Parse metodu sadece string dataları hedef türe dönüştürürken kullanılır.

Chechked bilinçli bir tür dönüşümü esnasında bir veri kaybı olacaksa bizi uyarır.

Unchecked Normal bir kod bloğu normal olarak uncheckdır.

Tür dönüşüşmlerinde dönüştürülecek türün hedef türe uygun olması gerekdi.

Bir sitisna olarak 0'ın dışındaki türlerin boolean bir ifadeye çevirirken true; 0'a dönüştürürken false olduğunu görürüz.

Cast ör: int \_a = (int)a;

Parse ör: int number = Int.Parse("1235");

İki farklı türde sayısal değer üzerinde yapılan aritmeik işlem sonucunda geriye buyuk olan turde sonuc donecektir.

Kucuk olan tur bilinçsiz tur dönüşümü yaparak gerçekleşir.

İki byte üstünde yapılan aritmetik işlem sonucunda geriye INT deger doner.

ya da(^) operatörü, şartlardan kesinlikle birtanesinin yerine getirmesini ister...

Ternary ör:

int sonuc = sayi < 3 ? sayı \*5 :

(sayı > 3 && sayı sayı < 9) ? sayı \* 3 :

(sayı >= 9 && sayı % 2) == 0 ? sayı \* 10 :

sayı % 2 == 1 ? sayı : -;

Boxinge tabni tutulmuş değerin türünü ogrenmek için kullanırız. Geriye bool deger doner.

C# proglamlama dilinde deger tipler null deger alamaz.

As operatoru: Cast operatorune alternatif olarak çıkmıştır. Unboxıng işlemlerinde as operatoru degeri dışarı çıkartır ve ataycagı degere uygun ise atar;

ama değilse null deger doner. Bu yuzden deger tipler ile çalışamaz, null alabilecek degerlerle çalışır.

Bir değer turlu degiskenin null deger alabilmesi için ? operatorunu kullanması lazım. int? a = null;

ardından butun null ile çalışan operatorler uzerinde çalışabiliriz.

Null Coalescing: ??= gönderilen değer null ise verilen değeri ata null değilse değerini koru ör: int? a = null; a ??= 1; a'ya 1 değerini atayacak.

Switch case de case degerleri bir değişkenden alınamaz. Sabit olmalı.

Type Pattern: object içerisindeki değerleri kontol etme aşamasında bize çok kolaylık sağlıyor.

ör: if(x is string a) a da x string ise atama yaptık. if içerisinden kullandığımız değişken ismini başka if'de

kullanamayız.

Constant Pattern: bizim == ile kontrol sağladığımız olayları is operatoru ile sağlayabiliyoruz.

int a=0;

if(a is 5), burada is opertorunun diğer olayı if(a is int) a int bir değer mi bunu da unutmayalım.

ister değer türlü istersen de referans türlü kontrolleri sağlayabilirsin.

bir değer türü kontrolü yapılıyorsa bu Constatnt Operatorudur.

Var Pattern: değer atama yaparken Type pattern da type yerine var yazıyoruz.

ör: if(x is var a) var ın türü runtime da belirlenecek.

var keyword u ile var pattern arasındaki fark keyword derleme esnasında türüm belirlenirken, var pattern runtime da

belirleniyor.

Relational Pattern: Switch ile if arasındaki farkı ortadan kaldırıyor.

Try catch: Maliyetli bir yapıdır. Catch blogunda Exception hataların annesidir. Finally blogu try cath blokları çalışsada çalışmasada

çalışır. When ile bir şart daha koyabiliriz.

For: ilk olarak deger atandıktan sonra şarta bakar doğruysa scopelar arsındaki işlemi yapar ardından aritmetik işlemi yapar sonra

şartı kontrol eder dogruysa devam eder aynı şekilde.

Keyword: Proglamlama dilinin en atomik parçalarıdır.

Derleyici için ön tanımlı olan, nerede hangi amaca hizmet ettiği bellidir sınırlandırılmıştır.

operatorler belirli bir operasyonu eylemi gerçekleşştiren yapılardır.

Keyword operatorleri kaplar. Her operasyon keyworddur.

Manevratik Komutlar: yapamayacağımız şeyleri yapmaya yarayan komutlar değildir. Kodun yönlendirmelerini daha efektif yapmaktadır.

break: Döngü ve switch case de kullanilabilir keyworddur. Kullanıldığı yapıyı sonlandırır. foreach de de kullanılır. foreach bir iterasyondur.

continue: sadece döngülerde çalışır. Döngüde bir sonraki periyoda direk geçiş yaptırır. Döngünün içersinde nerdeyse oradan bırakır ve bir sonraki periyoda geçer(i=0 -> i=1)

Döngülerin içinde herhangi bir keyword içerisinde de kullanılır.

return: metot içerisinde her yerde kullanılabilir erişilebilridir.

iki işlevi görmektedir. 1-) metodtan çıkış yapar. Metodu sonlandırır.

2-) metotlarda geriye değer döndürür.

goto: kodun senkranizasyonunu bozup akisi ters istikamete almaya yarayan kreyworddur. davranıssal olarak dongulere benzer.Metot içerisinde her yerde kullanılır.

Ekstra Bilgi: Bazen (ki ileride asenkron proglamlamda) içi boş sonsuz döngülere ihtiyacımız olabilir. Bu durumlarda normal döngü

gövdelerini kullanabildiğimiz gibi scopesuz da kullanabiliriz. ; opertoru ile temiz kod yazımı.

DİZİLER : Diziler içeriside aynı türden birden fazla veri tutabilen yapılardır.

Prosedurel proglamlamanin temel veri kümeleridir.

Diziler prosedurel prog. Temel yapıları oldukları icin gelisimis yapılar olan koleksşyonlarinda temelini teskil eder.

Diziler referans turlu degerlerdir. Ozlerinde classtirlardır.

Yapisal olarak Ram da heap te tutulutlar

Dizi içlerine koyulan dgerler islevsel olarak aynı maliyette olmalıdırlar.

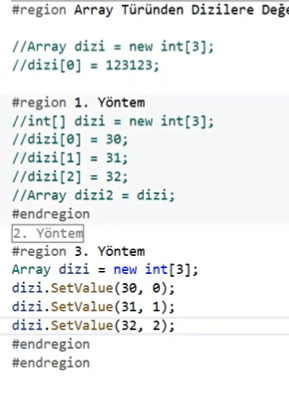
Ornegın yas için kullanılan bir diziye maaş bilgisi girilmemelidir.

Alan tahsisi yapıldıgında alanlara varsayılan deger tahsisi yapılır.

int[ ] yas={1,2,3,4,5}

diziler Array sınıfından türemiştir. Array sınıfından gelen metotları içerir.

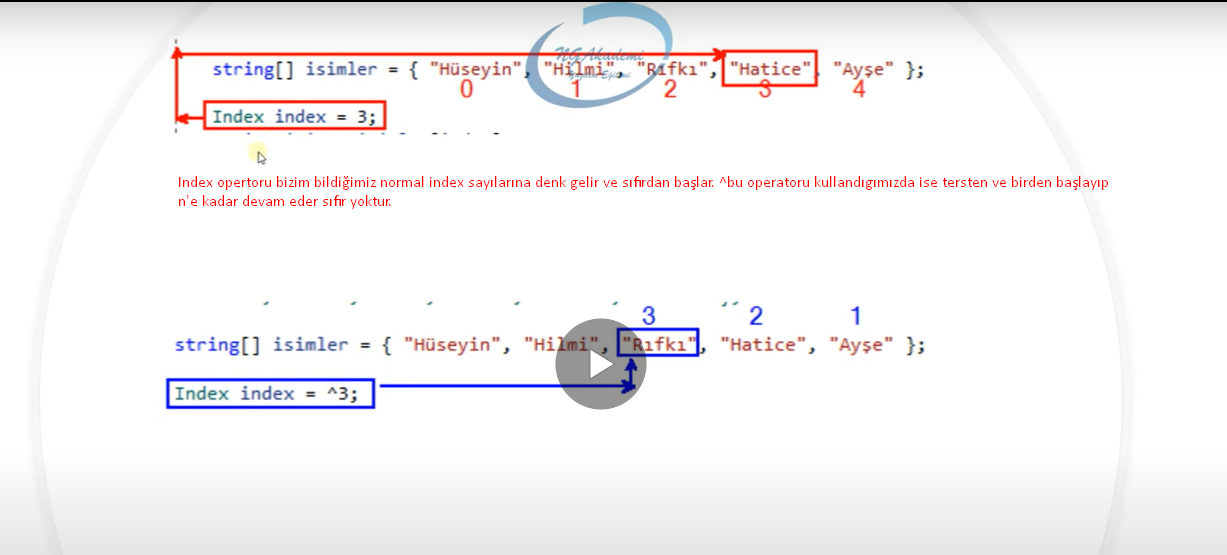
Array sınfından turetilen bir dizide indexer opertoru kullanamayız.



CLEAR: Dizi içerisndeki tum elemanlara türe göre varsayılan degerleri atayan metottur

IsFixedSize dizinin boyutunun sabit olup olmadığını kontrol eder. Tüm dizilerde sabit olacağından true döner List lerde işimize yarayacak.

Dizi yapılanması esasında Array sınıfın arka planında CreateInstance metodu ile oluşturulur. Array.CreateInstance(typeof(type),length)



Range operatoru bize dizilerde belli bir arlıkta çalışmamizi sağlıyor.

Range range = 3..7; mesela burada 3 index numarası iken 7 sıra numarası.

Bize geriye range türünde değer dönecek ve biz ilgili aralığı bir dizi mantığıyla kullancağız Range a sağdan ^3 dediğimizde bize sağdan 4. Yü verecektir.

Dizi içerisinde dizi tanımlama. Visiual studio keyword solution

STRING:

String değişkenler stack de tutulur değerleri heapte tutulur.

String ifadeler esasında bir char dizisidir.

Yazılım açısından string ifade yoktur.

String ifadeler ozunde bir char dizisi/yani dizi olmasından dolayı referans türlü değişkenlerdir. Çünkü diziler referans türlü değişkenlerdir.

Yani nesnedirler ve heapte tutulurlar.

String ifadeler char dizisi olduğunda dolayı her bir karakter baştan sona indexlenir.Dolayısıyla string bir ifade üzerinde bixler indexer opertaorunu kullanabiliriz.

String interpolation string bir veride araya girmemizi sağlıyor.

Bir string değer,nbasşına $ simgesi konulursa 

Bu ifadenin içersinde interpolation opertorunun/opersayonunun kullanilabilmesini sağlar. Yapısal olarak arka planda string format fonksiyonu kullanılır

İhtiyaç halinde süslü parantez kullanmak için tekrardan süslü parantezi kullanarak string interpolationı ezer. {{merhaba süslü parantez ile birlikte ekrana çıktı verecek}}

Kaçış karakteri(\ backward slash) kendinden sonra gelen ifadenen özel bir şey olmadığını ifade eder.

Peki kendisini kullanmak istersekl(\) kendisini tekrar ezerek kullanırız.

Bir değişken ismi kullanmak istiyorsak verbatim(@) operatorunu kullanırız.

İnterpolation operatoru ile birlikte verbatim kullanmak istenirse ilk baş verbatim @ sonra interpolation gelcek $ = @$

Dizilerde Verisel Performans:

Esasında diziler üzerinde işlem yapmak maliyetli bir işlemdir.

Diziler üzerinde yapılan çalışamalarda bu maliyeti ortadan kaldırmabilmek ve daha performanslı çalışabilmek için ArraySegment ve StringSegment türleri geliştirilmiştir.

StringSegmnent strin veriler üzerinde çalışma yapmak için geliştrilmiştir.

Elimizdeki dizide belirli bir alanda o alanı referans etmek için kullanılır.

SPAN

Span bellek üzerinde belirli bir alanı referans etmemizi sağlar.

Belirli bir alandan kasıt tabi ki ardışıl alan kaplayan Array değerleridir.

Noramlde arrayler heap kısımda tutulurken stackalloc keywordu ile belleğin stack ksımında da array tanımlayabiliriz.

Span bize bellekte Stack veya Heap bölümü fark etmeksizin bize bellek üzerinde referans etmeye olanak sağlıyor.

Span, dizi ve string gibi maliyetli veriler üzerinde yapılacak işlemler için bize maliyet ve performans açısından olabildiğince verimli oluyor. Tüm faaliyetleri de gerçekleştirebiliyoruz.

ReadOnlySpan<T> Span niteliklerinin tümünü sağlamakta, orijinal veri üzerinde bir değişiklik olmadan sadece oradaki verileri okuyabilmekte.

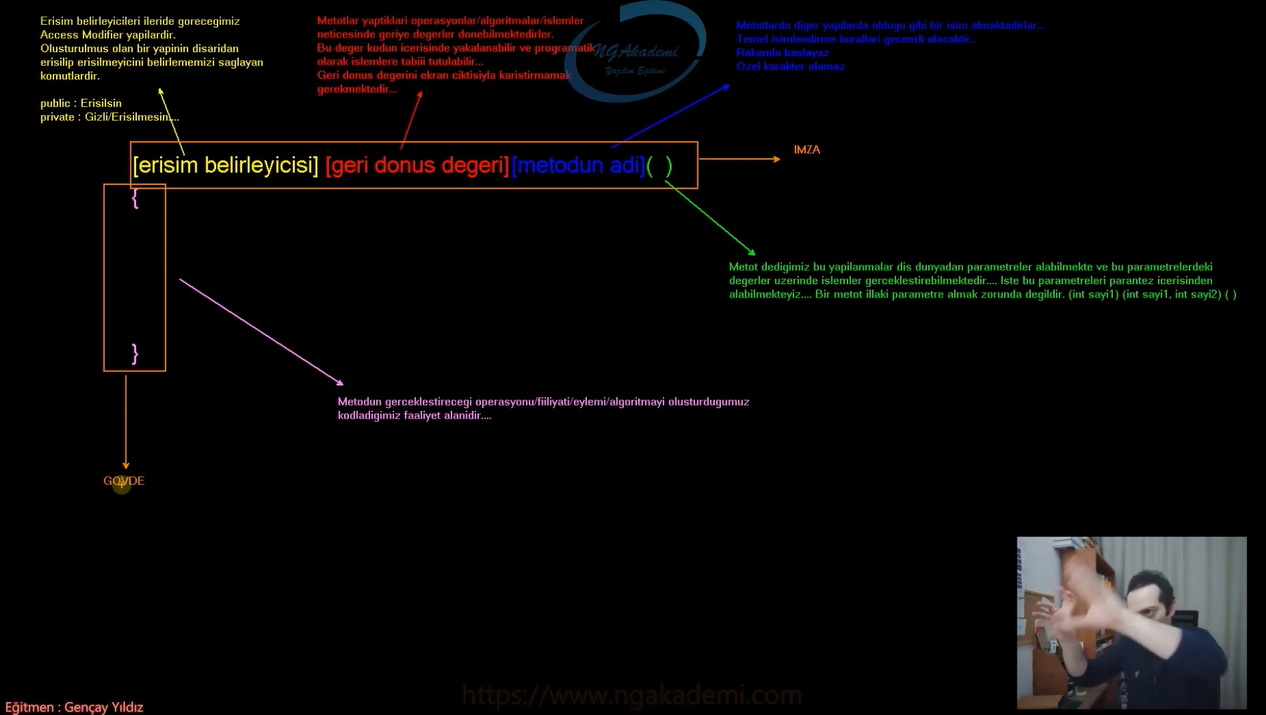
Regular Expression İşlemleri

METOTLAR

Yazılımda gerçekleştirdiğimiz, fiili gerçekleştirdiğimiz tumm operasyonlar metottur.

Metot prosedürel proglamalamanın temel elemanıdır.

Bir iş fiiliyat operasyon yapan en küçük program parçacıklardır.



Yapılacak işleme göre 4 farklı türde/varyasyonda metot oluşturulabilr.

1-> Geriye değer döndürmeyen. Parametre almayan.

2-> Geriye değer döndürmeyen. Parametre alan.

3-> Geriye değer döndüren. Parametre almayan.

4-> Geriye değer döndüren. Parametre alan.

* Bir metot bir işlem yapmalı.
* Parametreli bir metot çağrılırken tetiklenirken alacağı parametleri göndermek zorunlu.
* Bir metodun parametrelerine zorunlu bir şekilde parametre göndermek istemiyorsak, parametreye değeri isteğimize göre /opsiyonel olarak göndermek istiyorsak o parametrelerin bu durumu karşılayabilecek bir ozellikte olması gerekiyor. İste bu ozelliğide opsiyonel parametreler denmektedir
* Opsiyonel parametre demek biz değer göndermesek bile o parametreni bir default değeri vardır.
* Birden fazla opsiyonel parametre durumunda opsiyonel parametreler sağ tarafta tanımlanmalidir

NON-TRAİLİNG NAME ARGUEMENTS

Hangi parametreye hangi değerlerin gönderildiğini direk görebilmek için bu özelliği kulllanırız

Görecele bir şekilde çok parametreli fonksiyonlarda hedef parametrelere değer göndermemizi sağlayan bir yapıdır

In keywordu ilgili parametredeki değerleri readonly yapar

LOCAL FUNCTION

//Bir metot içersinde tanımlanmiş olan metottur

//Metotoların tanımlı olduğu yerler: struct, abstract class, interface, record yapılanmalarında metot tanımlanır.

// metotlar bu saydıklarımız dışında başka bir yerde tanmlanamaz

//Local function özelliği sayesinde metot içerisinde metot tanmımlanabilir.

Tanımlama Kuralları

//1. Erişim belirleyici(Access Modifier) yazılmaz.

//2. Local function olarak tanımlanan fonksiyonun adı tanımlandığı fonksşyonun adından farklı olmalıdır. Aksi taktirde derleyci hata vermez!

Kullanım Kuralları

//Bir local function sade ve sadece tanımlandığı metodun içerisnde kullanılır.

//Local function tanımlandığı metodun içerisinde her yerden erişilebilir.

Recursive Fonk: Kendi içerisinde kendisini çağıran/tetikleyen fonksiyonlardır.

Özyinelemeli/Tekrarlamalı Fonk.

Recursive Fonk. bir yaklaşımdır.

Kullanılması ve anlatılmsaı zordur.

Recursive fonk. ne amaçla kullanılmaktadır?

Öngörülemeyen, derinliği tahmin edilemeyen , sonu bilinmeyen durumlarda tercih edilebilir.

Ref Keywordu

/\*

\* Referansın kısaltmasıdır. Referanstan gelmektedir

\* Referans OOP kavramıdır.

\* OOP'de nesneler RAM'de Heap bölgesinde tutulur. Heap deki verilere direk erişemeyiz.

\* OOP'de referanslar = operatoru ile iletişime geçebilmektedirler. Bir referans işaretlediği herhangi bir nesneyi = operatoru ile farklı bir referans a

\* işaretletebilir.

\* Yani referanslarda = operatoru neticesinde herhangi bir verisel/nesnesel türeme söz konusu olmmamakta. İşaretlenmiş nesne diğer referans tarafından

\* işaretlenmektedir.

\*

\* Değer türlü değişkenlerde referans çalışması yapmak istiyorsak ref keywordu kullanırız.

\* Ref keywordu değer türlü değişkenlerin referanslarını almamızı sağlar. Referans türlü gibi çalışmamızı sağlar.

\* Değer türlü değişkenlerde Shallow Copy(yzeysel kopyalama) yapmamızı sağlar. Normalde Deep Copy oluyordu.