C#. NET

Ders Konusunun Hedefleri

Bu sunumda,

- **C#.Net Programlama Diline Giriş**
- **NameSpace,Class,Metod Kavramları**
- **Console Sınıfı Property ve Metodları**

gibi konular hakkında bilgi verilecektir.

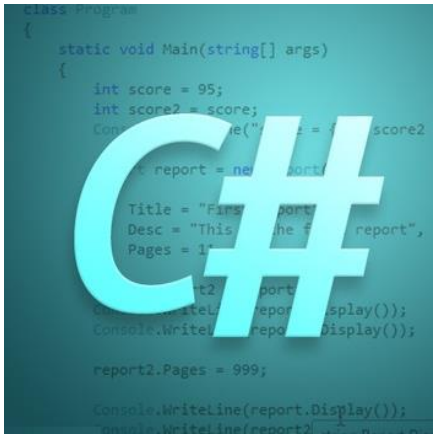
Bu Derste İşlenecek Konular

- C#.Net Genel Bakış
- NameSpace, Class, Metod, Property Kavramları
- Console Sınıfı Özellikleri ve Metodları
- Çeşitli Örnek Çözümleri

C#.NET Genel Bakış

C#.NET programlama dili Microsoft tarafından .NET platformu için geliştirilmiş, nesne tabanlı bir programlama dilidir. C++ ve Java dillerinden etkilenerek geliştirilmiş bir dildir. Bir anlamda Visual Basic'in görselliği ve C++'ın gücünü içermektedir. Ayrıca, C#, Java gibi, bir Web programlama dilidir.

C#, C++'da olan bellek yönetimi, pointer gibi sorunları çözmek ve daha kolay bir C programcılığı yapmak için geliştirilmiştir. Çöp temizleme (garbage collection), otomatik bellek yönetimi gibi birçok özelliğe sahiptir. C# programlama dili görsellik açısından özellikle Windows Form uygulamalarında çok kuvvetli bir dildir.



C# ve .NET FrameWork

.NET Framework: Microsoft tarafından geliştirilen bir yazılım çatısıdır. Microsoft tarafından geliştirilen, açık İnternet protokolleri ve standartları üzerine kurulmuş bir "uygulama" geliştirme platformudur.

Bir masaüstü uygulamasından bir web tarayıcı uygulamasına kadar her şey bu platform içinde düşünülmüştür ve desteklenmiştir. Bu uygulamaların birbirleriyle ve geliştirildiği ortam farketmeksizin dünyadaki tüm uygulamalarla iletişimi için kolayca web servisleri oluşturulmasına imkân verilmiştir. Bu platform, işletim sisteminden ve donanımdan daha üst seviyede taşınabilir olarak tasarlanmıştır.

Visual Studio ile C# dilinde kodlamaya başlamadan önce kodların yazılacağı dosyada bazı referans ve kütüphanelerin proje içerisinde entegre edilmesi gerekir. Böylelikle proje içerisinde yazılacak komutlar .NET Framework ile uyumlu çalışacaktır.

C# - Namespace (İsim Uzayları - Kütüphaler)

Programlama dillerinde, programcıların işlerini kolaylaştırmak için bir takım hazır kütüphaneler mevcuttur. Bu kütüphanelerden bazıları standart olmakla birlikte bazıları programcılar tarafından sonradan geliştirilmiş ve kullanıcıların hizmetine sunulmuştur.

.NET teki sınıf kütüphaneleri bir dilden bağımsız yapıdadır. Bu yüzden bir projenin içerisine gerekli olan sınıfı çağırabilmek için öncelikle o sınıfın kütüphane dosyasının proje içerisine dahil edilmesi gerekir. İşte bu kütüphane dosyalarına **Namespace (İsim Uzayı)** adı verilir.

Namespace'ler .NET Framework sınıf kütüphanesindeki veri türlerini ve sınıfları kullanabilmek için **C# dilinde using** anahtar sözcüğü ile birlikte kullanılır ve derleyiciye bildirilir.

```
using System; // Burada using deyimi ile System kütüphanesi program içerisine çağrılır.
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

C# Console uygulamalarının geliştirilebilmesi için yukarıda belirtilen 4 kütüphanenin program içerisine çağırılması gerekir. Örneğin Console sınıfına erişilebilmesi için program içerisine System NameSpace i using anahtar sözcüğü ile çağrılır.

C# - CLASS (Sınıflar)

Sınıflar Nesneleri oluşturan temel yapıdır. Sınıf o nesnenin yapabileceği işleri ve nesnenin özelliklerini barındıran kod bloğudur. Sınıfların içerisinde çeşitli metodlar ve o kullanılan sınıfın çeşitli özellikleri yer alır. Ve sınıflar sayesinde her hangi bir platform üzerinde çeşitli işlemler yapılabilir.

Visual Studio ile kodlama yapılabilmesi için öncelikle programcı kendi namespace ini oluşturur.

```
namespace Uygulama
{
        //Sınıfların tanımlanacağı bölüm
}
```

Burada kullanılan süslü parantezler belirtilen isim uzayının nerede başladığını ve nerede biteceğini gösterir.

Not: Visual Studio Yazılım geliştirme arabirimi program dosyasını oluştururken otomatik olarak kullanıcının isim uzayını tanımlar.

İsim uzayı tanımlandıktan sonra artık sınıf tanımlaması yapılabilir. Sınıf tanımlamasının yapılabilmesi için class anahtar sözcüğü kullanılır.

```
class Program
{
        //Alt Programların tanımlanacağı bölüm
}
```

C# - ALT PROGRAMLAR

C# dilinde sınıf oluşturulduktan sonra program yazımına başlayabilmek için alt program tanımlaması yapılır. Alt program tanımlaması yapılabilmesi için void anahtar sözcüğü kullanılır.

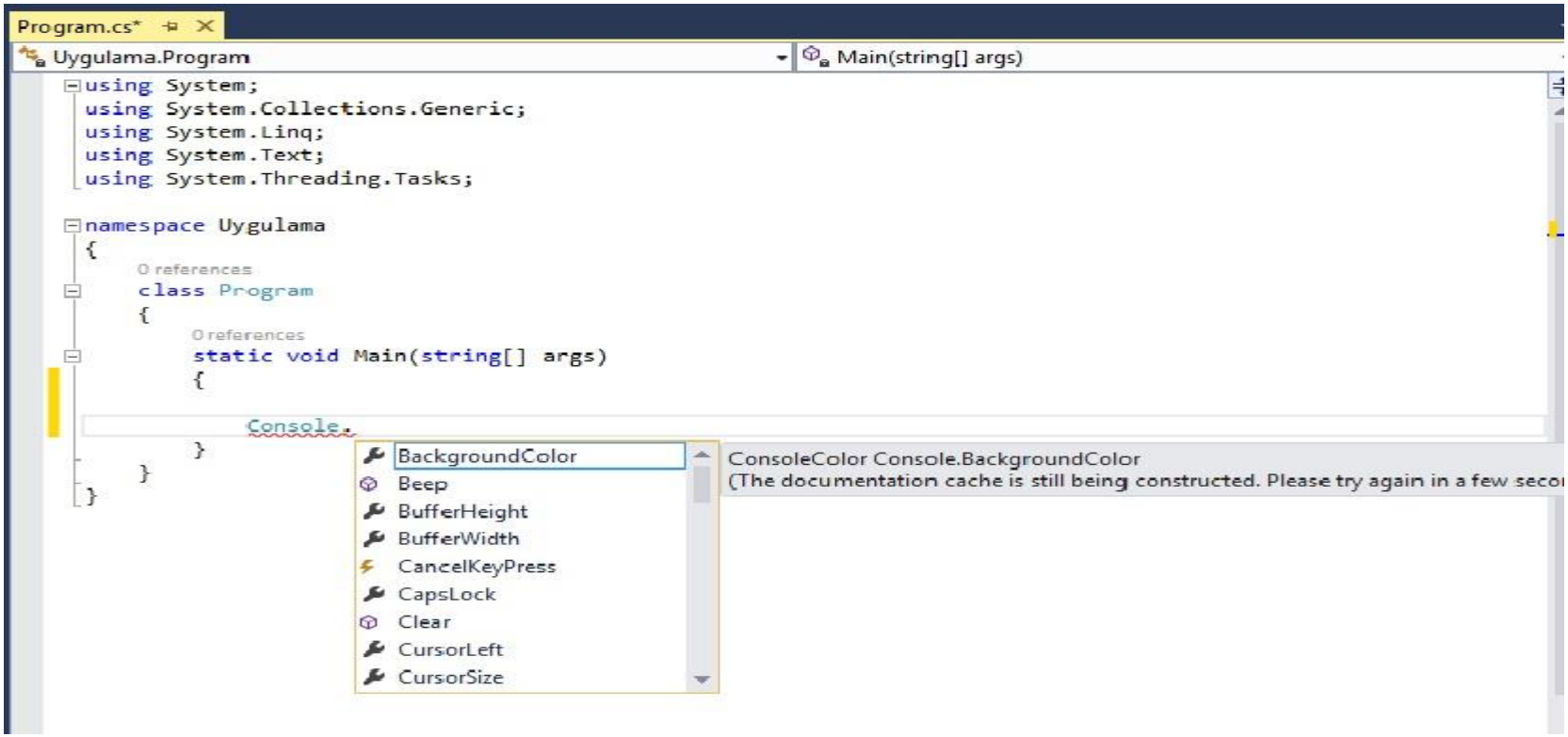
```
namespace Uygulama
{
        class Program
        {
        static void Main(string[] args)
                {
                        // Kodların Yazılacağı bölüm
                }
        void hesapla()
                {
                }
        }

}
```

Burada birden fazla alt program ve fonksiyon oluşturulabilir. Yalnız bu oluşturulan alt programlardan öncelikle çalıştırılacak olanı, alt programın ismi belirler. Yazılan alt programlar içerisinde hangisinin ismi Main ise öncelikli çalıştırılacak alt program bu kısım olur.

C# - CONSOLE Sınıfı

Visual Studio içerisinde C# dili ile görsel özellikleri olmayan Console tabanlı uygulamalar oluşturulabilir. Konsol kullanıcı arayüzü, Windows kullanıcı arayüzü kadar çekici değilken örnekleri konsol arayüzü ile kısıtlamak bize grafik arayüzünün kompleksliği ile uğraşmak yerine öğrenmeye çalıştığımız C# temellerine odaklanma fırsatı sunar.

Konsol Ekranında kod Satırları;



C#- CONSOLE Sınıfı

Console Sınıfı System Kütüphanesinin (namespace) altında yer alır. Console sınıfının çağırılabilmesi için mutlaka;

Using System;

sözdizimi kullanarak System kütüphanesinin çağrılması gerekir. Aksi takdirde Console sınıfına erişim sağlanamaz.

Console sınıfının altında console ekranında çeşitli işlemler yapabilmek için bazı özellikler (property) ve bazı yöntemler (method) yer alır.

Property (Özellik) : Nesnenin yada sınıfın sahip olduğu çeşitli nitelikler.

Metod (Yöntem) : Nesnenin yada sınıfın sahip olduğu çeşitli davranışlar.

Örnek;

```
Console.BackgroundColor=ConsoleColor.White; // Zemin rengi değiştirme
Console.WriteLine(‘’ Merhaba Dünya’’);// Ekrana yazı yazdırma metodu
```

C#- CONSOLE Sınıfı

Console Sınıfı Özellikleri;

Özellik Adı	Açıklama
BackgroundColor	Konsol arkaplan rengini ayarlar.
CapsLock	Capslock un açık olup olmadığının denetlenmesini sağlar
ForegroundColor	Konsol önplan rengini ayarlar
In	Standart giriş işlemleri için kullanılır.
KeyAvailable	Bir tuşun basılmaya uygun olup olmadığını kontrol eder.
Out	Standart çıkış işlemleri için kullanılır.
Title	Konsol başlık çubuğunda görünecek başlığı ayarlar.
WindowHeight	Konsol penceresinin yüksekliğini ayarlar.
WindowWidth	Konsol penceresinin genişliğini ayarlar.

C#- CONSOLE Sınıfı

Console Sınıfı Metodları;

Metod Adı	Açıklama
Beep()	Konsol hoparlöründen beep sesi çıkarmak için kullanılır.
Clear()	Konsol penceresini temizler.
Read()	Konsol ekranında standart veri girişi yapmak için kullanılır.
ReadLine()	Konsol ekranında standart veri girişi yapmak için kullanılır.
ReadKey()	Konsol ekranında sadece karakter girişi yapmak için kullanılır.
ResetColor()	Ön ve arkaplan konsol renklerini standart haline döndürmek için kullanılır.
SetCursorPosition()	Konsol ekranında imleç konumunu ayarlar.
SetWindowSize()	Konsol ekranının yüksekliğini ve genişliğini istenilen boyutlarda ayarlar.
Write()	Konsol ekranında standart çıktı işlemi yapmak için kullanılır.
WriteLine()	Konsol ekranında standart çıktı işlemi yapmak için kullanılır.

DEĞİŞKENLER

Tür	Boyut	Kapasite	Örnek
byte	1 bayt	0, ..., 255 (tam sayı)	byte a=5;
sbyte	1 bayt	-128, ..., 127 (tam sayı)	sbyte a=5;
short	2 bayt	-32768, ..., 32767 (tam sayı)	short a=5;
ushort	2 bayt	0, ..., 65535 (tam sayı)	ushort a=5;
int	4 bayt	-2147483648, ..., 2147483647 (tam sayı)	int a=5;
uint	4 bayt	0, ..., 4294967295 (tam sayı)	uint a=5;
long	8 bayt	-9223372036854775808, ..., 9223372036854775807 (tam sayı)	long a=5;
ulong	8 bayt	0, ..., 18446744073709551615 (tam sayı)	ulong a=5;
float	4 bayt	$\pm 1.5*10^{-45}$, ..., $\pm 3.4*10^{38}$ (reel sayı)	float a=5F; veya float a=5f;
double	8 bayt	$\pm 5.0*10^{-324}$, ..., $\pm 1.7*10^{308}$ (reel sayı)	double a=5; veya double a=5d; veya double a=5D;
decimal	16 bayt	$\pm 1.5*10^{-28}$, ..., $\pm 7.9*10^{28}$ (reel sayı)	decimal a=5M; veya decimal a=5m;

Tür	Boyut	Açıklama	Örnek
char	2 bayt	Tek bir karakteri tutar.	char a='h';
string	Sınırsız	Metin tutar.	string a="Buraya Bir Metin Gelecektir.";

C#- CONSOLE Sınıfı

Write ve WriteLine Metodları

Write Metodu: Kendisine gönderilen değeri konsol ekranında aynı satıra yazdırmak için kullanılan bir metoddur.

```
Console.Write(‘’Algoritmalar’’);
```

WriteLine Metodu : Kendisine gönderilen değeri konsol ekranında satır sonu karakterini de ekleyerek bir alt satırdan itibaren yazdırmak için kullanılan bir metottur.

```
Console.WriteLine(‘’Algoritmalar’’);
```

C#- CONSOLE Sınıfı

Write ve WriteLine Metodlarında Kullanılan Kaçış Karakterleri

C# Konsol uygulamalarının ilk ve en basit konusu olan Write() ve WriteLine() metodları ekrana en basit haliyle yazı yazdırmaya yarar. Aralarında ki tek fark imleç farkıdır. Yani yazdıktan sonra yeni satıra geçer. Yazılar çift tırnak arasında gösterilir.

C ve C++ ta olduğu gibi (escape) karakterlerinin kullanımına olanak verir.

C# da bir kaçış karakteri tanımlaması yapabilmek için metnin içerisinde ters bölü \ ibaresi kullanmak gerekir.

C#- CONSOLE Sınıfı

Write ve WriteLine Metodlarında Kullanılan Kaçış Karakterleri

Karakter	Anlamı
\a	Ses üretir(alert)
\b	imleci bir sola kaydır(backspace)
\f	Sayfa atla. Bir sonraki sayfanın başına geç(formfeed)
\n	Bir alt satıra geç(newline)
\r	Satır başı yap(carriage return)
\t	Yatay TAB(Horizontal TAB)
\v	Dikey TAB(vertical TAB)
\"	Çift tırnak karakterini ekrana yaz
\'	Tek tırnak karakterini ekrana yaz
\\	\ karakterini ekrana yaz

C#- CONSOLE Sınıfı

Read, ReadLine ve ReadKey Metodları

Programlar çoğu zaman bizim girdiğimiz verilere göre işlemlerini yürütürler. C# konsol uygulamalarında veri girişi daima klavyeden yazı yazma ile gerçekleşir.

Genellikle konsol uygulamalarında program kullanıcıdan bir şeyler girmesini ister ve kullanıcının verdiği yanıta göre kullanıcının ne demek istediğini anlamaya çalışır.

Konsol ekranında klavyeden giriş yapmak için Console sınıfının altında yer alan Read ve ReadLine yöntemleri kullanılır.

Console.WriteLine ve Console.Write ekrana çıktı vermek için kullanılırken, Console.Read ve Console.ReadLine yöntemlerinde giriş argümanı olmamasına rağmen bu yöntemlerden dönen dönüş değerleri program içerisinde kullanılabilir. Yalnız her iki metodda da veri girişinin geçerli olabilmesi için **enter** tuşuna basılması gerekir.

C#- CONSOLE Sınıfı

Read, ReadLine ve ReadKey Metodları

Read Metodu: Kullanıcının klavyeden giriş yapmasını sağlar tek karakter okur ve geriye tam sayı tipinde bir değer döndürür.

```
int sayi=Console.Read();
```

ReadLine Metodu : Kullanıcının klavyeden giriş yapmasını sağlar ve enter tuşuna basıldıktan sonra verinin metinsel bir ifade olarak döndürülmesini sağlar.

```
string adi=Console.ReadLine();
```

ReadKey Metodu: Kullanıcının klavyeden tek karakterlik veri girişi yapmasını sağlar. Bu metodda enter tuşuna basılması beklenilmez. Ve buradan alınan değer **ConsoleKeyInfo** sınıfından üretilen bir değere aktarılır.

```
ConsoleKeyInfo karakter=Console.ReadKey();
```

DEĞİŞKENLER

Değişken tanımlama formatı;

```
veritipi veriAdı1, veriAdı2, … ;
```

```
1 int sayi1; /* bir tam sayıyı depolar (152) */
2 float sayi2=5.6f; /* bir ondalıklı sayıyı depolar (65.324) */
```

Atanmamış Yerel C# Değişkenler

C#, değer atanmamış değişkeni kullanmanıza izin vermez. Bir değişkeni kullanmadan önce, değişkene bir değer atamak zorundasınız; aksi takdirde programınız derlenmeyebilir. Bu koşul, Definite Assignment Rule (Kesin Atama Kuralı) olarak adlandırılır. Örneğin, aşağıdaki ifadeler, yas değişkenine bir değer atanmadığı için derleme hatası verecektir:

?

```
1 int yas;
2 Console.WriteLine(yas); //derleme hatası
```

Kapalı Türde Yerel C# Değişkenler

Bir değişkene atadığınız değerin, değişken ile aynı türde olması gerektiğini bahsetmiştik. Örneğin, int türünde değişkene sadece int türü değer atayabilirsiniz. C# derleyici, bir değişkene başlangıç değeri atamak için kullanılan ifadenin türünü hızlı bir şekilde ortaya çıkarabilir ve değişkenin türü ile aynı değilse size bildirebilir. Ayrıca, C# derleyicisinden bir ifadedeki değişkenin türünü bulmasını ve aşağıdaki gibi tür yerine var anahtar sözcüğünü kullanarak değişken bildirdiğinizde, bulduğu türü kullanmasını isteyebilirsiniz:

```
1 var myVariable = 99;
2 var myOtherVariable = “Hello”;
3 var yetAnotherVariable; // Hata
```

C# Değişkenler – Önek ve Sonek

++, artırma ve **--**, azaltma işleçlerini, değişkenden önce ve sonra yerleştirebilirsiniz. İşleç simgesinin değişkenden önce yer alması, işlenenin önek (prefix) biçimi olarak ve değişkenden sonra yer alması da sonek (postfix) biçimi olarak adlandırılır. Örneğin:

```
1 count++; // sonek ++count; // önek
2 count--; // sonek  - - count; //önek
```

ÖRNEK

```
1 int x = 42;
2 Console.WriteLine(x++);
3 // x şimdi 43'tür, 42 yazılır
4 x = 42;
5 Console.WriteLine(++x);
6 // x şimdi 43'tür, 43 yazılır
```

C#- CONSOLE Sınıfı

BackgroundColor, ForegroundColor, Clear

C# console uygulamalarında yazıya ve arkaplana rengi verilebilir. Console metodu altında BackgroundColor ve ForegroundColor kullanılır. BackgroundColor arkaplan rengini değiştirmek için, ForegroundColor ise yazı rengini değiştirmek için kullanılır.

Renk ataması ise ConsoleColor ile yapılmaktadır.

Renkler: Black DarkBlue DarkGreen DarkCyan DarkRed DarkMagenta DarkYellow Gray DarkGray Blue Green Cyan Red Magenta Yellow White Black DarkBlue DarkGreen DarkCyan DarkRed DarkMagenta DarkYellow Gray DarkGray Blue Green Cyan Red Magenta Yellow White

```
Console.BackgroundColor = ConsoleColor.Blue;
        Console.Clear();
        Console.ForegroundColor = ConsoleColor.White;
        Console.Write("Press any key to continue");
        Console.ReadKey();
```

C# Diziler

C# Diziler, Şimdiye kadar program içinde tanımladığımız değişken sayısı birkaç değişkeni geçmedi. Ancak çoğu zaman bu böyle olmaz. Bazı programlarda 200-300 değişkene kadar ihtiyaç duyabiliriz. Bunların hepsinin teker teker tanımlanması oldukça zahmetlidir. İşte bu yüzden programlama dillerinde dizi diye bir kavram vardır. Aslında bir dizi, birbiriyle ilişkili değişkenlerin oluşturduğu bir gruptan başka bir şey değildir.

```
int[] dizi=new int[10];
```

Veya

```
int[] dizi;
dizi=new int[10];
```

Dizilere sistematik erişim Foreach döngüsü

```
int[] dizi={3,2,6,7};
foreach(int eleman in dizi)
Console.WriteLine(eleman);
```

ÖNEMLİ

Foreach döngüsü sadece dizi elemanlarını okuma için kullanılır. Dizi elemanları üzerinde düzenleme işlemlerine izin verilmez. Bu tür engelleme ReadOnly (Sadece okunabilir) olarak bilir.

Dizi Methotları

Diziler üzerinde işlevsel olarak kullanılabilecek birçok methot vardır. Tüm dizi fonksiyonlarına Microsoft Msdn üzerinden erişebilirsiniz. Sık kullanılan dizi fonksiyonlarından başlıcaları;

GetLength()
Bir dizideki eleman sayısını int türünden veren methottur.

```
int[] dizi={1,4,7,9};
Console.Write(dizi.GetLength(0));// 4
```

Çok boyutlu dizilerde herbir dizi elemanın değerinin indeks numarası belirtilmelidir.

```
int[,] dizi={{2,4,2},{7,10,4},{7,12,6},{2,1,12}};
byte a=dizi.GetLength(1);
Console.WriteLine(a);
```

Dizi kopyalama

Daha önceden de belirtildiği gibi bir dizinin boyutu belirlendikten sonra boyutu değiştirilemez. Bu sebeple bir dizi daha büyük bir diziye dönüştürülmek istenirse mevcut dizi daha büyük boyutlu bir diziye kopyalanır.

```
int[] dizi1={1,2,3,4};
int[] dizi2=new int[10];
dizi1.CopyTo(dizi2,3);
```

Burada dizi1'in 3. Elemanından itibaren dizi2'ye kopyalama işlemi gerçekleşir. İstenirse dizi sınıf (Array) kullanılarak bir dizinin sadece belirli bir bölümü kopyalanabilir.

```
int[] dizi1={1,2,3,4};
int[] dizi2=new int[10];
Array.Copy(dizi1,dizi2,3);
```

Bu örnekte dizi1'in 3. Elemanından itibaren dizi2'ye kopyalama işlemi gerçekleşir.

Array.IndexOf();

Dizi içinde harf ya da kelime aramamıza yarar. Eğer harf ya da kelimeyi bulursa bulduğu ilk indexi gönderir.Bulamazsa geriye -1 gönderir.

Array.IndexOf(dizi1, 2);
2 ifadesini dizi1 dizisinde arar. Bulduğu ilk indexi gönderir.

Array.IndexOf(sayilar, 2, 3);
Bu methot ise 3.elemandan itibaren arama yapar.

Array.Reverse();

Dizinin elemanlarını ters çevirir. Ancak sadecetek boyutlu diziler için kullanılabilir.

```
Array.Reverse(dizi);
```

Array.Sort();

Dizinin elemanlarını küçükten büyüğe doğru olacak şekilde sıralar.

```
Array.Sort(dizi);
```

Array.Clear();

Bu kod dizi dizisinin 1. indeksinden itibaren 3 indeksini sıfırlar (varsayılan değere döndürür).

```
Array.Clear(dizi, 1, 3);
```

Kaynaklar

https://www.kodyaz.net/c-diziler/