bir kenarda saklamamız gerekiyor.

Tamam. Şunu anladık: bir adresi kullanarak hücreyi belirle. Bu hücreye kaydedilecek ya da buradan okunacak verinin niteliğinin ne olacağını belirle. İşte bu ikisinin oluşturduğu yapıya tam olarak "Değişken" diyoruz. Değişkenleri anladığınızı düşünüyorum. Peki bu değişkene, yani ram'deki bir hücreye nasıl veri yazacağız? Veriyi nasıl okuyacağız? Bu veriler arasında nasıl işlem yapacağız? İki tane sayıyı tutan iki tane değişkenin değerlerini nasıl toplayacağız? İşte burada bu işlemleri yapabilmek için "Operatör" leri kullanacağız. Operatörlerde tam olarak bu işe yarar. Örnek olarak "+" operatörü matematikte iki sayıyı toplamaya yarar. Hemen hemen bütün programlama dillerinde de aynı şekilde iki sayıyı toplamaya yaramaktadır. Bunun dışında da değişkenin niteliğine bağlı olarak "+" operatörü farklı davranışlar gösterir. Ya da nasıl bir davranış göstereceğini kodlayabiliriz. O zaman yapacağımız işi kısaca özetleyelim;

1. Saklamak istenilen verinin tipi (niteliği) belirlenir.
2. Saklamak istenilen veri için bir adres belirlenir.
3. Saklanan veri üzerinde operatörler kullanılarak işlemler yapılır.

Ne yapacağımızı anladık. Peki nasıl yapacağız. Hadi hepsini birden yapıp ardından teker teker inceleyelim.

```
static void Main(string[] args)
{
    int sayi = 10;
}
```

Şekil 1 Değişken tanımlama ve değer atama

Şekil 1’de görüldüğü üzere bir değişken tanımladık ve ona değer atadık. Kısaca açıklayalım. Ardından her bir başlığa derinlemesine bakalım;

İnt

Saklamak istediğimiz verinin tipini belirledik. Bu değişkenin bir tam sayı (integer - int) olduğunu söyledik.

Sayi

Saklamak istediğimiz değişkenin adresini belirledik;

=

İle 10 sayısını değişkene atadık

İşte değişken tanımlamak ve değer atamak bu kadar basittir. Değişkenleri bu kadar basit bir biçimde kullanabilirsiniz. Ancak siz yine de gelin beraber bu işlemin arka planına bir bakış atalım. Değişkenlerin doğasını ne kadar iyi bilerseniz o kadar ustaca kullanırsınız ve verimli program yazarsınız.

DEĞİŞKENLER

Değişkenler bakmamız gereken iki özellik vardır. Birincisi değişkenin tipi. İkincisi değişkenin adı. Bu konulara dalmadan önce kısa bir açıklamada bulunayım.

Phyton ve javascript gibi bazı dillerde değişkenin tipini belirtmenize gerek yoktur. Bu değişkenin tipi otomatik olarak algılanır ve atama yapılır. C# ya da C++ gibi dillerde ise değişkenin tipini belirtmek zorundasınız. Tip belirtmek gerekmeyen dillere “dinamik tür tanımlı diller” (dynamically typed languages), Tip belirtmek gereken dillere “statik tür tanımlı diller” (statically typed languages) denir.

Bir önceki dersten hatırlayacağınız üzere (KARAGÜZEL, Bilgisayar ve Hafıza, 2024), ram’deki bir hücreyi adresi ile kullanıyoruz. Bu adresler ise ikilik tabanda bir sayıdır. Biz ise onaltılık sistemde adres bilgilerini kullanıyorduk. Ancak biz değişkene adres vermedik. Sadece ad verdik. Verdiğimiz bu ad doğrudan o adresi göstermektedir. Aslında ikilik tabandaki o sayıya bir ad veriyoruz. Bu sayede o adresi tekrar tekrar uzun uzadıya yazmak zorunda kalmıyoruz. Sadece kendi yazdığımız adı kullanıyoruz.

C#'ta Değişken Tipleri, Boyutları ve Nitelikleri

Tablo 1C#'ta ilkel veriler ve nitelikleri

| Veri Türü | .NET Türü | Açıklama | Boyut (byte) | Değer Aralığı |
|----------------|----------------|---|----------------|---|
| bool | System.Boolean | Doğru/yanlış (true/false) | 1 | true veya false (1 veya 0) |
| byte | System.Byte | İşaretsiz 8 bit tamsayı | 1 | 0, 255 |
| sbyte | System.SByte | İşaretli 8 bit tamsayı | 1 | -128, 127 |
| char | System.Char | Unicode karakter | 2 | U+0000, U+FFFF |
| short | System.Int16 | İşaretli 16 bit tamsayı | 2 | -32,768, 32,767 |
| ushort | System.UInt16 | İşaretsiz 16 bit tamsayı | 2 | 0, 65,535 |
| int | System.Int32 | İşaretli 32 bit tamsayı | 4 | -2,147,483,648, 2,147,483,647 |
| uint | System.UInt32 | İşaretsiz 32 bit tamsayı | 4 | 0, 4,294,967,295 |
| long | System.Int64 | İşaretli 64 bit tamsayı | 8 | -9,223,372,036,854,775,808, 9,223,372,036,854,775,807 |
| ulong | System.UInt64 | İşaretsiz 64 bit tamsayı | 8 | 0, 18,446,744,073,709,551,615 |
| float | System.Single | 32 bit kayan noktalı sayı | 4 | $\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ |
| double | System.Double | 64 bit kayan noktalı sayı | 8 | $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ |
| decimal | System.Decimal | 128 bit hassas kayan noktalı sayı | 16 | $\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$ |
| string | System.String | Unicode karakter dizisi (referans türü) | Boyut değişken | Bağımlı (dizideki karakter sayısına bağlı) |

Tablo 1'de C# programlama dilinde kullanacağımız ilkel veri tipleri ve bunların çeşitli nitelikleri verilmiştir.

Bir Hücrenin 8 bit veya 1 byte olduğunu söylemiştik. Tablo 1'den anlaşılıyor ki örnek olarak int veri tipi 4 hücrelik alan kaplamaktadır. Yani toplamda 32 bit. Maximum ve minimum değerlerine bakıldığında -2,147,483,648, 2,147,483,647 arasında olduğu görülür.

Buradaki ilginç durumu kısaca inceleyelim. Şimdi bizim verilerimiz bir hücrede 1v ya da 0v olarak tutulmakta. 4 Hücreden 32 bit demek. Yani 32 bit. Bu pillerin her biri ya 1v ya da 0v gerilime sahip. Bu değerlere ikilik sayı tabanındaymışız gibi davranıyoruz. Bu değerin ne olduğunu programa ya da

Gerilim değerlerini unutalım ve ram’de sadece ikilik tabanda sayıların olduğunu düşünelim. Bunlara doğrudan bit diyelim. Bir tam sayı yani “int” pozitif ve negatif değerler alabilmektedir. Peki sayının pozitif ve negatif olduğuna nasıl karar verilir. Hadi şu bitlere bakalım.

İŞARET BİT'İ : En baştaki bit ya da 32. bit işaret bitidir ve işaretli sayılarda sayının pozitif ya da negatif olduğunu belirtir. “uint” ya da “ulong” gibi işaretsiz sayılarda ise bu bit artık işaret biti yerine doğrudan sayı olarak davranılır. Anlayacağınız üzere hücrede sadece bitler bulunur. Biz onları veri tiplerimiz ile işaretleyerek anlamlandırırız.

SAYIMIZ İÇİN EN ANLAMLİ BİT : İkili tabanda yukarıdaki sayıya baktığımızda 8 olduğu hemen görülür. Sayının solunda bulunan 0'lar gereksizdir. Ancak bilgisayar için bu durum böyle değildir. Sayımız 8 olduğundan bize sadece 4 bitlik bir alan yeterli olacaktır. Ama bilgisayara bunu yaptıramayız. Ona belirli bir alana bize ayırması gerektiğini söylemeliyiz. O alanı da değiştiremeyiz. Ayrıca en küçük hafıza birimimiz bir hücredir ve 8 bitten oluşmaktadır. 8 bitten daha küçük veriyi saklayamayız.

Anlayacağımız en küçük alanı kaplayan verimiz en az 8 bitlik yani 1 byte'lık bir alan kaplayabilir. Yukarıdaki örnekte "int" veri tipini kullandığımızdan 32 bitlik alan ayrılmış olacaktır. Eğer "long" kullanmış olsaydık bu sefer 64 bitlik alan ayrılmış olacaktır. 8 sayısını long olarak da saklasak int olarak da sadece 4 bitlik alanı etkin bir biçimde kullanmış olacağız. Bundan dolayı sayımız için en anlamlı bit değeri en yüksek değere sahip 1 olan bittir.

İşaretsiz sayılarda “u” harfini kullandığımızı fark etmişsinizdir. Bu unsigned’in kısaltması olarak kullanılır. Uint : unsigned integer demektir.

Bu konuda son olarak “bool” ya da “Boolean” veri tipine değinmek gerekir. Bu veri tipi, modern matematiksel mantığın babası ve bilgisayarın temelini atmış olan, “George Boole” a ithafen “Boolean” ya da kısaca “bool” olarak adlandırılmıştır. Buy veri tipinde sadece doğru, “true”, 1 ya da yanlış, “false”, 0 değeri saklanabilir. 1 byte’lık bir alan kaplamasına rağmen sadece 1 bit etkin olarak kullanılır.

7


```

class Program
{
    unsafe static void Main(string[] args)
    {
        Console.WriteLine($"bool      : {Unsafe.SizeOf<bool>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<bool>() * 8} bit değere eşittir.");
        Console.WriteLine($"byte      : {Unsafe.SizeOf<byte>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<byte>() * 8} bit değere eşittir.");
        Console.WriteLine($"sbyte     : {Unsafe.SizeOf<sbyte>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<sbyte>() * 8} bit değere eşittir.");
        Console.WriteLine($"char      : {Unsafe.SizeOf<char>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<char>() * 8} bit değere eşittir.");
        Console.WriteLine($"short     : {Unsafe.SizeOf<short>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<short>() * 8} bit değere eşittir.");
        Console.WriteLine($"ushort    : {Unsafe.SizeOf<ushort>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<ushort>() * 8} bit değere eşittir.");
        Console.WriteLine($"int       : {Unsafe.SizeOf<int>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<int>() * 8} bit değere eşittir.");
        Console.WriteLine($"uint      : {Unsafe.SizeOf<uint>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<uint>() * 8} bit değere eşittir.");
        Console.WriteLine($"long      : {Unsafe.SizeOf<long>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<long>() * 8} bit değere eşittir.");
        Console.WriteLine($"ulong     : {Unsafe.SizeOf<ulong>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<ulong>() * 8} bit değere eşittir.");
        Console.WriteLine($"float     : {Unsafe.SizeOf<float>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<float>() * 8} bit değere eşittir.");
        Console.WriteLine($"double    : {Unsafe.SizeOf<double>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<double>() * 8} bit değere eşittir.");
        Console.WriteLine($"decimal  : {Unsafe.SizeOf<decimal>()} byte (Ya da birim hücre) yer kaplar. {Unsafe.SizeOf<decimal>() * 8} bit değere eşittir.");
        Console.ReadLine();
    }
}

```

Şekil 2 Veri tipleri ve kapladığı alanlar için oluşturulan kod bloğu

Yukarıdaki programda tüm ilkel veri tipleri ve onların bellekte kapladığı alanlar yazdırılmıştır. Çıktıyı incelersek;

```

C:\Users\Oguzh\OneDrive\Mz x + v
bool      : 1 byte (Ya da birim hücre) yer kaplar. 8 bit değere eşittir.
byte      : 1 byte (Ya da birim hücre) yer kaplar. 8 bit değere eşittir.
sbyte     : 1 byte (Ya da birim hücre) yer kaplar. 8 bit değere eşittir.
char      : 2 byte (Ya da birim hücre) yer kaplar. 16 bit değere eşittir.
short     : 2 byte (Ya da birim hücre) yer kaplar. 16 bit değere eşittir.
ushort    : 2 byte (Ya da birim hücre) yer kaplar. 16 bit değere eşittir.
int       : 4 byte (Ya da birim hücre) yer kaplar. 32 bit değere eşittir.
uint      : 4 byte (Ya da birim hücre) yer kaplar. 32 bit değere eşittir.
long      : 8 byte (Ya da birim hücre) yer kaplar. 64 bit değere eşittir.
ulong     : 8 byte (Ya da birim hücre) yer kaplar. 64 bit değere eşittir.
float     : 4 byte (Ya da birim hücre) yer kaplar. 32 bit değere eşittir.
double    : 8 byte (Ya da birim hücre) yer kaplar. 64 bit değere eşittir.
decimal   : 16 byte (Ya da birim hücre) yer kaplar. 128 bit değere eşittir.

```

Şekil 3 Veri tipleri ve kapladığı alanlar için oluşturulan kod bloğu çıktısı

Görüldüğü üzere boole veri tipi 1 byte'lık alan kaplamaktadır. Buradan anlamamız gereken en küçük hafıza birimimiz byte'tır. Bit değildir. Bundan dolayı byte oldukça önemlidir.

Bu konuyu anladığınızı düşünerek diğer başlığa geçiyorum. Eğer burada merak ettiğiniz şeyler var ise lütfen internetten araştırın veya yapay zeka ile sohbet edin.

Değişken Adları (Adres Takma Adı)

Şekil 1'de ki int kavramını öğrenmiş olduk. Ram'de bir alan ayıracağız ve bu alan int değeri için 4 hücreden yani 4 byte'dan oluşacak. İşlemci bu hücrelere ulaşmak için bu hücrelerin adreslerini kullanacaktır. Bunu bir önceki konuda (derste) işlemiştik. Hangi adresler olduğunu işlemciye söylememiz gerekiyor. Ayrıca bir tane değil tam tamına 4 tane adrese ihtiyacımız var. Çünkü int değişkeni 4 hücrelik alan kaplıyor.

Peki bunu nasıl yapacağız?

```
static void Main(string[] args)
{
    int sayi;
}
```

Şekil 4 Değişken Tanımlama ve adlandırma

Şekil 4’te olduğu gibi biz sadece ad vereceğiz. Başka hiçbir şey yapmayacağız. Orta ve yüksek seviyeli dillerin en önemli özelliklerinden biri de budur. Sizin ikilik ya da onaltılık tabanda bir veya birden fazla sayı yazmanıza gerek yoktur. Bellek adresleri arka planda otomatik olarak oluşturulur ve sizin programınızda verilen ad ile temsil edilir. Yani şekil 1 ve şekil 4’teki değişkenlerde “sayi” olarak adlandırdığımız değişken aslında adresi temsil etmektedir. Sizin adresle uğraşmanıza gerek yoktur. Bu işletim sistemi tarafından otomatik olarak yapılır.

Şimdi bunu görmek için kısa bir kod yazalım. Tavsiyem bu kodları siz yazmayın ve denemeyin. Nedenine daha sonra değineceğim.

```
unsafe static void Main(string[] args)
{
    int sayi;

    int* ptr = &sayi;

    // Adres değerini alıyoruz
    ulong address = (ulong)ptr;

    // Adresi onluk tabanda yazdırıyoruz
    Console.WriteLine("Adres (Onluk): " + address);

    // Adresi onaltılık tabanda yazdırıyoruz
    Console.WriteLine("Adres (Onaltılık): 0x{0:X}", address);

    // Adresi ikilik tabanda yazdırıyoruz
    string binaryAddress = Convert.ToString((long)address, 2).PadLeft(IntPtr.Size * 8, '0');
    Console.WriteLine("Adres (İkilik): " + binaryAddress);

    Console.Read();
}
```

Şekil 5 Değişkenin adresini bulma

İlk olarak bu kod bloğunun ne yaptığına kısaca bir bakalım. Şu an ki seviyenizi oldukça aşan bir kod bloğu. Satır satır açıklayalım.

`int sayi;`

İlk olarak bir değişken tanımlıyoruz. Bu değişkenin türü int. Yani bellekte 4 hücrelik yer kaplayacak. Sonra bu hücrelerden bir tanesinin adresi “sayi” dediğimiz değişken adı ile temsil edilecek. “Sayi” dediğimiz değişkene ulaşmaya çalıştığımızda bu adresteki değere ulaşmaya çalışacağız.

Soru: int değişkeni 4 hücre demek. 4 hücre ise 4 adres demek. Biz bu adresi “sayi” ile temsil edeceğiz. Peki bu 4 adresin 4’ü de sayi değişkeni ile mi temsil edilecek? Yoksa sadece 1 tanesi mi?

Cevap: sadece 1 tanesi temsil edilecek.

Soru: peki hangi hücrenin adresi temsil edilecek?

Cevap: Bu sistemden sisteme deđişmektedir. Ancak ya en anlamlı ya da en anlamsız hücrenin adresi temsil edilir. Bununla alakalı kısa bir açıklama;

Bellek Düzeni (Endianness)

Little-Endian ve Big-Endian Nedir?

- **Little-Endian:**
 - En düşük anlamlı byte (Least Significant Byte - LSB) en düşük bellek adresinde saklanır.
 - Intel işlemciler ve çođu modern PC bu düzeni kullanır.
- **Big-Endian:**
 - En yüksek anlamlı byte (Most Significant Byte - MSB) en düşük bellek adresinde saklanır.
 - Bazı ağ protokolleri ve eski sistemler bu düzeni kullanır.

Örnek:

- **Sayı:** 0x12345678 (hexadecimal olarak)
- **Little-Endian Bellek Düzeni:**

| Adres | Byte Deđeri |
|-------|-------------|
| 0 | 0x78 |
| 1 | 0x56 |
| 2 | 0x34 |
| 3 | 0x12 |

- **Big-Endian Bellek Düzeni:**

| Adres | Byte Deđeri |
|-------|-------------|
| 0 | 0x12 |
| 1 | 0x34 |
| 2 | 0x56 |
| 3 | 0x78 |

Not: Çođu modern PC ve sunucu **little-endian** düzeni kullanır.

Soru: Daha önceki başlıkta öğrenmiştik. İkilik tabanda en sağda bulunan hane deđeri en küçük sayılar. En anlamsız byte (en anlamsız 8 bit). En solda bulunan en anlamlı byte (en anlamlı 8 bit).

11111111 11111111 11111111 11111111

Kırmızı en anlamsız. Mavi en anlamlı.

Burada en anlamsız olan (kırmızı) bitler'in bulunduğu hücrenin adresi (1 byte'ın adresi) "sayı" ile temsil edilecek. Bu durumda bizler "sayı" değişkenine ulaşmaya çalıştığımızda en anlamlı 8 bit'e nasıl ulaşacağız?

Cevap: İşte burada değişkenin türü devreye giriyor. Değişkenin türü "int" olduğu için "sayı" ile temsil edilen adres ile bu sayının tamamına ulaşılır. Örnek olarak "sayı" ile "0x72" (0x onaltılık taban demektir.) adresi tutulsun. Bu adreste en anlamsız byte bulunmaktadır. "int" değişkeni olduğu için 4 hücre olduğu bilinir. "sayı" daki adres ile ilk hücreye ulaşılır. Bu adrese bir eklenerek ikinci hücreye. Bir daha eklenerek 3. Hücreye. Bir daha eklenerek 4. Ve son hücreye ulaşılır. Bu sayede sayının tamamına ulaşılır. Bu konuya bu doküman içerisinde daha sonra detaylı bir örnek verilecektir.

Soru: "Bilgisayarda hafıza" adlı derste adreslerden hücrelere ulaşırken adreslere 8 ekleyerek ilerlemiştik. Burada ise sadece 1 ekleyerek ilerliyoruz. Farkı nedir?

Cevap: Bunun sebebi "Bilgisayarda hafıza" adlı derste sizlere "Çoklayıcı" devresini anlatmamış olmamdır. Bu elektronik bir devre olduğundan konu dışına çıkmaktadır. Ancak günümüzde bilgisayarlar bu adreslere 1 ekleyerek hücrelere ulaşmaktadır.

Şimdi kodlara kaldığımız yerden devam edelim. "int sayı" ile tanımlanmış değişkenin adresini bulalım. Belki bazılarınız "pointer" ları duymuştur. Bu konuya girmeyeceğim. "int* ptr = &sayı;" ile "sayı" değişkeninin adresine ulaşırız.

Ardından "ulong address = (ulong)ptr;" ile adresi bir değikene atıyoruz. Ardından "address" değişkeninin sakladığı bilgiyi çeşitli formatlarda ekrana yazdırıyoruz.

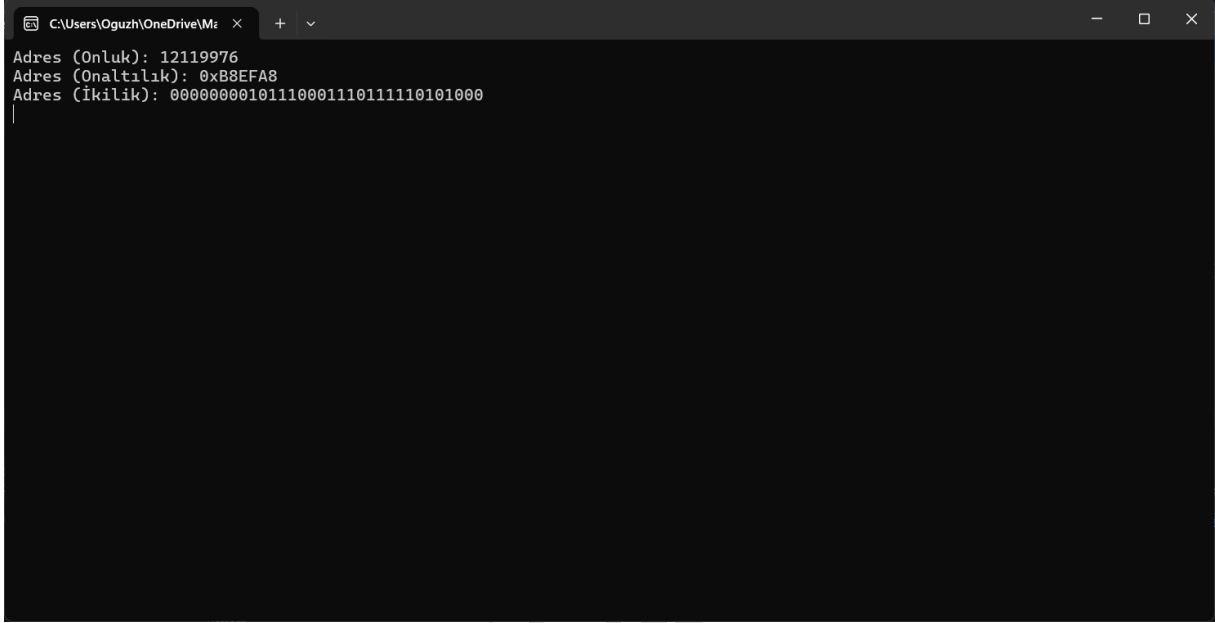
Bu programı çalıştırdığımızda çıktı şu şekildedir;

```
C:\Users\Oguzh\OneDrive\Mz x + v
Adres (Onluk): 5829688
Adres (Onaltılık): 0x58F438
Adres (İkilik): 000000001011000111010000111000
```

Şekil 6 Değişken adresini bulma programının çıktısı

Soru: bu adres nasıl atandı? Bu adresi biz yazmadık. Sadece "sayı" dedik ama adres belirtecek bir sayı yazmadık.

Cevap: İşletim sistemi bunu sizin yerinize otomatik olarak yapar. Ram’i yöneten işletim sistemidir. Bilgisayarınızı ve hafızasını yöneten işletim sistemidir. Siz “int sayı” diyerek işletim sistemine “Ram’de bana bir alan aç. Bu alanın adresini bana ver.” demiş oluyorsunuz. Programı bir daha çalıştırdığınızda bu değişikliği göreceksiniz. İlk çalıştırdığınızda elde ettiğiniz adres ile ikinci çalıştırdığınızda elde ettiğiniz adres farklı olacaktır. Hatta neredeyse her çalıştırdığınızda elde ettiğiniz adres bir öncekinden farklı olacaktır. İşletim sistemi o an uygun olan rastgele bir hücre adresini size verecektir. Hadi programı bir daha çalıştırarak görelim.



```
C:\Users\Oguzh\OneDrive\Mz > + v
Adres (Onluk): 12119976
Adres (Onaltılık): 0xB8EFA8
Adres (İkilik): 00000000101110001110111110101000
```

Şekil 7 Değişken adresinin 2. çalıştırmada çıktısı.

Gördüğünüz üzere ilk çıktımızda ki adres ile ikinci çalıştırmamızda elde ettiğimiz adres farklıdır.

Neden bu şekilde program yazmamalısınız ya da bu program üzerinde işlem yapmanızı neden önermiyorum.

Ram’de herhangi bir alana bu adresler ile ulaşabilirsiniz. Tehlike olan kısımda budur. Ulaştığınız adresteki bilgi başka bir programın çalışması için gerekli olan bir bilgi olabilir. O program ile aynı anda o bilgiye ulaşmaya çalıştığınızda ya da değiştirmeye kalktığınızda en hafifinden sadece o program çökecektir. Yüksek ihtimalle yazacağınız kod düzgün çalışmayacaktır. Çünkü işletim sistemi bu işlemi yapmanıza izin vermeyecektir. Ancak bunu aşmanın yolları bulunmaktadır. İsterseniz bunu yapabilirsiniz.

Oyunlarda kullanılan hileler bu şekilde çalışmaktadır. Daha doğrusu oyun içi hileler değil, dışarıdan indirilen ayrı hile programları bu şekilde çalışmaktadır. Hile için üretilmiş program başlı başına ayrı bir programdır. Bu program hile yapılması istenilen hedef oyunun ram’de tuttuğu alanları bulur. Buradaki değerleri değiştirir. Bu sayede oyunda çok fazla paranız olabilir ya da ölümsüzlük elde edebilirsiniz?

2024 yılında bütün bilgisayarların çöktüğünü. Uçuşların iptal edildiğini ya da bankacılık sitelerinin çalışmadığını hatırlıyor olmalısınız. 1 günlük bir süreçti. “CrowdStrike” adlı bir güvenlik uygulaması yeni bir güncelleme yayınladı. Bu güncellemede bir sorun vardı. Bu güvenlik programı tüm ram’i tarayarak tehlike arz eden bir veri olup olmadığını kontrol ediyordu. Ancak yayınlanan güncellemede ram’de olmayan ram’in sınırını aşan bir alan kontrol ediliyordu. Örnek olarak ram 48

byte ise 49. Byte kontrol edilmeye çalışılıyordu. Bundan dolayı işletim sistemi çöküyordu. Bundan dolayı bu şekilde kod yazmak oldukça tehlikelidir.

Bu güvenli olmayan (unsafe) kodlarla biraz daha devam edeceğiz. Ancak bu kodlara hiç ihtiyaç duymayacağız. Genel kültür olarak kalması bizim için en iyisi.

Artık değişkenleri öğrenmiş olduğunuzu varsayıyorum. Kısaca özet geçelim;

Ram'de her zaman bilgiler 1 volt ve 0 volt olarak saklanır. Biz bu gerilim değerlerini ve karmaşık elektronik devrelerini kullanarak ikilik tabanda bir sayıymış gibi işlem yaparız. Yani ram'de her zaman ikilik tabanda sayı saklanır diye kabul edebiliriz.

Burada saklayacağımız verinin kaç hücreyi kapsayacağınız ve bu sayılara nasıl davranılacağını değişkenin türü ile belirleriz. Örnek olarak "int" için 4 hücre kullanılır. 32. Bit'e işaret bit'i olarak davranılır. "char" için ise 2 hücre kullanılır ve bu sayıya karakter gibi davranılır. "bool" için önerme olarak davranılır. "bool" 1 byte alan kaplasa bile içinde bulunan 8 bitten sadece bir tanesi işleme alınır. Geriye kalan 7 bit görmezden gelinir.

Eğer değişkenimiz 1 byte'dan daha fazla alan kaplıyor ise en anlamsız byte'ın adres değeri bize geri döndürülür. İşletim sistemi tarafından bu byte'lar ram'de sıralı bir biçimde kaydedilir. En düşük adres değeri ya da en anlamsız byte'ın adresi bize verilir. Bizde bu adrese 1 ekleyerek sonraki byte'lara ulaşırız. Bu byte'lara ulaşma işlemini de biz yapmayız. Bu da bizim için otomatik olarak yapılır. Yani biz "sayı" değişkenine bir sayı atarığımızda tekrardan "sayı" değişkenini çağırarak sayıyı olduğu gibi geri çağırılmış oluruz.

Artık operatörlere geçebiliriz.

OPERATÖRLER

Değişkenlerden anladığınız üzere bunlar bizim saklamak, manipüle etmek ya da aritmetik işlem yapmak istediğimiz verilerdir. Bu veriler üzerinde bu gibi çeşitli işlemleri yapmak için operatörleri kullanırız. Operatörler çok çeşitlidir ve birbirinden bağımsız oldukça fazla görevi bulunmaktadır. Eğitim serisi boyunca çeşitli operatörleri sürekli olarak kullanacağız. Operatörleri kullanırken bilinmesi gereken birkaç kural bulunmaktadır. Bu kurallara dikkat ettikten sonra operatör ve değişkenleri kullanarak artık basit program yazmaya başlayabilirsiniz. Operatörlere kısaca bir bakalım.

C# Programlama Dilinde Operatörler

C# dilinde operatörler, değişkenler ve değerler üzerinde işlemler yapmamızı sağlayan sembollerdir. Operatörleri kategorilere ayırarak inceleyelim.

1. Aritmetik Operatörler

Aritmetik operatörler, sayısal değerler üzerinde matematiksel işlemler yapmamızı sağlar.

| Operatör | Açıklama | Örnek Kullanım | Sonuç |
|----------|-----------------|----------------|-----------|
| + | Toplama | 5 + 3 | 8 |
| - | Çıkarma | 5 - 3 | 2 |
| * | Çarpma | 5 * 3 | 15 |
| / | Bölme | 10 / 2 | 5 |
| % | Modül (Kalan) | 10 % 3 | 1 |
| ++ | Artırma (Ön/Ek) | a++ veya ++a | a = a + 1 |
| -- | Azaltma (Ön/Ek) | a-- veya --a | a = a - 1 |

Not: ++ ve -- operatörleri, değişkenin değerini bir artırır veya azaltır. Ön ek (++a) veya son ek (a++) olarak kullanıma göre işlem sırası değişir. Eğer ++a olarak kullanırsanız ilk olarak değişkenin değeri 1 arttırılır. Ardından değişkenin değerine ulaşılır. Eğer a++ olarak kullanırsanız ilk olarak değışkene ulaşılır. Ardından değeri 1 arttırılır.

2. Atama Operatörleri

Atama operatörleri, bir değışkene değeri atamak veya mevcut değeri güncellemek için kullanılır.

| Operatör | Açıklama | Örnek | Eşdeğer Uzun Yazımı |
|----------|--------------------|--------|---------------------|
| = | Atama | a = 5 | a = 5 |
| += | Toplayarak atama | a += 3 | a = a + 3 |
| -= | Çıkararak atama | a -= 2 | a = a - 2 |
| *= | Çarparak atama | a *= 4 | a = a * 4 |
| /= | Bölerek atama | a /= 2 | a = a / 2 |
| %= | Modül olarak atama | a %= 3 | a = a % 3 |

a = a+3 ifadesi şu anda kafanızı karıştıracaktır. Bu bir sonraki başlıkta değineceğiz.

3. Karşılaştırma Operatörleri

Karşılaştırma operatörleri, iki değeri karşılaştırarak sonuç olarak true veya false döndürür.

| Operatör | Açıklama | Örnek | Sonuç |
|----------|---------------------|--------|-------|
| == | Eşit mi? | 5 == 5 | true |
| != | Eşit değil mi? | 5 != 3 | true |
| > | Büyük mü? | 5 > 3 | true |
| < | Küçük mü? | 5 < 3 | false |
| >= | Büyük veya eşit mi? | 5 >= 5 | true |
| <= | Küçük veya eşit mi? | 3 <= 5 | true |

4. Mantıksal Operatörler

Mantıksal operatörler, boolean ifadeler üzerinde işlemler yapar.

a = 1 ve b = 2 olmak üzere

| Operatör | Açıklama | Örnek | Sonuç |
|----------|-----------------------|--------------------|-------|
| && | Mantıksal VE (AND) | (a > 0) && (b > 0) | true |
| | Mantıksal VEYA (OR) | (a > 0) (b > 0) | true |
| ! | Mantıksal DEĞİL (NOT) | !(a > 0) | false |

Not:

- && ve || operatörleri **kısa devre (short-circuit)** mantığıyla çalışır.
 - **&& Operatörü:**
 - İlk ifade false ise, ikinci ifade değerlendirilmez.
 - **|| Operatörü:**
 - İlk ifade true ise, ikinci ifade değerlendirilmez.

5. Bit Düzeyinde Operatörler

Bit düzeyinde operatörler, sayıları bit düzeyinde işler.

| Operatör | Açıklama | Örnek | Sonuç |
|----------|-------------------------|-------|-------|
| & | Bit düzeyinde VE (AND) | 5 & 3 | 1 |
| | Bit düzeyinde VEYA (OR) | 5 3 | 7 |

| Operatör | Açıklama | Örnek | Sonuç |
|----------|-------------------------------|--------|-------|
| ^ | Bit düzeyinde Özel VEYA (XOR) | 5 ^ 3 | 6 |
| ~ | Bit düzeyinde DEĞİL (NOT) | ~5 | -6 |
| << | Bitleri sola kaydırma | 5 << 1 | 10 |
| >> | Bitleri sağa kaydırma | 5 >> 1 | 2 |

Örnek Açıklama:

- 5 ve 3'ün ikilik karşılıkları:
 - 5 → 0101
 - 3 → 0011
- 5 | 3 işlemi:
 - 0101 OR 0011 → 0111 → 7

Not:

- Bit düzeyinde operatörler, sayısal değerlerin **bitlerine doğrudan** etki eder.
- Mantıksal operatörler ile karıştırılmamalıdır.

Burada bilmeniz gereken şu dur. 5 ve 3 değeri 32 bitten oluşur. Her bir bit birbiri ile doğrudan işleme sokulur. Sayı tabanları ve mantık konularını tamamladıysanız bu işlemleri rahatlıkla anlayabilirsiniz. Burada bit'lere sayı olarak değil mantıksa ifade olarak davranılır. Mantık dersinden hatırlayacağınız üzere

1 veya 0 = 1 – 1 ve 0 = 0 'dır. Bütün 32 bit sıra ile bu işleme tabi tutulur. İşlem sonucunda elde edilen bilgi yine 32 bit'lik bir sayı olarak elde edilir.

Sola ve sağa kaydırma işlemlerine şimdi bakmanıza ya da anlamamanıza gerek yoktur. Ancak oldukça basittir. 32 bit'in her biri sağa kaydırmada bir sağ tarafa kaydırılır. Sola kaydırmada ise bir soldakine kaydırılır. Bunu basit programlar yazarak görebilirsiniz.

Değil operatöründe 32 bit'in her birine mantıksa bir ifade gibi davranılır. Eğer bit değeri 1 ise 0, 0 ise 1 yapılır.

6. Diğer Operatörler

6.1. Tip Dönüşüm Operatörü

- **Tip Dönüştürme (())**
 - **Açıklama:** Bir değeri belirli bir türe dönüştürmek için kullanılır.
 - **Örnek:**

```
double d = 9.5;
```

```
int i = (int)d;
```

// i: 9

6.2. Üyelik ve Erişim Operatörleri

| Operatör | Açıklama | Örnek |
|----------|------------------------------|---------------|
| . | Üye erişimi | nesne.Metod() |
| [] | Dizi veya koleksiyon erişimi | dizi[0] |

Burada ki nesne.Metod() ifadesine daha sonra değinilecektir. Koleksiyon [] ifadesi ise bir sonraki konuda işlenecektir. Şimdilik bunları bilmenize gerek yoktur. Ancak nesne.Metod() ifadesini anlamış olmalısınız. Bunu şimdiye kadar kullandınız. Console.WriteLine() gibi.

7. Öncelik ve Birleştirme (Associativity)

Operatörlerin öncelik sırası, ifadelerin nasıl değerlendirileceğini belirler. Yüksek öncelikli operatörler önce değerlendirilir.

| Öncelik | Operatörler |
|---------|------------------|
| 1 | () [] . |
| 2 | ! (type) ++ -- |
| 3 | * / % |
| 4 | + - |
| 5 | << >> |
| 6 | < <= > >= |
| 7 | == != |
| 8 | & |
| 9 | ^ |
| 10 | |
| 11 | && |
| 12 | |
| 13 | = += -= *= /= %= |

Not:

- Parantezler () operatör önceliğini değiştirmek için kullanılır.
- Operatörlerin **birleştirme yönü** (soldan sağa veya sağdan sola) da önemlidir, ancak temel düzeyde bu ayrıntıya girmeye gerek yoktur.

8. Örnek Kullanım

```
int a = 10;
```

```
int b = 5;
```

```
int c = 2;
```

```
int sonuc = a + b * c;
```

```
// sonuc: 20 (Önce çarpma yapılır) int sonuc2 = (a + b) * c;
```

```
// sonuc2: 30 (Parantez önceliği değiştirir) bool karsilastirma = (a > b) && (b > c);
```

```
// karsilastirma: true
```

9. Mantıksal ve Bit Düzeyinde Operatörlerin Farkı

- **Mantıksal Operatörler (&&, ||, !):**

- Boolean (true veya false) ifadeler üzerinde çalışır.
- Kısa devre (short-circuit) mantığıyla çalışır.
- Örnek:

```
bool sonuc = (a > 0) && (b > 0);
```

- **Bit Düzeyinde Operatörler (&, |, ^, ~):**

- Sayısal değerlerin bitleri üzerinde çalışır.
- Kısa devre yapmaz, her iki tarafı da değerlendirir.
- Örnek:

```
int sonuc = a & b;
```

Önemli Not:

- & ve | operatörleri, sayısal değerlerle kullanıldığında bit düzeyinde işlem yapar.
- Eğer & ve | operatörlerini boolean ifadelerle kullanırsanız, mantıksal VE ve VEYA işlemleri yaparlar, ancak **kısa devre yapmazlar**.
- Mantıksal işlemler için genellikle && ve || operatörleri tercih edilir.

10. Özet

Operatörler, C# dilinde ifadelerin değerlendirilmesinde temel yapı taşlarıdır. Doğru ve etkin bir şekilde kullanıldığında, kodunuzun okunabilirliğini ve işlevselliğini artırır.

OPERATÖRLER İLE İLGİLİ BİLİNMESİ GEREKENLER

Başlarda en fazla atama , aritmetik ve karşılaştırma operatörlerini kullanacağız. Bunlar oldukça sabittir. Ancak basit kullanımı ve doğal anlayışa yakın davranışlarından dolayı gözden kaçırabileceğiniz bazı durumlar olabilir. Bundan dolayı bilmeniz gereken basit birkaç kuralı anlatalım

1. Kural : Örnek olarak atama operatörünü ele alalım. “=” operatörü her zaman sağ taraftakini sola taraftakine atar. Eğer sağ tarafta bir işlem var ise işlemin sonucunda elde edilen değer atanır.

HATIRLATMA: Programın herhangi bir yerinde değişken tanımladıktan sonra değişkenin adını tekrar yazarsak değişkenin değerini okumuş ve oraya çağırmış oluruz. Örnek olarak “int sayi” değişkenini burada tanımlamış olduk. İlerleyen satırlarda “sayi” yazarsak bu değişkeni çağırmış ve değerini okumaya çalışmış oluruz.

Örnek bir program yazalım ve üzerinden gidelim.

```
unsafe static void Main(string[] args)
{
    int sayi = 10;
    Console.WriteLine(sayi);
}
```

Şekil 8 Değişkene değer atama operatörü ve değişkeni çağırma işlemi

İnt sayi ile değişkeni tanımladık. Ram’de 4 byte’lık bir alan açıldı ve ilk hücrenin adresi “sayi” olarak temsil ediliyor. “=” ile bu değişkene değer atadık. Değerimizin 10 olacağını söyledik. Ancak ram’de 10 diye bir bilgi tutulamaz. İlk olarak bu sayının ikilik tabana çevrilmesi gerekecektir. 10 sayısı ikilik tabanda 1010’dır. Bunu sayı tabanları dersinden biliyor olmalısınızdır. Bu değer dönüşümü yapıldıktan sonra “sayi” değişkeninin temsil ettiği adrese gidilir. 32 bit’e şu değer yazılır. 00000.....1010. Ardından atama işlemimiz tamamlanmış olur.

Console.WriteLine(sayi);

İşleminde ise sayi adlı değişkenin temsil ettiği adrese gidilir. Değişkenin tipi kontrol edilir ve 1 byte’tan daha fazla değere sahip olup olmadığı kontrol edilir. Eğer 1 byte’ta daha fazla değer sahipse diğer byte’lara da sıralı olarak (ya da duruma göre aynı anda gidilir! Buna bilgisayarda hafıza dersinde kısaca değindim. Data bus 64 bit ise tek seferde 8 byte’a kadar işlemci bilgi okuyabilir.) gidilir ve tüm değer ikilik tabanda okunur. Okunan 0000.....1010 değerinin tipine bakılır. “sayi” değişkeninin tipi tam sayı yani “int” olduğu için bu değer onluk tabana çevrilir ve ekranda “10” olarak yazdırılır.

Bu işlemlerin tamamı .Net framework ve işletim sistemi sayesinde otomatik olarak yapılır.

Size sadece “int sayi = 10” yazmak yeterli olacaktır.

Bunu daha detaylı görmeye çalışalım.

```
unsafe static void Main(string[] args)
{
    int sayi = 5+3;
    Console.WriteLine(sayi);
}
```

Şekil 9 İşlem sırası örneği.

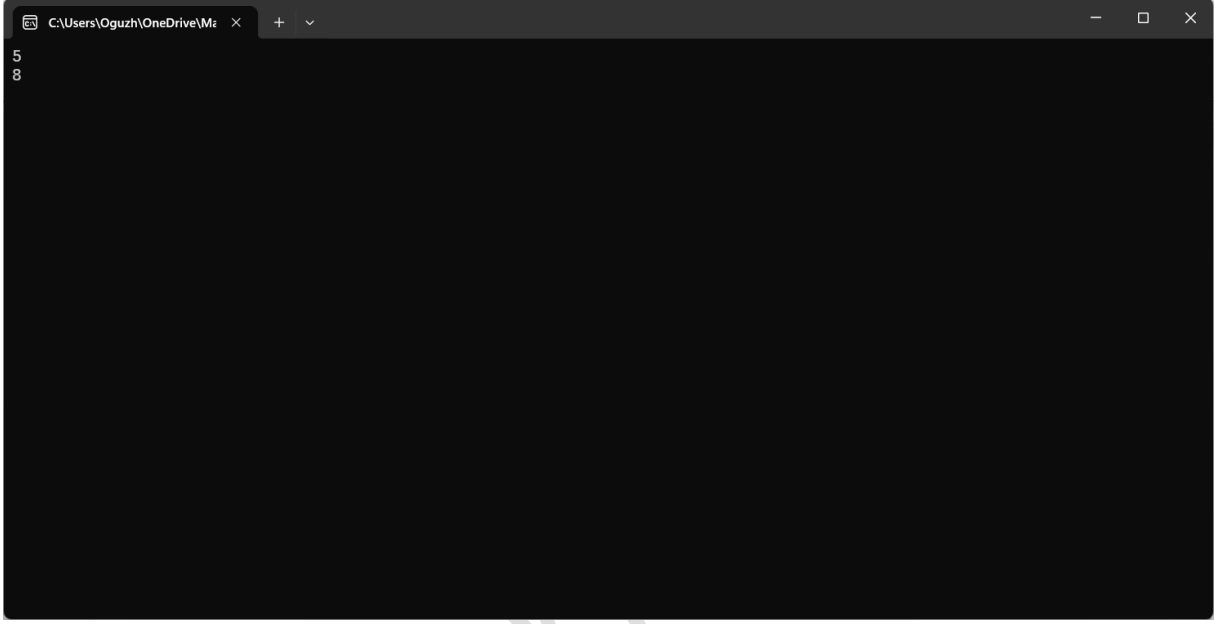
Her zaman sağ taraftaki değerın sola atandığını söylemiştik. Ancak şekil 4 ‘teki programı yazıp çalıştırdığınızda çıktının 8 olduğunu göreceksiniz. Atama yapılabilmesi için sağ tarafta bir değer olması gerekmektedir. Bundan dolayı ilk olarak “+” operatörü çalışmaktadır. “+” operatörünün elde ettiği sonuç değer olarak geri döner. Bu geri dönen değer “sayi” değişkeninin içine yazılır.

Birde kafa karıştırıcı konuya bakalım.

```
unsafe static void Main(string[] args)
{
    int sayi = 5;
    Console.WriteLine(sayi);
    sayi = sayi + 3; // burada sayi+=3 şeklinde de yazılabilir. ikiside aynı şeyi yapar.
    Console.WriteLine(sayi);
}
```

Şekil 10 İşlem Sırası Örneği 2

Şekil 10'daki işlemin çıktısı şu şekildedir;



Şekil 11 İşlem sırası örneği 2 programın çıktısı

Bu programda 3. Satır kafanızı karıştıran nokta. Şimdi adım adım inceleyelim.

İlk satırda “sayi” adından tam sayı olan bir değişken tanımladık ve tanımlama ile aynı anda 5 değerini içine atadık.

“sayi” değişkenini tekrardan yazarak bu değeri okumaya çalıştık. Console.WriteLine(sayi); metodunun içerisinde “sayi” yazarak bu değişkeni çağırdık ve 5 sayısını buraya çağırmış olduk. Console.WriteLine(5); işlemini elde ettik. Bu sayede konsol ekranında 5 sayısını görebildik. Son satırda da aynısını yaptık. Peki nasıl 8 sonucunu elde ettik.

“sayi = sayi + 3” burada “=” operatörünün atama yapabilmesi için sağ tarafta bir değer olması gerekiyor. Ancak sağ tarafta “sayi + 3” bulunmakta. Burada “+” operatörü bir değer döndürecek. Bu döndürdüğü değer ise “sayi” değişkenine atanacak. Bundan dolayı ilk olarak “+” operatörü çalışacaktır. Sağ tarafta ise sayi değişkenini çağırdık. Bu değişkenin değeri oraya getirilecek. Yani “sayi + 3” = “5 + 3” olacaktır. Toplama işleminin sonucunda 8 elde edilecektir. Bu 8 değeri ise tekrar “sayi” değişkenine atanacaktır.

DEĞİKEN ADLANDIRMA, ATAMA VE FORMATLAMA

Değişkenler ile çalışırken dikkat etmeniz ve bilmeniz gereken birkaç kural bulunmaktadır. Buradan adlandırma ve veriyi formatı en önemli konulardandır. İlk olarak adlandırmaya kısa bir bakış atalım.

Adlandırma

Çoğu programlama dilinde değişkenleri adlandırırken bazı kurallar uymak önemlidir. C# dilinde program geliştiren yazılımcılarda bu kurallar genel olarak uymaktadır. Sizlerden de bu kurallara uymanız beklenmektedir.

Programlama dillerinde adlandırma kuralları değişkenler dışında başka yapılarda da bulunmaktadır. Metodlar, alanlar vs.... Onlara zamanı gelince ayrıca değineceğiz. Şu anlık sadece değişkenlere bakacağız.

Değişkenler adlandırılırken genellikle sayı kullanılmamaya çalışılır. Kullanmakta bir sıkıntı yoktur. Ancak değişken sayı ile başlamamalıdır. Örnek olarak;

- `İnt sayi1;` doğru
- `İnt 1sayi;` yanlış

Değişkenler birkaç kelimeden oluşabilir. Bu kelimeler birbirinden “-” ve ya “_” ile ayrılabilir. Ancak genel olarak uyguladığımız kural ise şu şekildedir. İlk kelime tamamen küçük harflerden oluşur. Ardından gelen kelimelerin sadece baş harfleri büyük olur.

- `İnt sayiBir;` doğru
- `İnt sayibir;` yanlış

Olabildiğince kısa ancak anlamlı adlar verilmelidir.

- `İnt a;` yanlış
- `İnt yas;` doğru.

Bunun dışında uymanız gereken birkaç basit bir kural vardır. İlk olarak (“!”-“;”) gibi özel karakterler içeremez. Türkçe karakterler içeremez. Anahtar kelimeler içeremez. Anahtar kelimeleri kullandıkça daha çok fark edeceksiniz. Ayrıca anahtar kelimeler ileride işleyeceğimiz bir konu olduğundan şimdi burada girmeyeceğim. Ancak bir kaçını şimdiden öğrendiniz. “true, false, int, byte vs...” gibi.

Veriyi Formatlama

Kullandığımız değişken tiplerine göre değişkenlere atama yapmak zorundayız. Örnek olarak int değişken tipinde bir değişkene karakter ataması yapamayız. Karakterlere de sayıları atayamayız. Ancak rakamları atayabiliriz.

Ayrıca “double,float ve decimal” gibi kayan noktalı tiplerde dikkat edilmesi gereken bir nokta daha vardır. Buraya yaptığımız atamada hassasiyeti de belirtmemiz gerekir. Yani bir sayının yüksek hassasiyet ya da düşük hassasiyet belirttiğini “f,d,m” gibi harflerle belirtmemiz gerekmektedir. Çoğu zaman problem olmasa da burada verinin formatını belirtmekte fayda vardır. Bunu size ödev olarak vereceğim. “Epsilon” değerini araştırdığınızda bu hassasiyetin önemini daha çok anlayacaksınız.

Bir diğer konuda sayı tabanları ile alakalı. Farkındaysanız int değişken tipine sadece onluk tabanda sayı yazarak atama yaptık. Ancak istersek ikilik ve onaltılık tabanda atama da yapabiliriz. Sadece düzgün formatta yazmamız gerekmektedir.

Son konu ise char yani karakter değişken tipi ile alakalı. Bu değişkene atama yaparken sadece bir karakter atayabiliriz. Bu karakteride " arasında yazmamız gerekmektedir. Örnek;

'A' – 'b' – '3' vs... gibi.

Peki uzun metinleri nasıl yazacağız ve string değişken tipi nedir diye soruyor olabilirsiniz. Bunu bir sonraki konuda, array'ler yani sıralı değişkenlerde işleyeceğiz. Bunda dolayı bu derste string değişken tipinin üzerinde durmayacağız.

Hadi bunları kod üzerinde gösterelim;

```
static void Main(string[] args)
{
    // bool veri tipi (true/false)
    bool isAvailable = true;
    // byte (0-255)
    byte age = 25;
    // sbyte (-128 to 127)
    sbyte temperature = -15;
    // char ('c' formatında)
    char grade = 'A';
    // short (-32,768 to 32,767)
    short shortNumber = -12345;
    // ushort (0 to 65,535)
    ushort ushortNumber = 54321;
    // int (-2,147,483,648 to 2,147,483,647)
    int decimalNumber = 123456; // Ondalık tabanda
    int hexNumber = 0x1A3F; // 16'lık tabanda
    int binaryNumber = 0b101011; // 2'lik tabanda
    // uint (0 to 4,294,967,295)
    uint unsignedInt = 1234567890;
    // long (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
    long longNumber = 9876543210;
    // ulong (0 to 18,446,744,073,709,551,615)
    ulong unsignedLong = 12345678901234567890;
    // float (kayan noktalı sayılar, f ile)
    float pi = 3.14f;
    // double (kayan noktalı sayılar, d opsiyonel)
    double largePi = 3.141592653589793;
    // decimal (hassas kayan noktalı, m ile)
    decimal price = 99.99m;
    // string (karakter dizisi)
    string greeting = "Hello, World!";
}
```

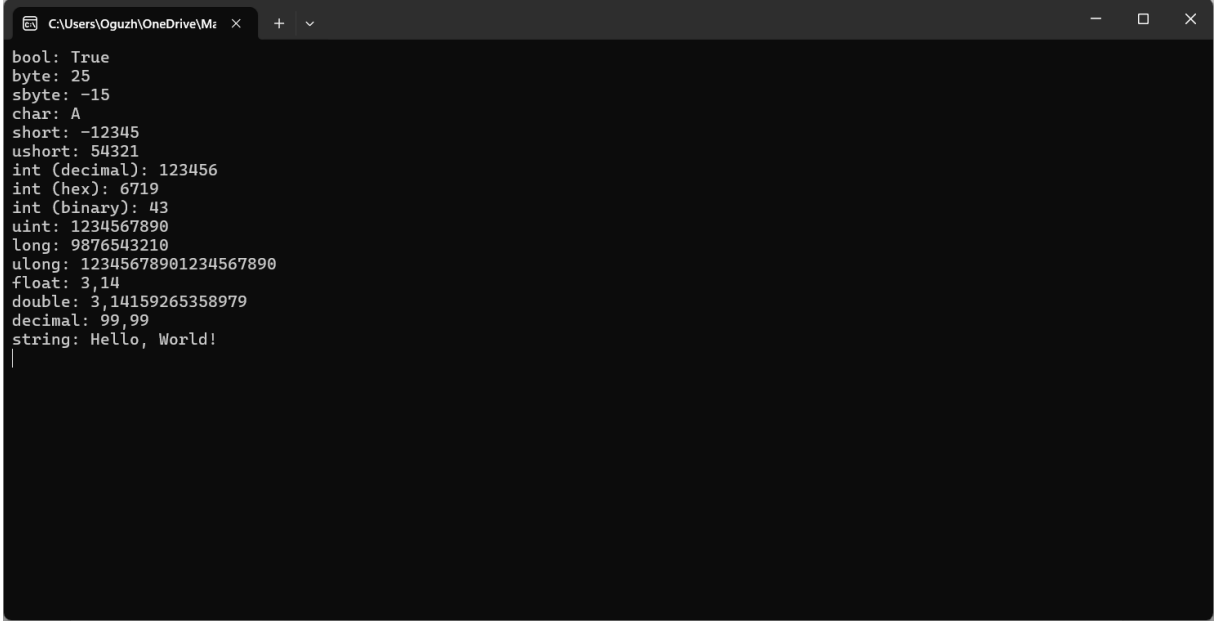
Şekil 12 Verinin çeşitli formatları

Bu programı lütfen el ile teker teker yazın. Aklınızda kalması için en iyi yöntem budur. Ancak çıktıyı görebilmek için kısaca kodu kopyala yapıştır yapmanıza izin vereceğim. Aşağıdaki kodu kopyalayarak en azından çıktıyı amanıza gerek yoktur.

```
Console.WriteLine($"bool: {isAvailable}");
Console.WriteLine($"byte: {age}");
Console.WriteLine($"sbyte: {temperature}");
Console.WriteLine($"char: {grade}");
Console.WriteLine($"short: {shortNumber}");
Console.WriteLine($"ushort: {ushortNumber}");
Console.WriteLine($"int (decimal): {decimalNumber}");
Console.WriteLine($"int (hex): {hexNumber}");
Console.WriteLine($"int (binary): {binaryNumber}");
Console.WriteLine($"uint: {unsignedInt}");
Console.WriteLine($"long: {longNumber}");
Console.WriteLine($"ulong: {unsignedLong}");
Console.WriteLine($"float: {pi}");
Console.WriteLine($"double: {largePi}");
Console.WriteLine($"decimal: {price}");
Console.WriteLine($"string: {greeting}");
```

`Console.Read();`

Programın çıktısı şu şekildedir;



```
bool: True
byte: 25
sbyte: -15
char: A
short: -12345
ushort: 54321
int (decimal): 123456
int (hex): 6719
int (binary): 43
uint: 1234567890
long: 9876543210
ulong: 12345678901234567890
float: 3,14
double: 3,14159265358979
decimal: 99,99
string: Hello, World!
```

Şekil 13 Veri formatla programı çıktısı

Gördüğünüz gibi. Tüm veriler formatlı bir biçimde atama yapıldı. Okuma yaparken ise beklediğimiz tipte okuma yaptık. Beklediğimiz tip ise şudur. Onluk tabanda sayılar. Onluk tabanda ondalıklı sayılar ve karakterler. İleride bu çıktıları istediğimiz şekilde manipüle etmeyi öğreneceğiz. Ancak şimdilik bu konuyu burada bitirmek iyi olacaktır.

Not: Ondalıkli sayılar bizde “,” ile ayrılmaktadır ve biz bunlara “virgüllü sayılar” demekteyiz. Ancak C# programlama dilinde bu sayılar “.” ile gösterilmektedir ve bunlara kayar noktalı sayılar denmektedir. İleride geliştireceğimiz uygulamalarda buna dikkat ederek geliştirme yapmalıyız.

Not: Bu bahse konu olan değişkenlerin içerisinde Sayı ya da Karakter gibi değerler sakladığımızdan bunlara “Değer Tipli Değişkenler” (value type) denmektedir.

ADRESLERİN SIRALI OLMASI ÜZERİNE AÇIKLAMA

İnt değişken tipini konuşurken bunun 4 byte alan kapladığını ve ilk byte’ın adresinin “sayı” ile temsil edildiğini söylemiştik. Peki 4 hücreye de nasıl ulaşacağız? Bunu .Net Framework ve işletim sistemi otomatik olarak yapar. “sayı” nın temsil ettiği adrese bakar. Sonra “sayı” nın tipine bakar ve kaç byte olduğunu görür. “int” için 4 byte olduğundan bakması gereken hücrelerin “sayı – sayı + 1 – sayı + 2 – sayı + 3” olduğunu hesaplar. Bu verileri okuyarak konuma verileri getirir. Hadi bunu kodlayarak gösterelim.

Bu kod bloğu resme sığmayacak. Bundan dolayı buraya kodları kopyalayacağım. Ancak Bu kodları okuyun. Çalıştırmayın.


```

class Program
{
    unsafe static void Main(string[] args)
    {
        int sayi = 1273; // Örnek sayı
        string binarySayi = Convert.ToString(sayi, 2).PadLeft(32, '0');
        string formattedBinarySayi =
UnsafeCode.InsertSpaceEvery8Bits(binarySayi);

        Console.WriteLine("Sayı: " + sayi);
        Console.WriteLine("Sayı (Binary): " + binarySayi);
        Console.WriteLine("Formatlanmış Sayı (Binary): " + formattedBinarySayi);
        // 'sayi' değişkeninin adresini alıyoruz
        int* ptr = &sayi;

        // 'bytePtr' adında bir byte işaretçisi oluşturuyoruz
        byte* bytePtr = (byte*)ptr;

        // Bellek düzenini belirlemek için bir değişken
        bool isLittleEndian = BitConverter.IsLittleEndian;
        Console.WriteLine("Bellek Düzeni: " + (isLittleEndian ? "Little-Endian" :
"Big-Endian"));
        Console.WriteLine();

        // Her bir byte'ı okuyup yazdırıyoruz
        for (int i = 0; i < sizeof(int); i++)
        {
            // Byte'ın bellek adresi
            byte* currentBytePtr = bytePtr + i;

            // Byte'ın değeri
            byte currentByte = *currentBytePtr;

            // Byte'ın onluk ve onaltılık değeri
            Console.WriteLine($"Byte {i + 1}: {currentByte} (Decimal),
0x{currentByte:X2} (Hex), Adres: 0x{(ulong)currentBytePtr:X}");

            // Byte'ın binary gösterimi
            string byteBinary = Convert.ToString(currentByte, 2).PadLeft(8, '0');
            Console.WriteLine($"Byte {i + 1} (Binary): {byteBinary}");

            // Orijinal sayının ilgili bitlerini alıyoruz
            int bitStartIndex = isLittleEndian ? (i * 8) : ((sizeof(int) - 1 - i)
* 8);
            string originalBits = binarySayi.Substring(32 - bitStartIndex - 8,
8);
            Console.WriteLine($"Bitler {bitStartIndex}-{bitStartIndex + 7}:
{originalBits}");

            // Karşılaştırma
            bool bitsEqual = originalBits == byteBinary;
            Console.WriteLine("Eşleşme: " + (bitsEqual ? "Evet" : "Hayır"));
            Console.WriteLine();
        }
    }
}

```


Bunların hepsinin aslında genel kültür olduğunu unutmayınız. Sonuç olarak sizler sadece “int sayı” ile değer atayarak bu değerleri kullanacaksınız. Ancak ne yaptığınızı bilmeni oldukça önemlidir. Bu sizi iyi bir yazılımcı yapacaktır.

ÖDEV

- Her tipte değişkenlerden birer tane oluşturup içlerine atama yapınız. (string haricinde! Bir sonraki konu) Ardından bu değişkenleri “Console.WriteLine()” kullanarak yazdırınız ve çıktıları görünüz.
- Float – double – decimal veri tipleri için epsilon değerinin ne olduğunu araştırınız. Bunun hakkında kısa bir rapor yazınız.
- Araştırma Konusu: ASCII (Bir sonraki derste öğreneceğiz. Ön hazırlık olması için şimdiden hazırlık yapınız)

Kaynakça

KARAGÜZEL, O. (2024). *Bilgisayar ve Hafıza - Hafıza Türleri ve Ram'in iç yapısı*. (O. KARAGÜZEL, Dü.) 2024 tarihinde github: <https://github.com/Oguzhankaraguzel/online-egitim-serisi/blob/main/3.Ders%20-%20Bilgisayar%20ve%20Haf%C4%B1za/3.%20Ders%20-%20Bilgisayarda%20Haf%C4%B1za.pdf> adresinden alındı

KARAGÜZEL, O. (2024, 11 20). *Sayı Tabanları - Kısa Bir Hatırlatma*. (O. KARAGÜZEL, Dü.) 2024 tarihinde github: <https://github.com/Oguzhankaraguzel/online-egitim-serisi/blob/main/Ek%20%C4%B0%C3%A7erik/Say%C4%B1%20Tabanlar%C4%B1/Say%C4%B1%20Tabanlar%C4%B1.pdf> adresinden alındı

Öğuzhan KARAGÜZEL

TEST

Her türlü yöntem ile kopya çekmek serbest. Sadece soruyu kopyalayıp yapıştırmak yasaktır. Her bir soru 5 puandır. Eğer 60'ın altında bir puan alırsanız lütfen dersi tekrar ediniz.

1. Değişken nedir?
A) Bellekte veri depolayan isimlendirilmiş bir konum
B) İşlemci içindeki bir hesaplama devresi
C) İşlemleri gerçekleştiren semboller
D) Bellekte sabit bir veri değeri
2. Bir değişkenin bellekte kapladığı alan neye bağlıdır?
A) İşletim sistemine
B) Değişkenin türüne
C) Programlama diline
D) İşlemci hızına
3. "int" veri tipi bellekte kaç byte yer kaplar?
A) 1
B) 2
C) 4
D) 8
4. C# dilinde "decimal" veri tipi nasıl tanımlanır?
A) Ondalık sayılar için kullanılan kayan noktalı bir veri tipi
B) İşaretsiz bir tam sayı veri tipi
C) Sadece pozitif sayılar için kullanılan bir veri tipi
D) Unicode karakterleri saklayan bir veri tipi
5. Aşağıdaki operatörlerden hangisi "arttırma" operatörüdür?
A) --
B) ++
C) +=
D) =
6. "Little-Endian" bellek düzeninde hangi byte en düşük bellek adresindedir?
A) En anlamsız byte (Least Significant Byte)
B) En anlamlı byte (Most Significant Byte)
C) Rastgele bir byte
D) En büyük byte
7. Aşağıdaki ifadelerden hangisi bir atama operatörüdür?
A) ==
B) +=
C) &&
D) |
8. Aşağıdaki değişken adlandırmalarından hangisi doğrudur?
A) int 1sayi;

- B) int sayiBir;
C) int sayı;
D) int true;
9. "bool" veri tipi bellekte kaç bit etkin olarak kullanılır?
A) 1
B) 8
C) 16
D) 32
10. Aşağıdaki veri tiplerinden hangisi en geniş aralığa sahip ondalıklı sayıları saklayabilir? (Dikkat hassasiyet değil! Aralık)
A) float
B) double
C) decimal
D) char
11. Bit düzeyinde sola kaydırma işlemi hangi operatörle yapılır?
A) <<
B) >>
C) &
D) |
12. "5 % 2" işleminin sonucu nedir?
A) 1
B) 2
C) 3
D) 0
13. "uint" veri tipi hangi aralıkta değer alır?
A) -128 ile 127 arasında
B) -2,147,483,648 ile 2,147,483,647 arasında
C) 0 ile 65,535 arasında
D) 0 ile 4,294,967,295 arasında
14. C# dilinde "==", "!=" gibi operatörler hangi kategoriye girer?
A) Aritmetik Operatörler
B) Atama Operatörleri
C) Karşılaştırma Operatörleri
D) Mantıksal Operatörler
15. Aşağıdaki ifadelerden hangisi bir mantıksal operatördür?
A) &
B) &&
C) ^
D) +=
16. C# dilinde bir karakteri göstermek için hangi format kullanılır?
A) "c"
B) 'c'

- C) c
- D) c'''

17. Aşağıdaki veri tiplerinden hangisi yalnızca doğru veya yanlış değerlerini saklar?

- A) float
- B) int
- C) char
- D) bool

18. Aritmetik operatörlerden hangisi kalan bulmak için kullanılır?

- A) *
- B) /
- C) %
- D) -

19. Değişkenlere onaltılık tabanda değer atanırken hangi önek kullanılır?

- A) 0b
- B) 0x
- C) 0o
- D) 0d

20. Aşağıdakilerden hangisi statik tür tanımlı bir dildir?

- A) Python
- B) JavaScript
- C) C#
- D) PHP

Cevap Anahtarı

1. A
2. B
3. C
4. A
5. B
6. A
7. B
8. B
9. A
10. B
11. A
12. A
13. D
14. C
15. B
16. B
17. D
18. C
19. B
20. C