

1. VISUAL STUDIO 2022 KURULUMU VE TEMEL GİRDİ-ÇIKTI KOMUTLARI

Bölümle İlgili Özlü Söz

Önce sorunu çözün, ardından kodu yazın.

John Johnson

Kazanımlar

1. Visual Studio 2022 editörünün kurulumunu yapabilir.
2. Visual Studio 2022 editörüyle çözüm ve proje oluşturabilir, daha önce kullanılan projeleri açabilir.
3. Konsol uygulaması hakkında bilgi sahibi olur.
4. Kaynak kodu derleyebilir ve derlenen programı çalıştırabilir.
5. C# konsol girdi ve çıktı komutlarını bilir, bu komutları kullanarak kullanıcıdan veri alabilir ve ekrana çıktılayabilir.

Birlikte Düşünelim

Bilgisayar programı geliştirmek için özel bir yazılım kullanılıyor mu?

Editör kullanmanın programcıya ne gibi faydası vardır?

Bilgisayar kodları nasıl çalıştırılabilir program haline gelmektedir?

Derleme nedir? Derlenen program nereye kaydedilir ve nasıl çalıştırılır?

Konsol ekranı nedir?

Konsola bir metin nasıl yazdırılır ve kullanıcı verisi nasıl alınır?

Kaynak kod, kod bloğu, deyim ve ifade terimleri nedir?

Başlamadan Önce

Bir problemin çözümüne yönelik algoritma tasarlandıktan sonra bir programlama dili ile bilgisayarın anlayacağı dile yani çalıştırılabilir forma dönüştürülür. Bu işe programlama veya kodlama denir.

Geliştiriciler yani programcılar kodlama sürecinde daha verimli ve hızlı olmak adına bir kodlama editörü kullanırlar.

Visual Studio 2022 yazılımı .NET tabanlı dillerde programlama yapılmasını sağlayan bir editördür. Kodlama sürecinde oluşan hataları programcıya listeler, kodun yazımı sırasında önerilerde bulunur, kaynak kodu bilgisayarın anlayacağı forma dönüştürür, hata ayıklama ve takım halinde kod yazımını sağlar.

Bu kapsamda Visual Studio 2022 editörünün kurulumu, en çok kullanılan temel fonksiyonları öğretilecektir. Kaynak kodların bilgisayarda çalıştırılabilir forma dönüştürülmesi sürecinin nasıl olduğu, derleme kavramı, derlenen programın nasıl çalıştırılacağı öğrenilecektir.

Bu ders boyunca programlamanın temelleri C# dili kullanılarak hiçbir görsel bileşeni olmayan konsol ekranı üzerinden öğretilenektir. Konsol uygulamaları modern görünümdeki yazılımlardan son derece uzak olsalar da öğrencinin sadece dile ve dilin yapılarına odaklanarak programlama becerisi edinmesini kolaylaştırmaktadır.

Konsol ekranına metin, değır veya değışkenin nasıl ıktılanacağı ve program kapsamında ihtiyaç duyulan kullanıcı verisinin nasıl alınacağı kısaca temel girdi-ıktı komutları öğrenilecektir.

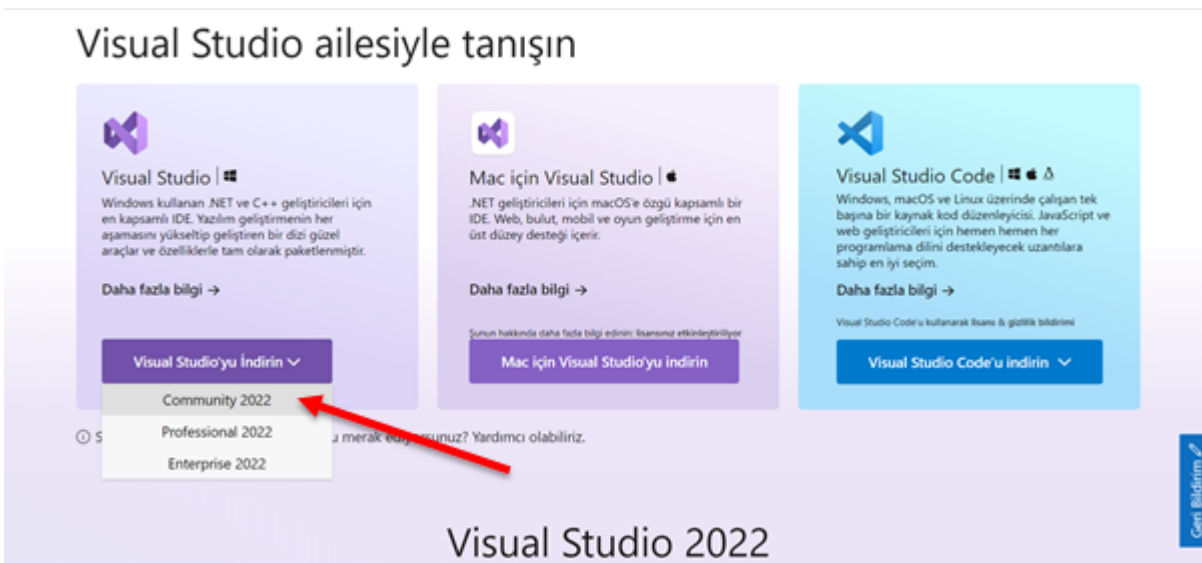
1.1. Visual Studio 2022 Kurulumu

C# dilinde program geliştirmek için çeşitli editörler kullanılmaktadır. Bunlar arasında Visual Studio 2022, Visual Studio Code, JetBrains Rider, Atom, SlickEdit editörleri sayılabilir. Ayrıca C# dilinde sadece konsol uygulaması geliştirmek için hiçbir editör kurulumuna gerek kalmadan online C# derleyicileri de kullanılmaktadır.

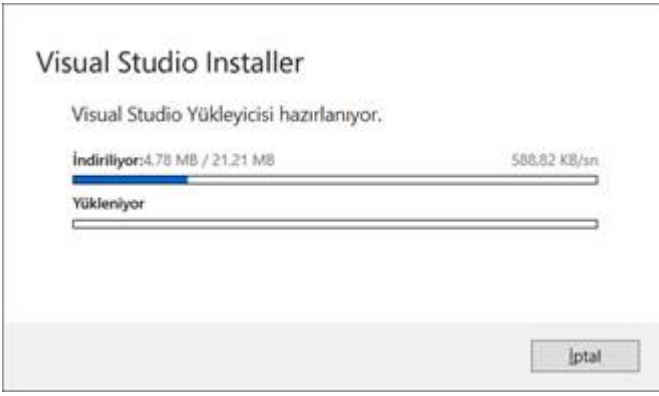
Esasında basit bir Notepad kullanarak da C# dilinde kodlama yapılabilir. Bütünleşik Geliştirme Ortamı (*Integrated Development Environment - IDE*) veya kısaca editör dediğimiz Visual Studio 2022 gibi yazılımlar kodlama yaparken programcının hatalarını eşzamanlı olarak gösterirler, kodun daha hızlı yazılmasını sağlamak için otomatik tamamlama özelliğiyle öneriler sunarlar, kaynak kodu makine diline çevirirler yani derlerler, programdaki olası hataların takibini ve ayıklanmasını sağlarlar, kodun test edilmesi için gereken araç ve ortamı sağlarlar, son haline gelen yazılımları çalışacak ortamlara konuşlandırırılar, versiyonlama hizmeti sunarlar ve takım halinde kodlama yapılmasını sağlarlar. Bunlar ve benzeri çeşitli avantajlarından ötürü editör kullanmak programcının en temelde üretkenliğini ve verimliliğini artırır ve maliyetleri düşürür.

Kitap boyunca C# dilindeki kodlar Visual Studio 2022 editörü ile yazılmıştır.

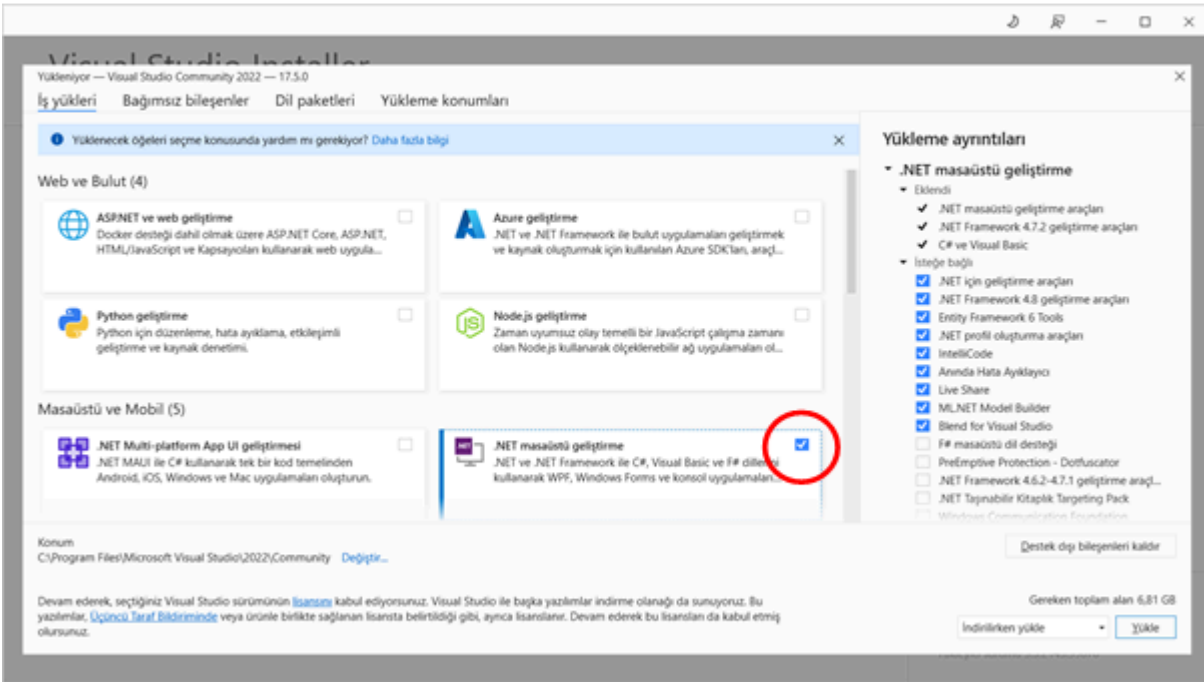
Visual Studio 2022 editörü <https://visualstudio.microsoft.com/tr/> adresinden indirilebilir. Visual Studio 2022 editörünün Windows ve Mac işletim sistemleri için ayrı sürümleri bulunmaktadır (Microsoft, 2023a). Windows işletim sistemleri için Topluluk, Profesyonel ve Kurumsal versiyonları bulunmaktadır. Topluluk sürümü ders kapsamında kullanılacak sürüm olduğundan web sayfasındaki açılır listeden **Community 2022** seçeneğine tıklandığında otomatik olarak yükleme dosyası indirilecektir.



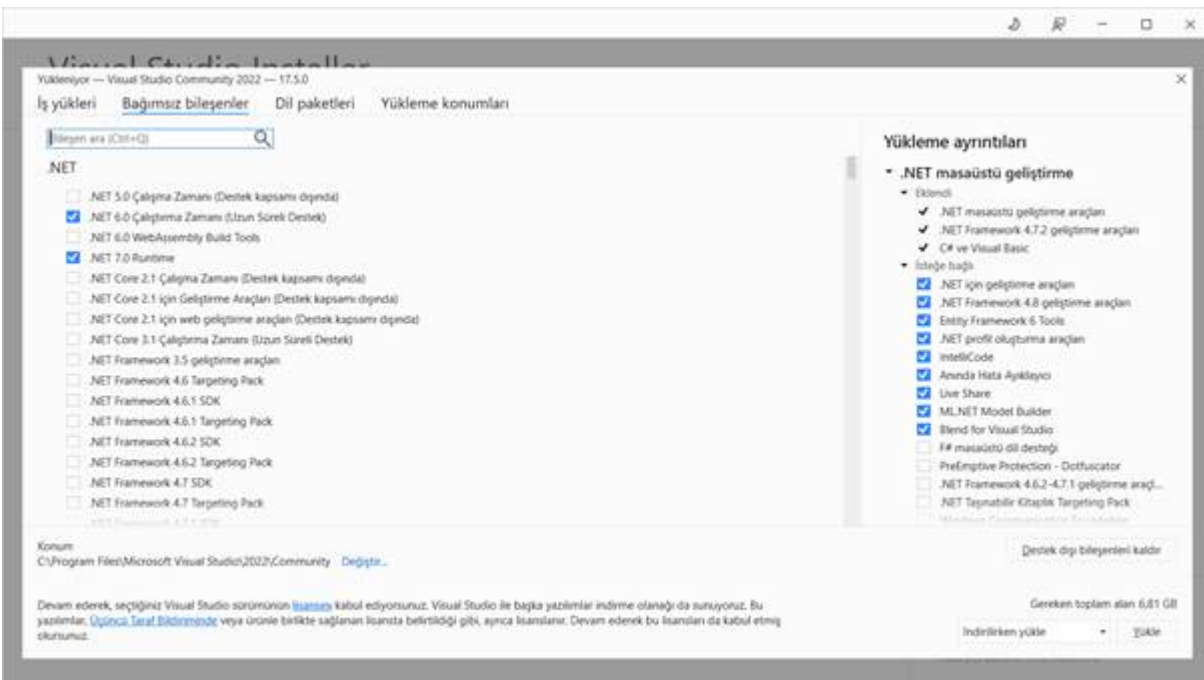
İndirilen **VisualStudioSetup.exe** isimli dosyaya çift tıklandığında kurulum için izin isteyen ekran gelecektir. Gereken izinler verildikten sonra yükleyici için gereken modüller internette indirilecektir.



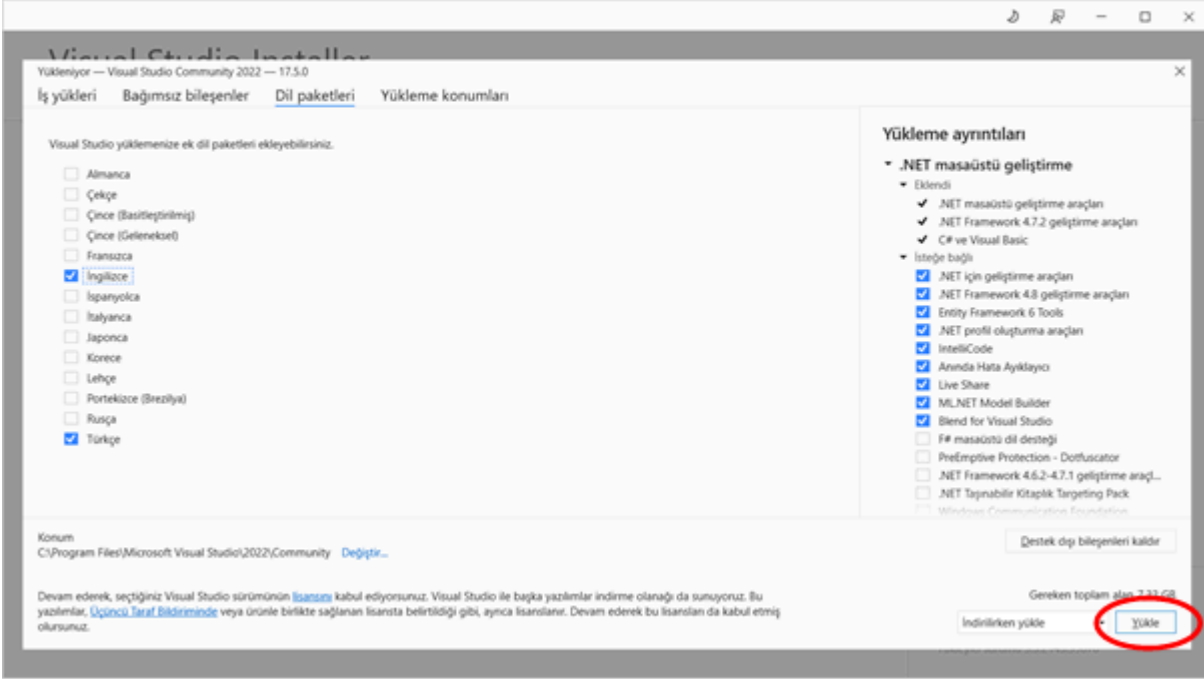
Daha sonra Visual Studio 2022 içinde yüklenilecek modüllerin seçileceği ekran gelecektir. Bu ekranda **.NET masaüstü geliştirme** seçeneği işaretlenmelidir.



Bağımsız bileşenler sekmesinde .NET 7.0 mutlaka seçili olmalıdır.

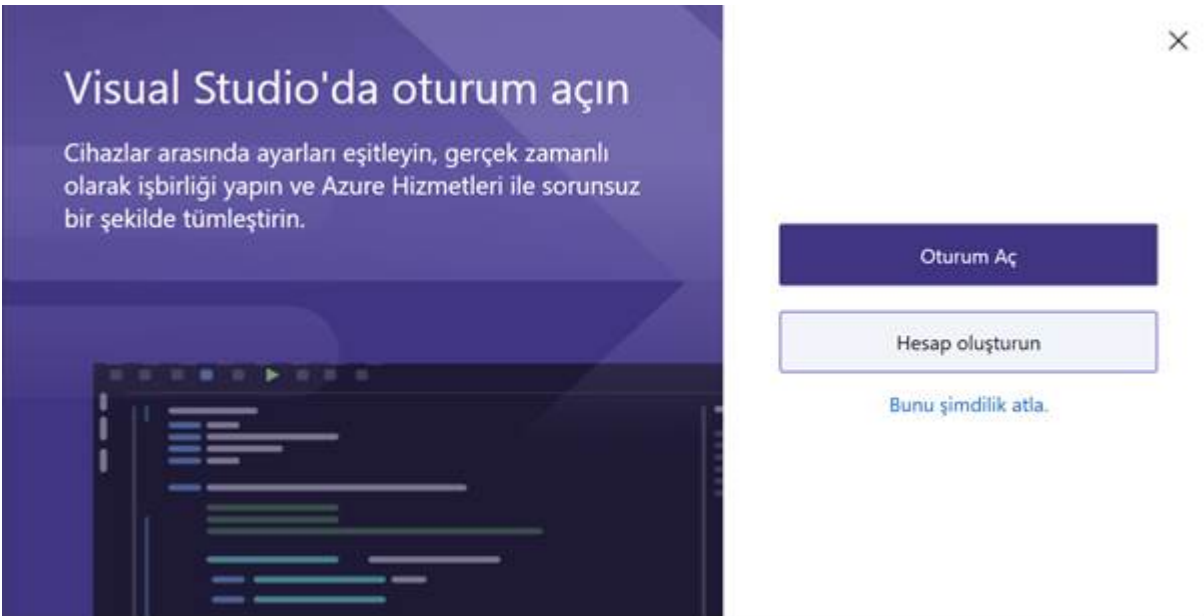


Dil paketleri sekmesinde arzu edilen diller seçilebilir.

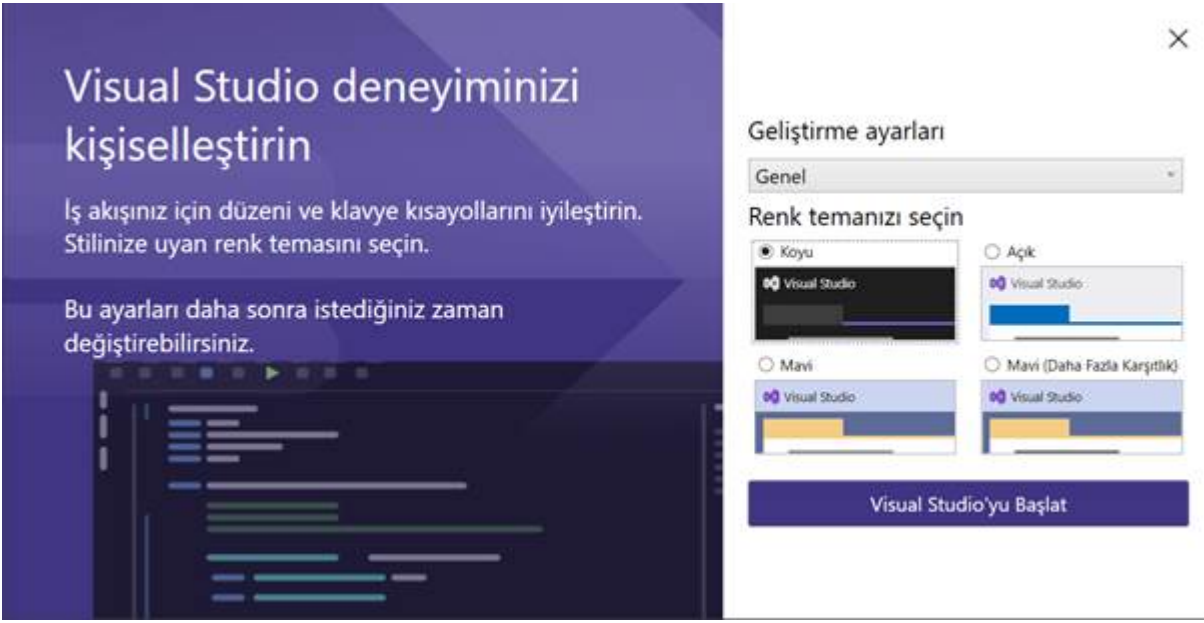


İstenilen bileşenler seçildikten sonra **Yükle** butonuna basılarak kurulum başlatılır. Visual Studio 2022 editörünün son sürümü Şubat 2023 itibariyle 17.5.0 versiyonudur. Eğer halihazırda Visual Studio 2022'nin eski bir sürümü kullanılıyorsa gelen ekranda **Güncelleştir** butonu tıklanmalıdır.

Yükleme bittiğinde Microsoft hesabıyla oturum açma veya hesap oluşturma ekranı gelecektir. Bu adımı atlanarak sonraki ekrana geçilebilir.



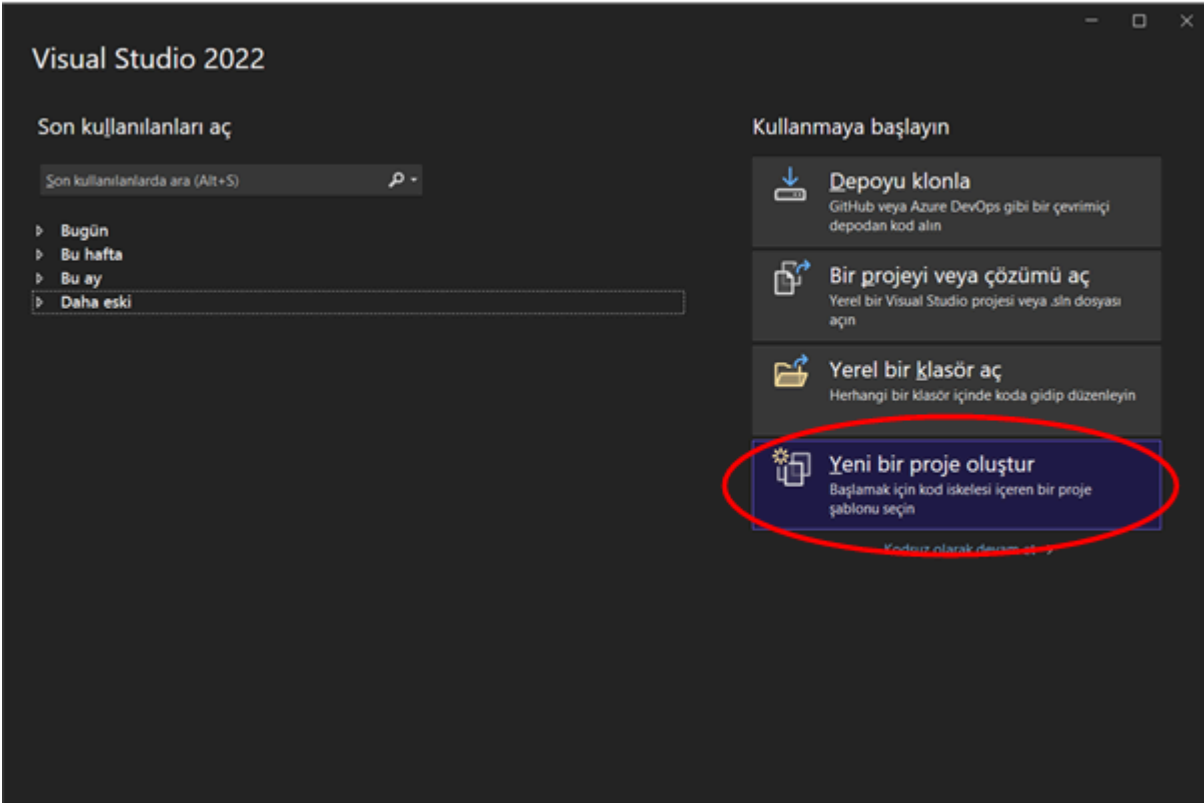
Sıradaki ekranda editörü kullanırken tercih edilecek temalardan birinin seçilmesi istenmektedir. Arzu edilen tema seçildikten sonra **Visual Studio'yu Başlat** butonu tıklanıp proje oluşturma ekranının açılması sağlanmalıdır.



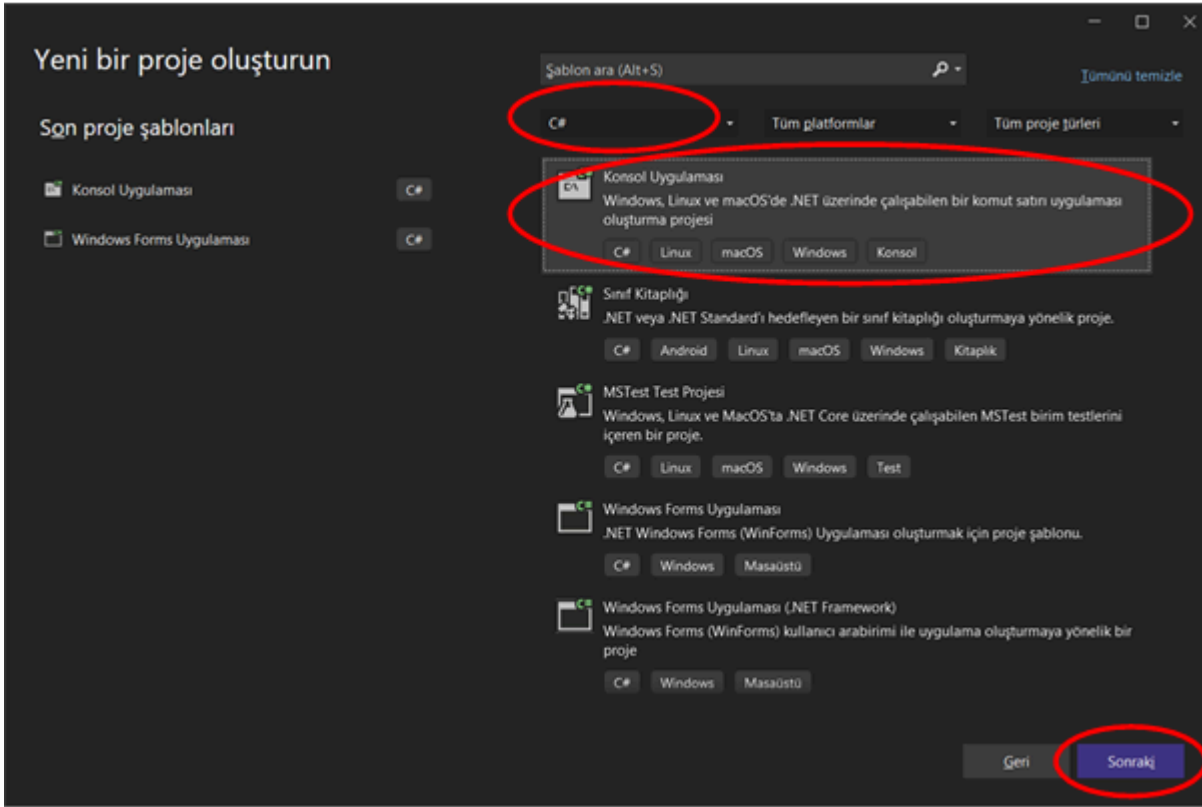
1.2. İlk Program: Merhaba Dünya!

Programlama eğitiminin başlangıcında hangi dilin öğrenildiğinden bağımsız olarak ekrana **Merhaba Dünya** yazdırılır. Bu hem programlama dili için gereken dosyaların kurulduğunu hem de editörün sağlıklı bir şekilde çalıştığını test etmek amaçlı yapılır.

Öncelikle masaüstündeki veya başlat çubuğundaki Visual Studio 2022 programının simgesine çift tıklayarak çalıştırılır. Aşağıdaki ekranda “**Yeni bir proje oluştur**” seçeneğini tıklayalım.



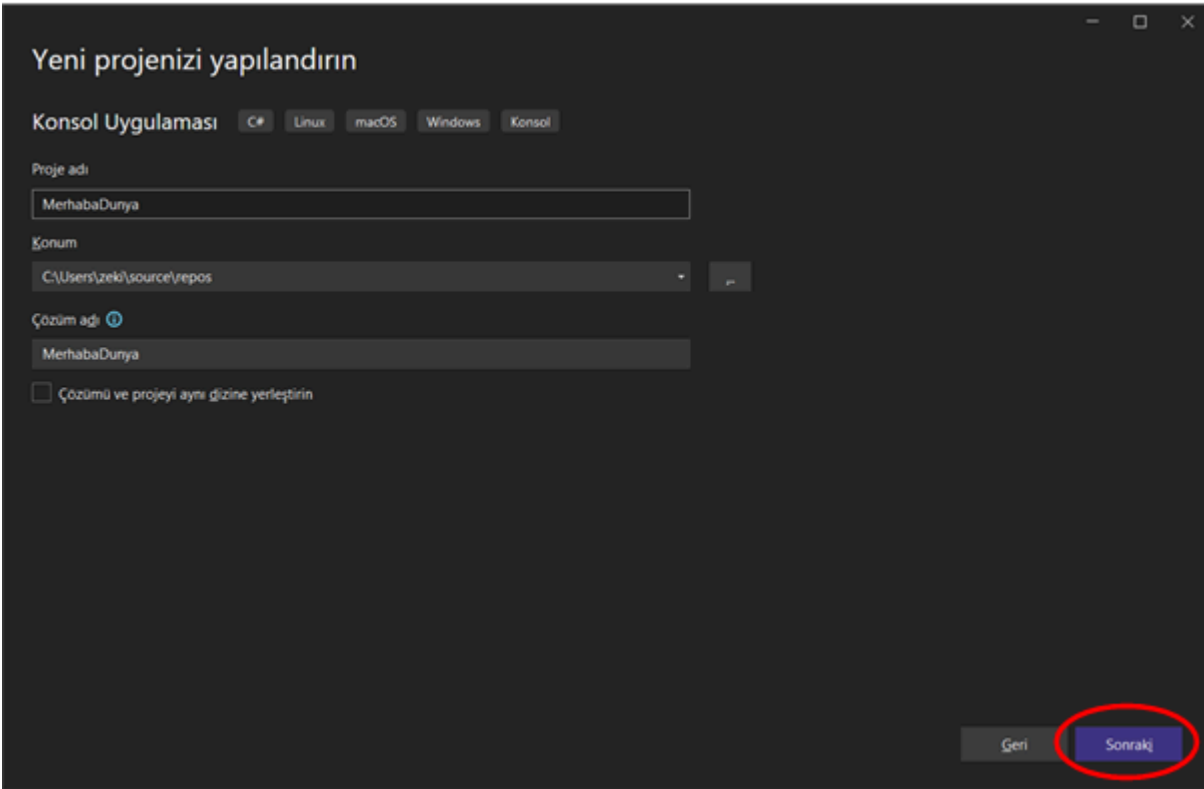
Gelen ekranda dillerden **C#** dilini seçelim ve uygulama türlerinden **Konsol Uygulaması** seçeneğine tıklayıp **Sonraki** tuşuna basalım.



Sonraki ekranda projeye bir isim vermemiz istenecektir. İlk programımıza **MerhabaDunya** ismini verip **Sonrakı** tuşuna basalım. Bu ekranda **Konum** alanında proje ve buna bağlı kaynak kod dosyalarımızın nereye kaydedileceği de sorulmaktadır. Windows işletim sisteminde C# projeleri kullanıcı klasörünün altında source\repos klasörüne kaydedilir. Kullanıcı adından bağımsız olarak başlat çubuğuna %userprofile%\source\repos yazılıp Enter tuşuna absıldığında her bir projenin kendine ait klasörleri görüntülenecektir.

Bu ekrandaki son seçenekte kullanıcıdan **Çözüm adı** yazması istenmektedir. Varsayılan olarak projeye yazılan isim otomatik olarak çözüm adı olarak da verilmektedir. Bu noktada Proje ve Çözüm (*solution*) adındaki iki farklı kavramın ne olduğu bilinmelidir. Çözüm, bir veya birden fazla projeyi barındıran en geniş kapsamlı dosyalama yapısıdır (Aktaş, 2017).

Bir işletme için yazılım geliştirme yaptığımızı farz edelim. Çözüm ismi olarak işletmenin adını verelim. İşletmenin muhasebe departmanı için geliştirilecek form ekranlarını, kodları vb. "işletme çözümünün içinde" Muhasebe isimli projede geliştirelim. Bu şekilde işletmenin her bir departmanı için aynı çözüm içinde ayrı birer proje oluşturulabilir.



Yeni projenizi yapılandırın

Konsol Uygulaması C# Linux macOS Windows Konsol

Proje adı
MerhabaDunya

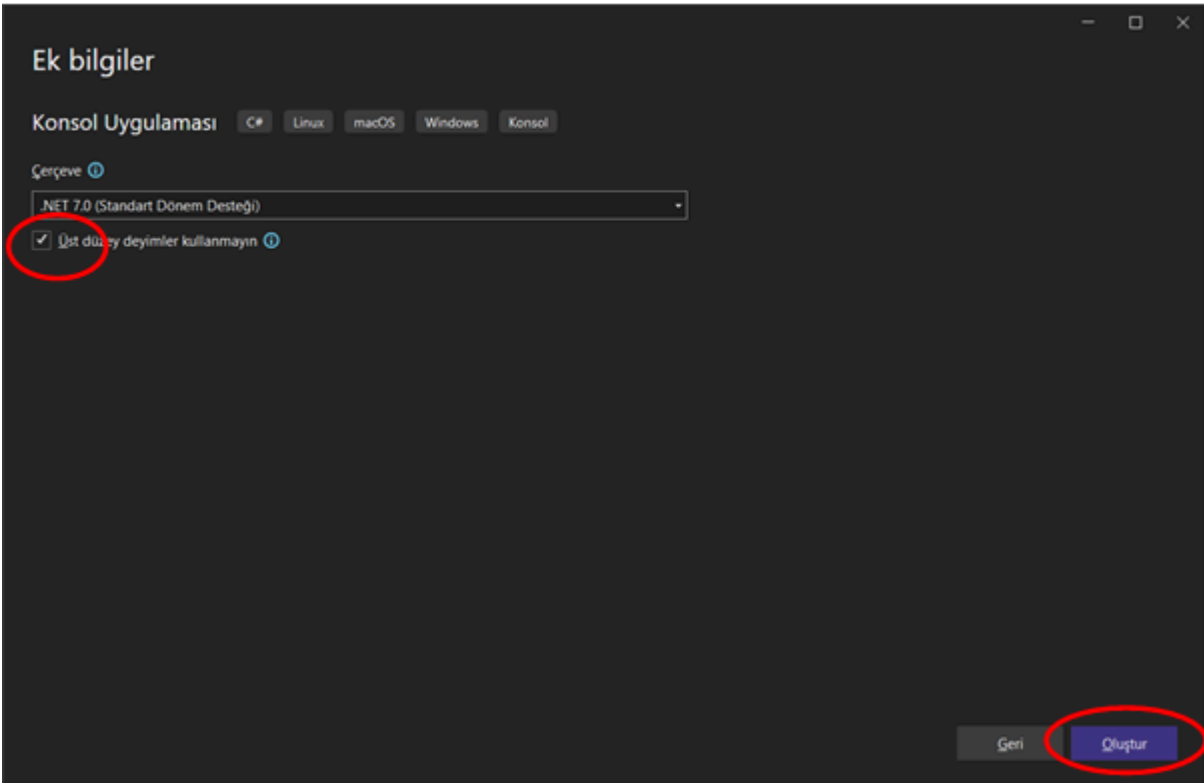
Konum
C:\Users\zeki\source\repos

Çözüm adı ⓘ
MerhabaDunya

☐ Çözümü ve projeyi aynı dizine yerleştirin

Geri Sonraki

Sonraki ekranda kodlarımızın çalıştırılacağı .NET sürümü istenmektedir. Bu ekranda en güncel sürüm olan .NET 7.0 seçip **Oluştur** butonuna tıklayalım. Bu ekranda ayrıca “Üst düzey deyimlerin” kod mimarisinde olup olmaması sorulmaktadır. Bu seçeneği şimdilik işaretleyelim, konunun ayrıntısı ilerleyen sayfalarda açıklanacaktır.



Ek bilgiler

Konsol Uygulaması C# Linux macOS Windows Konsol

Çerçeve ⓘ
.NET 7.0 (Standart Dönem Desteği)

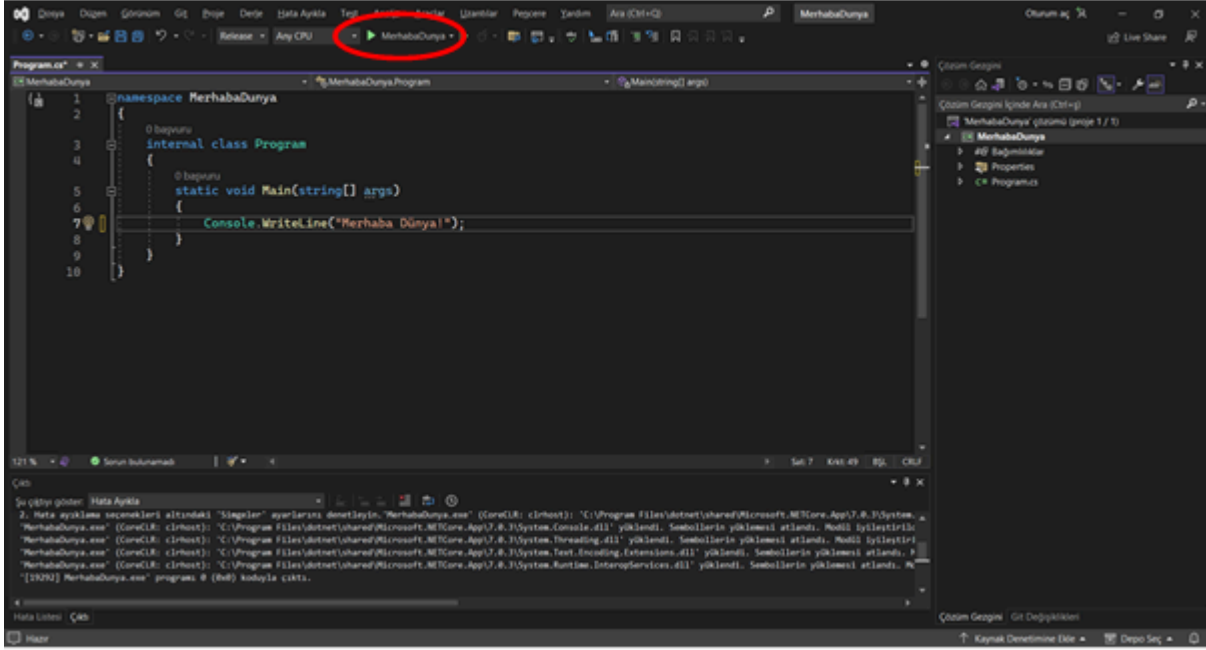
☒ Üst düzey deyimler kullanmayın ⓘ

Geri Oluştur

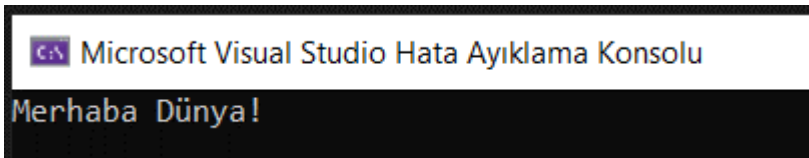
Sonraki ekran, oluşturduğumuz ilk konsol uygulamasının varsayılan kod satırlarını içeren editör ekranıdır. MerhabaDunya isimli projemizin kodları aşağıda verilmiştir. Varsayılan olarak ekranda yer alan kaynak kodun 7. satırında çift tırnak içinde verilen turuncu renkli metni Merhaba Dünya! olarak değiştirelim.

```
Console.WriteLine("Merhaba Dünya!");
```

İlk programımızı kırmızı oval şekil içinde yer alan **yeşil renkli üçgen butona** tıklayarak veya **F5** tuşuna basarak çalıştıralım.



Program çalıştığında aşağıdaki ekran gösterilecektir.



Program, Merhaba Dünya! metnini çıkılamıştır. Kurulum ve proje oluşturma adımları sorunsuz ilerlediğinde bu ekran gösterilecektir. Böylece bundan sonra C# dilinde sorunsuz kodlama yapılabilir.

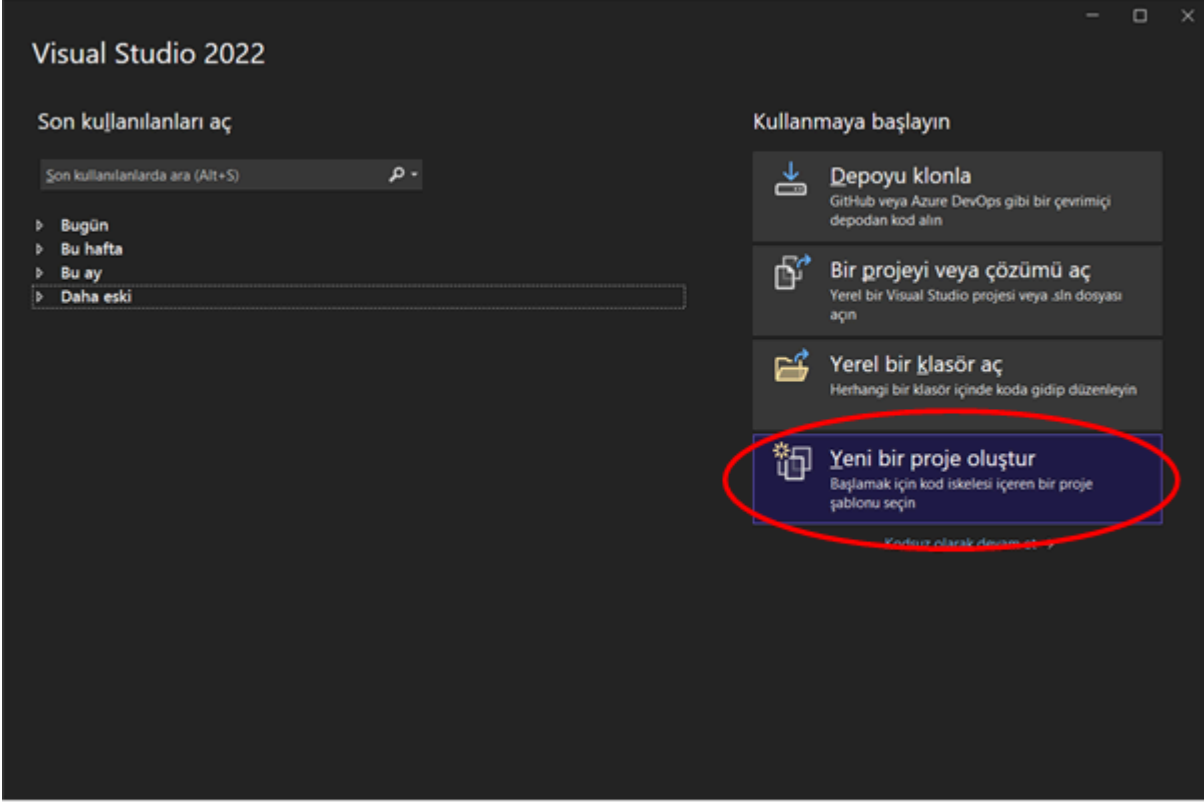
F5 butonuna basıldığında veya içi yeşil renkli üçgene tıklandığında öncelikle kodda bir problem olup olmadığı, kodun dilin gramerine uygun yazılıp yazılmadığı kontrol edilir. Kaynak kodda bir hata yoksa sıradaki iş C# dilindeki bu kod satırlarının makinenin anlayacağı dile yani makine diline çevrilmesidir. Kaynak kodun makine diline çevrilmesi işlemine **derleme (compiling)** denir. **Derleyici (compiler)** adı verilen yardımcı programlar bu işlevi yerine getirirler. Makine diline çevrilen yani çalıştırılabilir/yürütülebilir (*executable*) formdaki dosya Windows işletim sisteminde **.exe** uzantısıyla oluşturulur.

1.3. İkinci Program: Hello World

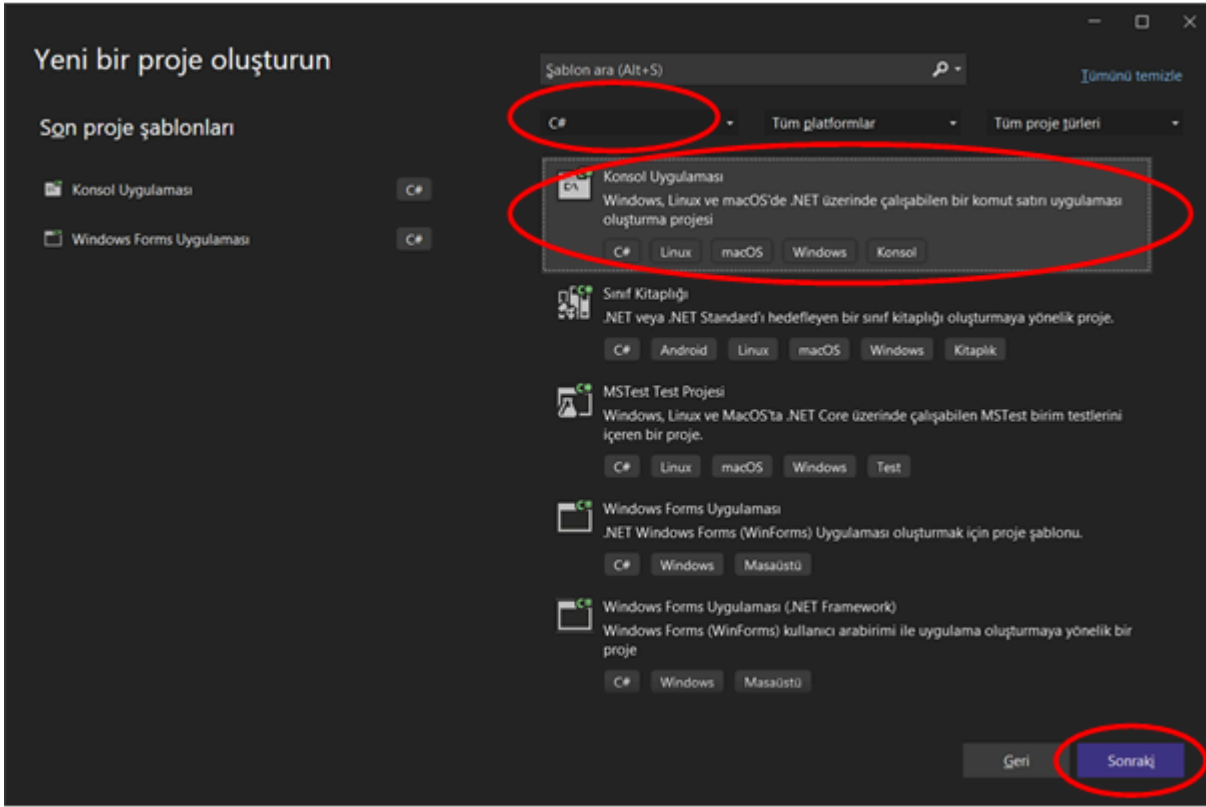
C# dili 9.0 versiyonu ve .NET 5.0 sürümüyle birlikte basit bir şekilde ekrana çıktı almak için kaynak kodun üst kısmındaki namespace, class, Main metodunu kaldırıp doğrudan kodun yazımına izin veren mimariyi geliştirmiştir. **Üst düzey deyimler** olarak adlandırılan bu yeni özellik bir alan uzayı, sınıf veya yöntem kullanmaya gerek kalmadan doğrudan kod yazılmasına imkân sağlamaktadır. Bu yöntemde proje oluşturma adımları aşağıda ekran görüntüleriyle verilmiştir.

Kitabın bundan sonraki tüm konu ve örnekleri **Üst düzey deyimler** kullanılarak oluşturulan kodları içermektedir. Kitaptaki kodlar .NET 7.0 platformunda yazılmıştır. Dolayısıyla kodların sorunsuz çalışabilmesi bilgisayarda .NET 7.0 kurulu olmalı ve projeler aşağıda örneklediği gibi oluşturulmalıdır.

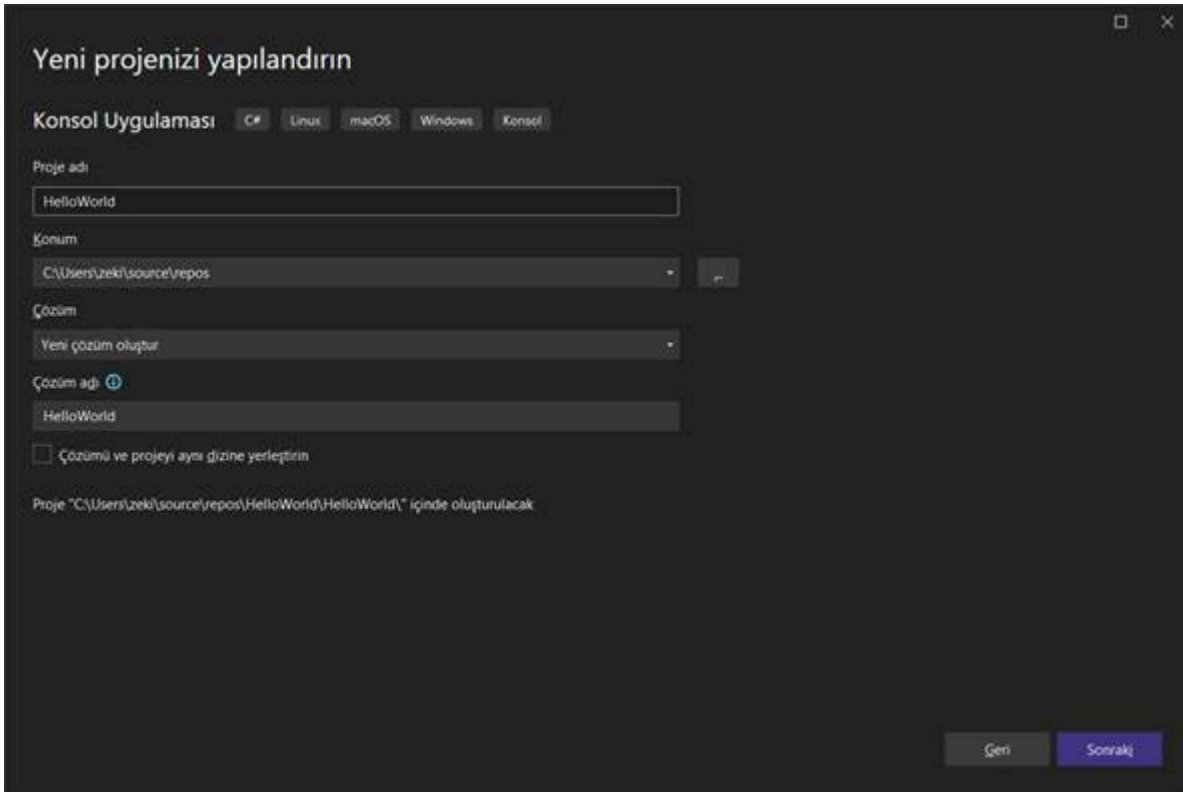
Öncelikle Visual Studio 2022 programının masaüstündeki veya başlat çubuğundaki simgesine çift tıklayarak programımızı çalıştıralım. Aşağıdaki ekranda **Yeni bir proje oluştur** seçeneğini tıklayalım.



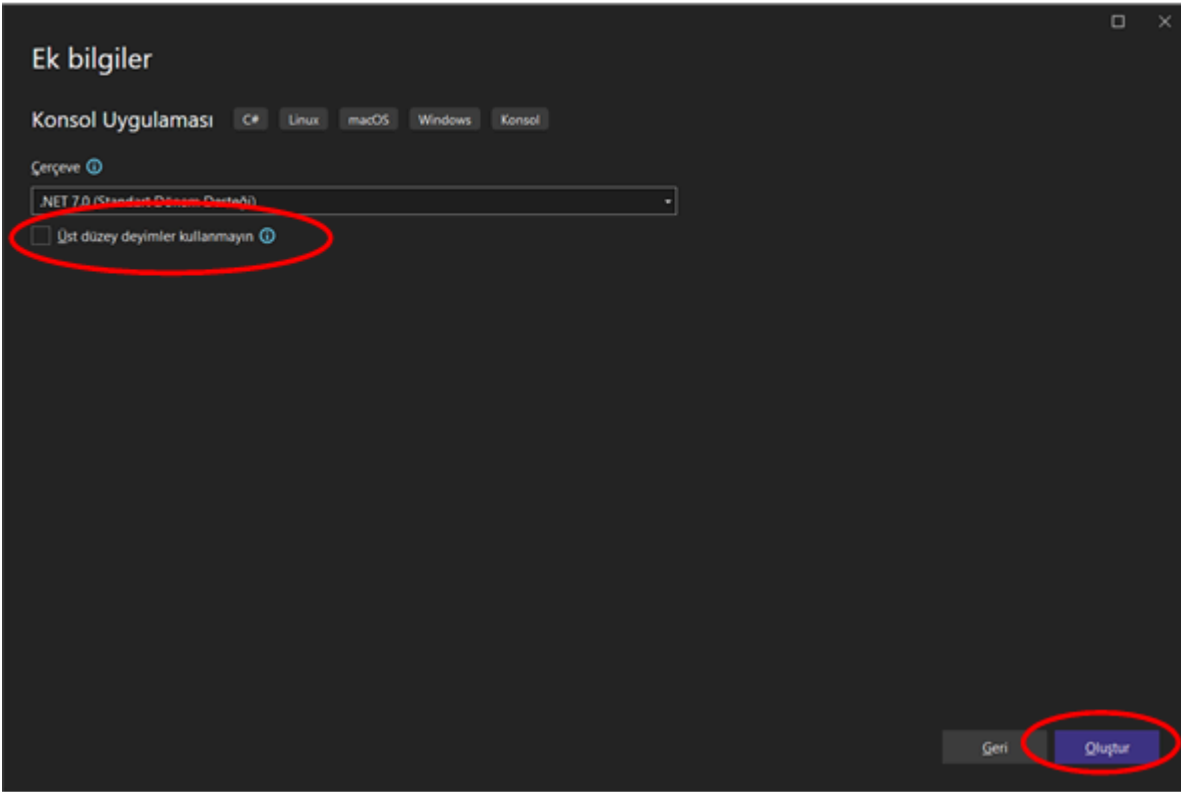
Gelen ekranda dillerden **C#** dilini seçip ve uygulama türlerinden **Konsol Uygulaması** seçeneğine tıklayıp **Sonraki** tuşuna basalım.



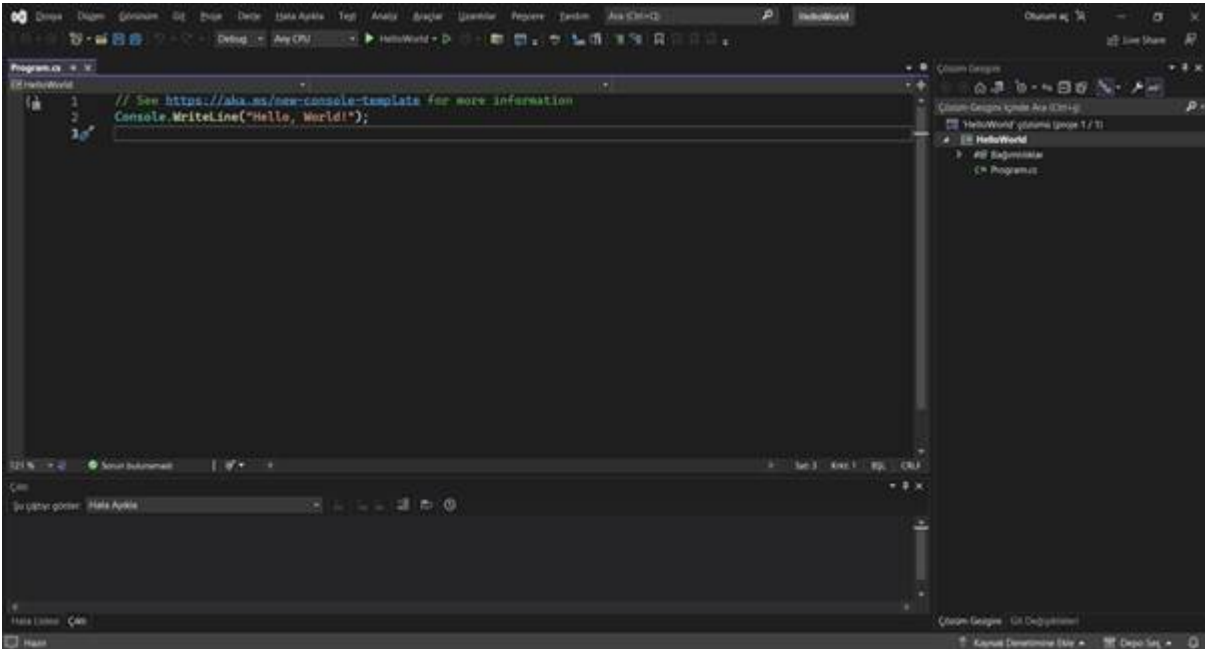
Sonraki ekranda projeye bir isim vermemiz istenecektir. Projeye HelloWorld ismini verelim.



Sonraki ekranda .NET versiyonu olarak 7.0 versiyonu seçilmeli ve **Üst düzey deyimleri kullanmayın** kutusunun işareti temizlenmeli yani seçim kaldırılmalıdır.



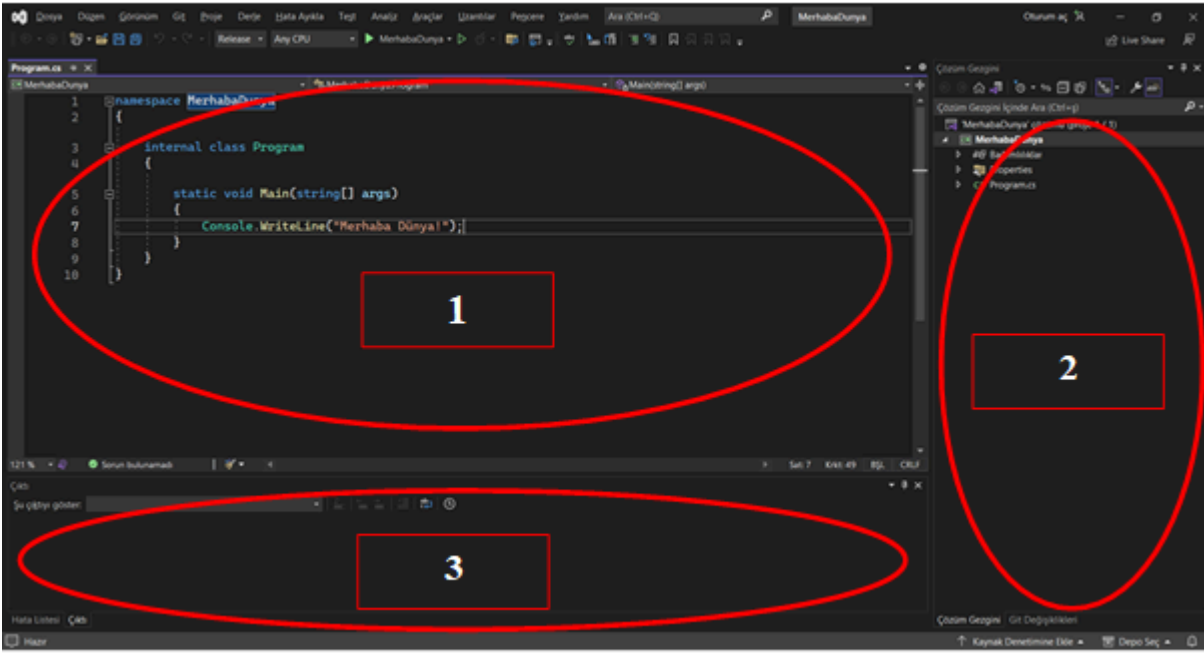
Oluştur butonuna basıldıktan sonra editör aşağıdaki ekranı gösterecektir. Görüldüğü üzere kaynak kod iki satırdan oluşmaktadır. İlk satır derleyici tarafından dikkate alınmayan yorum satırıdır. İkinci satır ekrana çıktı vermeyi sağlayan `Console.WriteLine()` komutunu içermektedir.



Bu yöntemle daha az kod satırıyla son derece sade ve önceki programla aynı işlevi gören program yazılmaktadır.

1.4. Visual Studio 2022 Kullanımı

İlk programımızın olduğu ekran üzerinden Visual Studio 2022 editörünü tanıyalım.



1 numaralı bölme kaynak kodu yazdığımız editör kısmıdır. Font, font büyüklüğü gibi ayarlar **Araçlar -> Seçenekler -> Ortam -> Genel -> Yazı Tipleri ve Renkler** menüsü aracılığıyla değiştirilebilir.

2 numaralı bölme çözüm, proje ve C# kaynak kod dosyalarının listelendiği **Çözüm Gezini** ekranıdır. Eğer gözükmüyorsa **Görünüm -> Çözüm Gezini** menüsüne tıklandığında belirtilen yerde görünür olacaktır. 2 numaralı kısımda da görüleceği üzere çözüm adı MerhabaDunya, çözümün altındaki proje adı da aynı şekilde MerhabaDunya olarak listelenmektedir. Çözümün bilgilerini içeren dosyanın uzantısı **.sln**, projenin bilgilerini içeren dosyanın uzantısı **.csproj**'dir. Projenin altında Program.cs dosyası yer almaktadır. Bu dosya C# kaynak kodunun kaydedildiği dosyasının ismidir. C# kaynak kod dosyasının uzantısı **.cs**'dir.

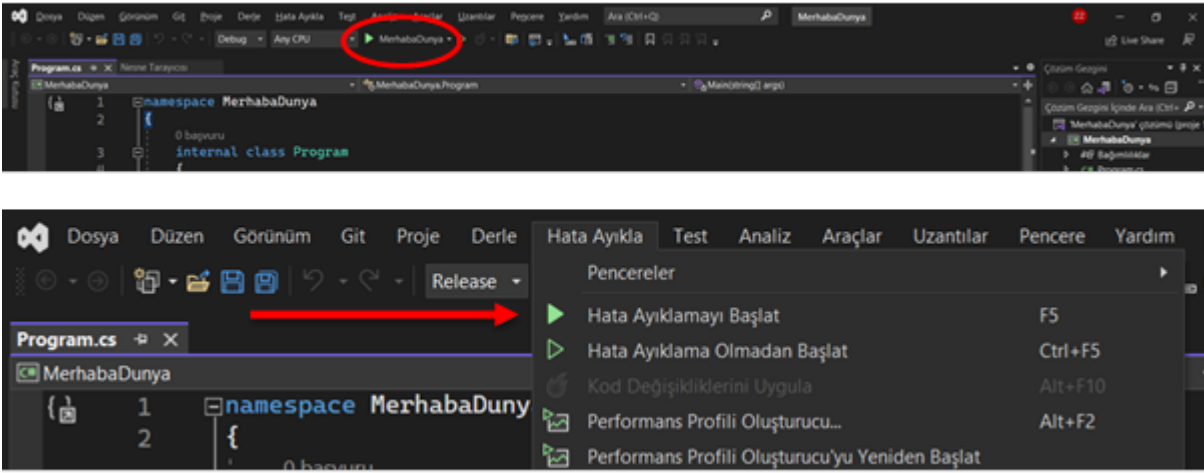
3 numaralı bölme programın kaynak kodundaki hataların listelendiği bölümdür. Eğer ilk ekranda görüntülenmiyorsa **Görünüm -> Hata Listesi** menüsüne tıklandığında belirtilen yerde görünür olacaktır. Bu bölümde kodun yazımı sırasındaki gramer hataları eş zamanlı olarak (bunlara derleme hatası denir) listelenir. Örneğin 7. satırın sonundaki noktalı virgöl silmez hemen bu kısımda ; bekleniyor hatası listelenecektir.

Editörün dili **Araçlar -> Seçenekler -> Uluslararası Ayarlar** menüsü takip edilerek istenilen dile değiştirilebilir. Değişikliğin etkili olması için editörün kapatılıp yeniden açılması gerekecektir.

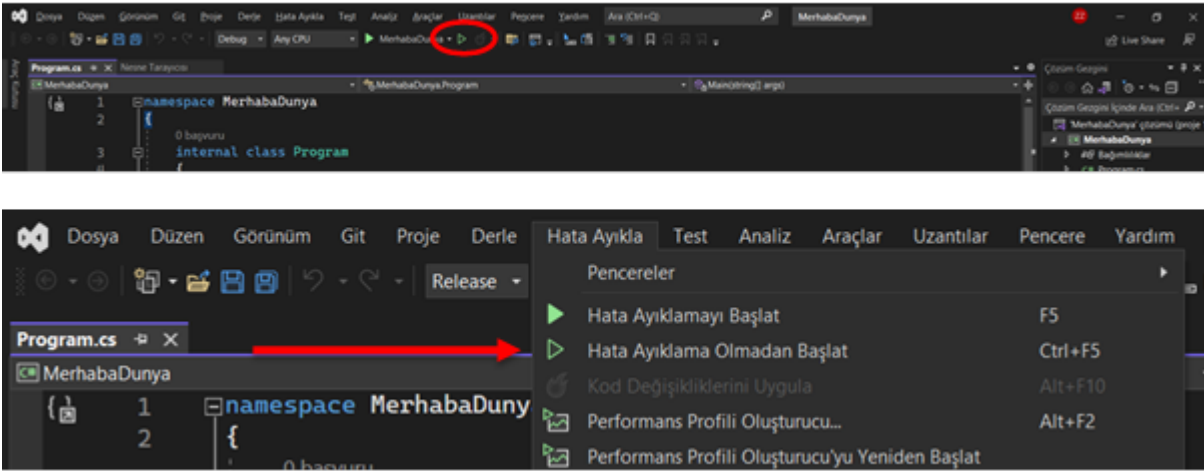
Visual Studio 2022 editörünün kodlama yaparken öneride bulunması için **Ctrl + Boşluk** tuşlarına basılabilir. Kaynak kodun girinti ve satırlarını düzenlemek yani kodu biçimlendirmek için **Ctrl + K + D** tuşlarına basılmalıdır.

1.4.1. Kaynak Kodu Çalıştırma

Kaynak kodda bir hata gözükmüyorsa programı kaynak koda derleyip çalıştırmak için **Hata Ayıkla -> Hata Ayıklamayı Başlat** menüsüne tıklamak, **yeşil renkli üçgen buton** veya **F5** tuşuna basmak gerekmektedir. Böylece kaynak kod derlenecek ve akabinde program konsol ekranında çalışacaktır. Hata ayıklama modu, programın olası hata veren satırlarına kesme noktası koyup programı adım adım çalıştırmak, kaynakları izlemek ve program çıktısını gözlemlemek için kullanılmaktadır.



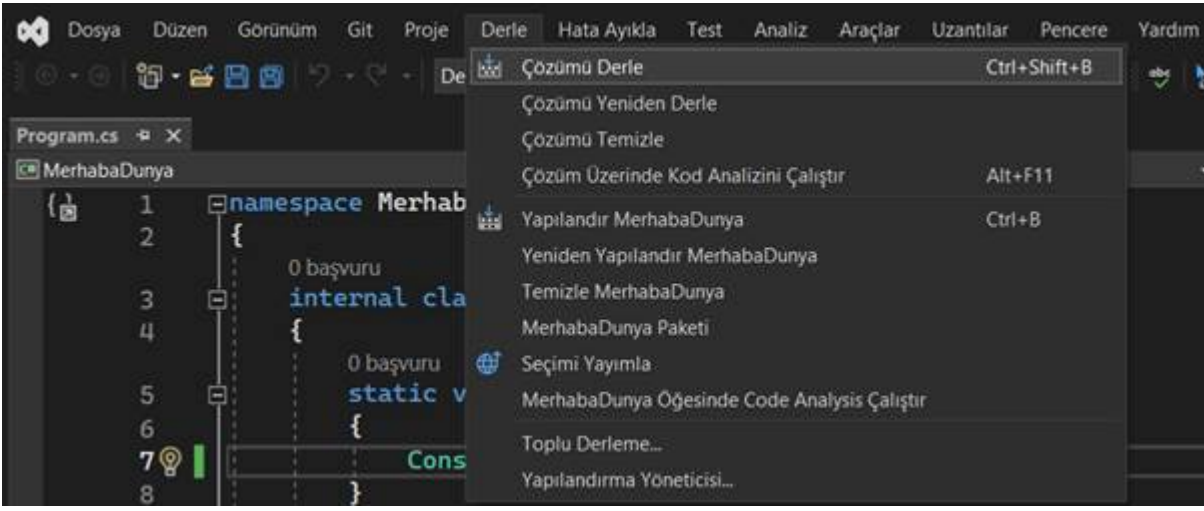
Programı çalıştırmanın bir diğer yöntemi de hata ayıklama modu olmadan programı doğrudan çalıştırmaktır. Bu modda program adım adım çalıştırılmaz, hata takibi yapılamaz, kaynaklar izlenemez. Bu yöntemde kaynak kodun programa çevrilip çalışması için **Hata Ayıkla -> Hata Ayıklama Olmadan Başlat** menüsü veya **Ctrl + F5** tuşları veya aşağıda gösterilen içi boş kenarları yeşil renkli butona tıklamak gerekmektedir.



1.4.2. Programı Derleme

Programın kaynak kodunun derlenmesi için üst taraftaki **Derle** menüsünün altında yer alan **Çözümü Derle** tıklanmalı veya **Ctrl + Shift + B** tuşlarına basılmalıdır. Bu sayede kaynak kod makine koduna dönüştürülür ve oluşturulan projenin altında bin\Release\net7.0 klasöründe programın çalıştırılabilir formu (Windows için .exe uzantılı dosyası) oluşturulur.

Örneğin MerhabaDunya isimli projenin derleme sonrası çalıştırılabilir dosyaları
 %userprofile%\source\repos\MerhabaDunya\MerhabaDunya\bin\Release\net7.0\ klasöründe oluşturulur.



Çözümü Yeniden Derle menüsü, daha önceki derleme sonrası oluşan tüm dosyaları siler ve projeyi sıfırdan derler. Proje ve kaynak kod dosyaları üzerinde yapılan değişiklikler dikkate alınarak taze bir derleme yapar.

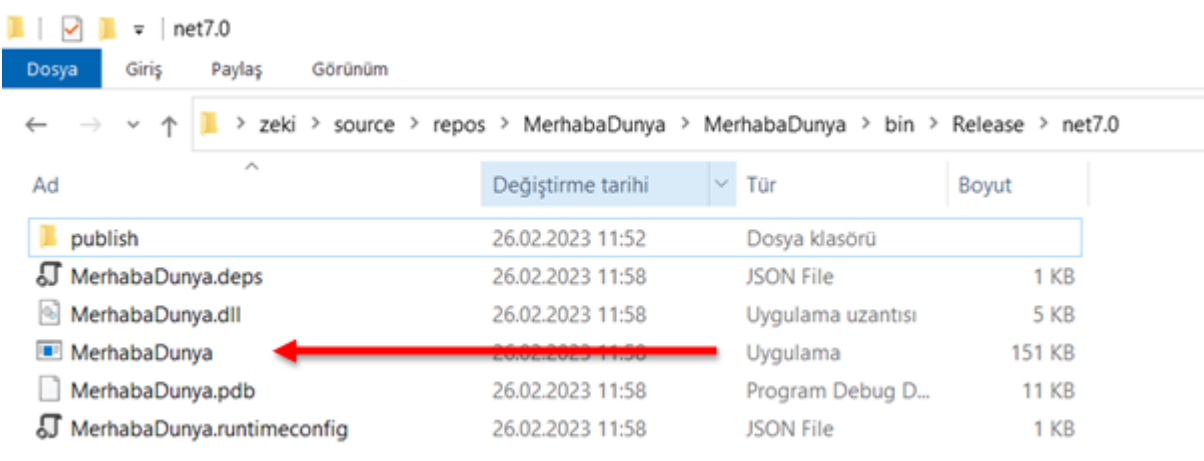
Çözümü Temizle menüsü, derleme sonrası oluşan tüm dosyaları siler.

Derleme, yazılım geliştirme süreçlerinden sadece biridir. Yazılımın ihtiyaç duyduğu kütüphanelerin yazılıma bağlanması (*linking*), derlenmesi, test, kurulum dosyalarının oluşturulması vb. diğer süreçleriyle hazır hale getirilmesi (*building*) için Derle menüsü altındaki **Yapılandır** seçeneği tıklanmalıdır.

Tüm süreçleri bitmiş ve artık son kullanıcının istifadesine sunulabilir hale gelen yazılımın yayımlanması için Derle menüsündeki **Seçimi Yayımla** seçeneği tıklanmalıdır. Bu işlem ile yazılımın kullanılabilir versiyonu bulut ortamında, konteynır platformlarda (Docker vb.) veya bilgisayarda bir klasörde konuşlandırılır.

1.4.3. Derlenen Programı Çalıştırma

Proje veya çözüm derlendiğinde proje\bin\Release klasörü altında seçilen .NET versiyonu ne ise o isimde klasör oluşturulur ve programın çalıştırılabilir formları o klasörde yer alır. Derlenen programı çalıştırmamanın en basit yöntemi bahsedilen klasördeki .exe uzantılı dosyaya çift tıklayıp normal bir program çalıştırır gibi çalıştırmaktır.



Diğer yöntem ise programı konsol uygulaması üzerinden çalıştırmaktır. Başlat tuşuna basılıp **cmd** veya **Komut İstemi** yazıldığında konsol uygulaması listelenecektir. Uygulamaya tıklanınca siyah konsol ekranı açılacaktır. Konsol ekranında programın çalıştırılabilir formunun yer aldığı klasöre gitmek için aşağıdaki komut yazıldığında konsolun listeleyeceği klasör değişecektir.

```
cd %userprofile%\source\repos\MerhabaDunya\MerhabaDunya\bin\Release\net7.0
```

Konsol istenilen klasöre geldiğinde aşağıdaki komut ile ilk yazdığımız program konsol ekranında çalışacaktır.

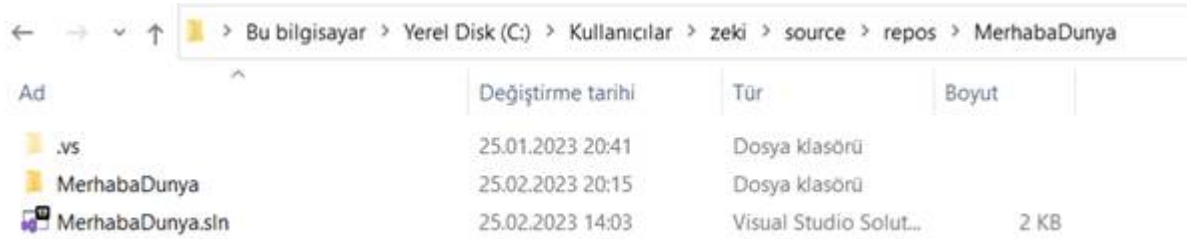
MerhabaDunya.exe

```
Komut İstemi
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\zeki>cd %userprofile%\source\repos\MerhabaDunya\MerhabaDunya\bin\Release\net7.0
C:\Users\zeki\source\repos\MerhabaDunya\MerhabaDunya\bin\Release\net7.0>MerhabaDunya.exe
Hello, World!
C:\Users\zeki\source\repos\MerhabaDunya\MerhabaDunya\bin\Release\net7.0>
```

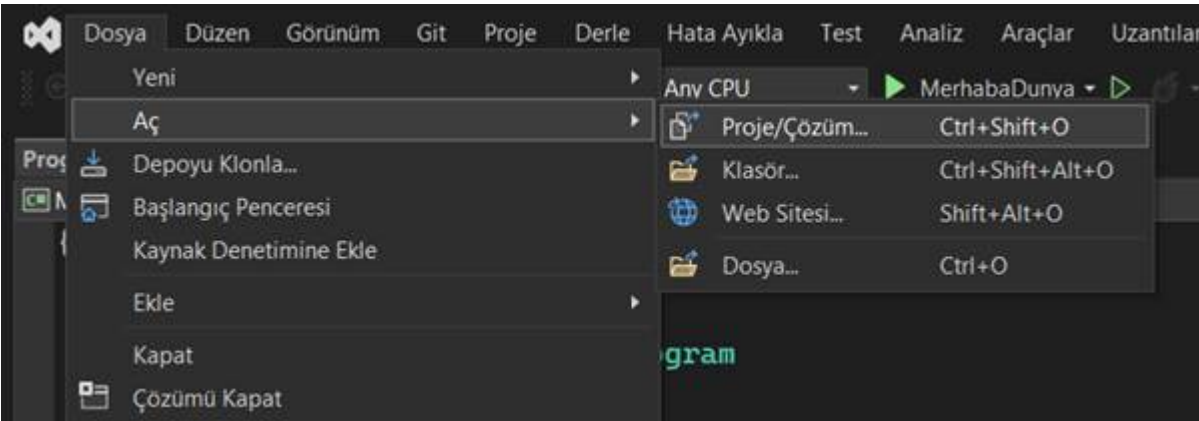
1.4.4. Bir Çözüm veya Proje Dosyasını Açma

Çözümün yer aldığı klasörün görüntüsü aşağıda verilmiştir. Başlat butonuna tıklanıp %userprofile%\source\repos yazıldığında Visual Studio 2022 editörü ile yazılmış olan çözümler listelenecektir. **MerhabaDunya.sln** dosyasına çift tıklandığında Visual Studio 2022, çözümü içindeki projelerle birlikte açacaktır.

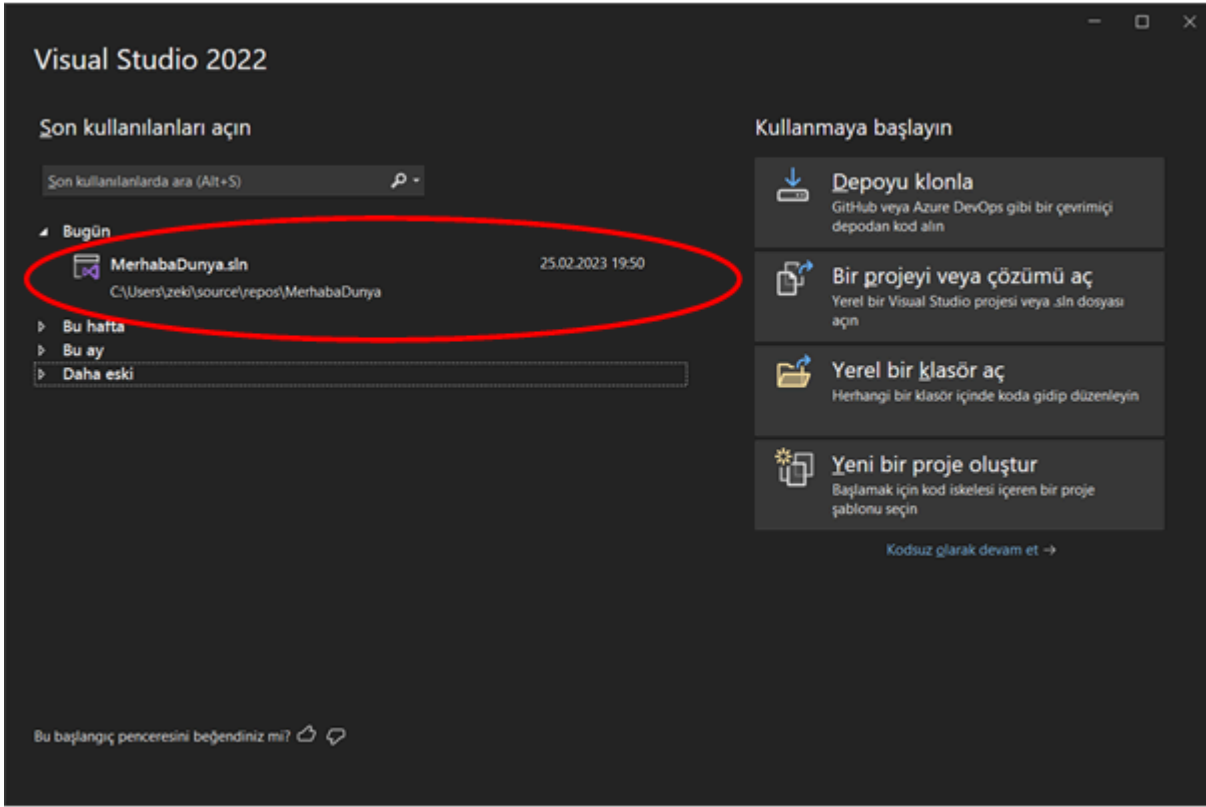


Ad	Değiştirme tarihi	Tür	Boyut
.vs	25.01.2023 20:41	Dosya klasörü	
MerhabaDunya	25.02.2023 20:15	Dosya klasörü	
MerhabaDunya.sln	25.02.2023 14:03	Visual Studio Solut...	2 KB

Bir diğer yöntem olarak **Dosya -> Aç -> Proje/Çözüm** menüsü de aynı işlevi yerine getirmektedir.



Bir çözümü açmanın son yöntemi de Visual Studio 2022 ilk açıldığı ekranda son yazılan çözümlerin listelendiği sol taraftan istenilen çözüme tıklamaktır.



1.5. Temel Girdi-Çıktı Komutları

Bu ders kapsamında programlamanın temelleri konsol üzerinden C# dile ile öğrenilecektir. Bu sebeple yalnızca konsol ekranı ile etkileşimde bulunulacaktır. Konsol ekranı klasik programlardan aşına olduğumuz herhangi bir pencere, form elemanı, görsel bileşen vb. içermez, yalnızca ekrana istenilen şeyleri çıktılar ve kullanıcıdan veri alır. Kısaca konsol ekranı sadece temel girdi-çıkı işlevlerini yerine getirir. Konsol uygulaması System alanı içindeki Console sınıfında tanımlıdır (Microsoft, 2023b).

1.5.1. Çıktı Komutları

Konsol üzerinden ekrana çıktı vermek için `Console.Write()` ve `Console.WriteLine()` metodları kullanılmaktadır (Microsoft, 2023c, 2023d).

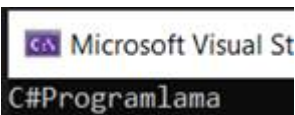
1.5.1.1. Console.Write()

`Console.Write()` metodu metni veya değişkeni imleç en son nerede kaldıysa oradan itibaren ekrana yazar. Bir metni ekrana çıktılamak için metin çift tırnak karakterleri arasında "" yazılmalıdır.

```
Console.Write("C#");
```

```
Console.Write("Programlama");
```

Bu iki satır ekrana bitişik halde `C#Programlama` yazdırır.



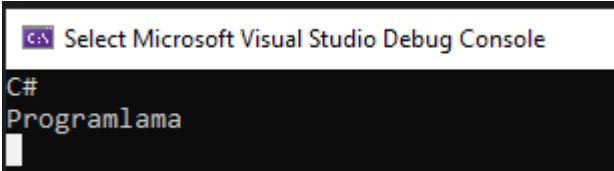
1.5.1.2. Console.WriteLine()

Console.WriteLine() metodu yazdırılacak metni veya değişkeni imleç en son nerede kaldıysa oradan itibaren ekrana yazar ve devamına yeni satır (*new line*) verir.

Aşağıdaki iki satır her bir metni ekrana çıktılıp devamına yeni bir satır verir. Bu durumda sonuncusu boş bir satır olmak üzere ekranda üç satır çıktılacaktır.

```
Console.WriteLine("C#");
```

```
Console.WriteLine("Programlama");
```



Aşağıda verilen kodun ilk satırı ekrana C# yazdırdıktan sonra yeni satıra geçer ve Programlama kelimesini yeni satırda yazdırır.

```
Console.WriteLine("C#");
```

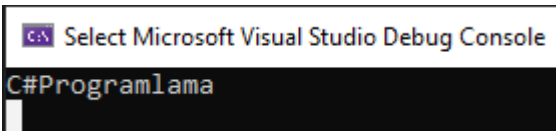
```
Console.Write("Programlama");
```



Aşağıda verilen kod ise ekrana C# ve devamına Programlama yazdırdıktan sonra yeni satır verir.

```
Console.Write("C#");
```

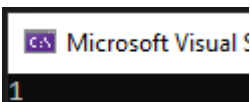
```
Console.WriteLine("Programlama");
```



Console.Write() ve Console.WriteLine() metodları metin haricinde değişkenleri de ekrana çıktılar. Değişkeni ekrana çıkıtmak için çift tırnak kullanılmaz.

```
byte sinif = 1;
```

```
Console.Write(sinif);
```



Yer Tutucu Yöntemiyle Çıktılama

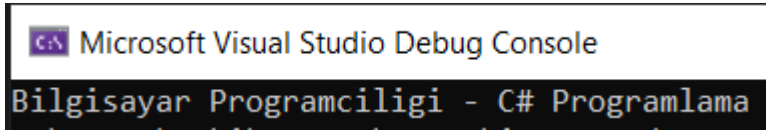
Konsol arayüzünde değişken(ler) ve metin aynı satırda **yer tutucu** (*string placeholder*) yöntemiyle ekrana çıktılabilir. Yer tutucu bir değişken veya bir değer yerini tutar. Çift tırnakların arasında süslü parantezler verilir ve içinde ekrana çıkılacak değişkenin numarası yazılır. Çift tırnak bittikten sonra virgül verilir ve çıkılacak değişken(ler) yazılır. Verilen ilk değişkenin numarası sıfırdır, sonrakiler bir artarak devam ederler. Temel sözdizimi aşağıda verilmiştir.

```
"{0}", degisken0/deger0
```

```
"{0} {1}", degisken0/deger0, degisken1/deger1
```

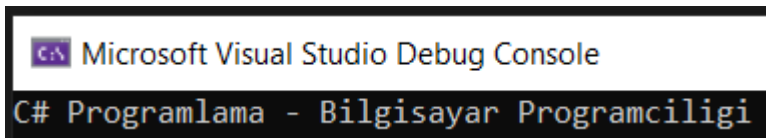
Aşağıdaki programda yer tutucu yöntemi kullanılarak iki değişken aynı satırda çıktılandırılmıştır. Verilen kodda 0 numaralı değişken bolum, 1 numaralı değişken ders değişkenidir.

```
string bolum = "Bilgisayar Programcılığı";  
  
string ders = "C# Programlama";  
  
Console.Write("{0} - {1} ", bolum, ders);
```



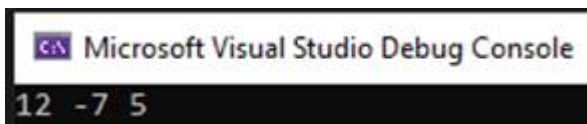
Süslü parantezler içinde yer tutucu numaralarının düzenli sırada verilmesi zorunlu değildir. Yukarıdaki programda yer tutucu numaraları 0 ve 1 olarak verilmiştir. Fakat kod içinde yer tutucu numaraları aşağıdaki satırda olduğu gibi 1 ve 0 şeklinde değiştirildiğinde program çıktısı da değişecektir. İlk değişkenin numarasının 0, sonrakinin 1 ve böylece artarak gittiği hatırlandığında 0 numaralı değişken bolum, 1 numaralı değişken ders değişkenidir. Bu durumda önce 1 numaralı değişken olan ders değişkeni ve daha sonra 0 numaralı değişken olan bolum değişkeni çıktılandırılacaktır. Bu durumda programın çıktısı aşağıdaki gibi olacaktır.

```
Console.Write("{1} - {0} ", bolum, ders);
```



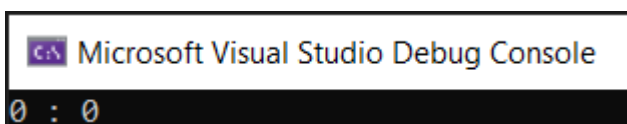
Konunun daha iyi anlaşılması için bir örnek daha verelim. Verilen kodda 0 numaralı değişken c değişkeni, 1 numaralı değişken a değişkeni ve 2 numaralı değişken b değişkenidir. Programdaki yer tutucular 2, 0 ve 1 sırasıyla verildiğinden önce 2 numaralı değişken yani b değişkeni, sonra 0 numaralı değişken yani c değişkeni ve sonra 1 numaralı değişken yani a değişkeni çıktılandırılacaktır.

```
int a = 5;  
  
int b = 12;  
  
int c = -7;  
  
Console.Write("{2} {0} {1}", c, a, b);
```



Yer tutucu yöntemiyle ekrana çıktı verilirken her değişkenin yer tutucu parantezleri içinde bildirilmesi zorunlu değildir. Aşağıdaki programda 0 numaralı değişken a ve 1 numaralı değişken b değişkenidir. Fakat yer tutucu süslü parantezleri içinde iki kere 1 numaralı değişken yani b değişkeni çıktılandırılmıştır.

```
int a = -1;  
  
int b = 0;  
  
Console.Write("{1} : {1}",a, b);
```



Yer tutucu yönteminde ilk değişkenin numarası sıfır olduğundan süslü parantezler içindeki en büyük numaranın bir fazlası kadar değişken çıktı komutunda verilmelidir. Aksi halde program hata verir. Örneğin aşağıdaki programda süslü parantezler içindeki en büyük numara 2 olarak verilmiştir. Bu durumda çıktı komutunun içinde 3 tane değişken verilmelidir. Fakat programda a ve b olmak üzere yalnızca iki adet değişken verilmiştir. Bu yüzden program çalıştırıldığında hata verecektir.

```
int a = -1;
```

```
int b = 0;
```

```
Console.Write("{2} : {0}",a, b);
```

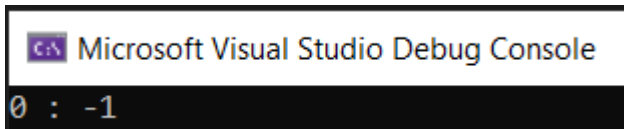
Yer tutucu yönteminde süslü parantezler içindeki en büyük numaranın bir fazlasından daha fazla değişken çıktı komutunda verildiğinde program hata vermez fakat fazlalık olan değişkenler ekrana çıktılanmaz. Örneğin aşağıdaki programda süslü parantezler içindeki en büyük numara 1 olarak verilmiştir. Bu durumda çıktı komutuna 2 adet değişken verilmelidir. Fakat programda a, b ve c olmak üzere üç adet değişken verilmiştir. Program normal şekilde çalışır fakat ziyade olarak verilen c değişkeni ekrana çıktılanmaz.

```
int a = -1;
```

```
int b = 0;
```

```
int c = 1;
```

```
Console.Write("{1} : {0}", a, b, c);
```



Dizge Enterpolasyonu Yöntemiyle Çıktılama

Sabit metin ve değişkenleri bir arada çıktılamanın bir diğer yöntemi ise değişkenleri metnin arasına doğrudan yerleştirmektir (*string interpolation*). Bu yöntemin sözdizimi de aşağıda verilmiştir. Çift tırnaktan önce \$ operatörüyle derleyiciye aynı satırda sabit metin ve değişken(ler)in bulunduğu bildirilir ve süslü parantezler içinde değişken(ler)in adı verilir.

```
$"Sabit metin {degisken}"
```

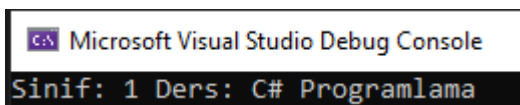
```
$"Sabit metin {degisken1} {degisken2}"
```

Yukarıda verilen örneği bu sefer araya koyma yöntemiyle ekrana çıktılayalım. Bu yöntem yer tutucu yöntemine göre daha pratiktir. Bu sayede hangi indiste hangi değişkenin verileceği karıştırılmamış olur.

```
string ders = "C# Programlama";
```

```
byte sinif = 1;
```

```
Console.WriteLine($"Sınıf: {sinif} Ders: {ders}");
```



Son olarak + operatörüyle ekrana çıktılanacak ifade ve değişkenler birleştirilebilir. + operatörü yan yana iki nümerik veri tipinde değişken olduğunda bunlara toplama işlemini uygulayacağından arzu edilen çıktı alınmayabilir. Bu sebeple + operatörüyle metin ve değişkenlerin birleştirilmesi pek tavsiye edilmez.

```
string ders = "C# Programlama";
```

```
byte sinif = 1;
```

```
Console.WriteLine("Sınıf: " + sinif + " Ders: " + ders);
```



```
Microsoft Visual Studio Debug Console  
Sınıf: 1 Ders: C# Programlama
```

1.5.2. Girdi Komutları

Kullanıcıdan veri almak için `Console.Read()`, `Console.ReadKey()` ve `Console.ReadLine()` olmak üzere 3 farklı girdi metodu bulunmaktadır (Microsoft, 2023e, 2023f, 2023g).

1.5.2.1. Console.Read()

Kullanıcıdan tek bir karakter almak için kullanılır. Girilen karakterin kendisi değil ASCII kodu `int` veri tipinde tutulur.

```
Console.WriteLine("Bir karakter giriniz:");
```

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

```
Console.Write("Girdiğiniz karakterin ASCII kodu: {0}", ch);
```



```
Microsoft Visual Studio Debug Console  
Bir karakter giriniz:  
A  
Girdiğiniz karakterin ASCII kodu: 65
```

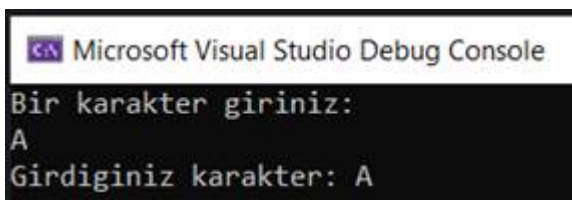
Programın çıktısından da görüldüğü üzere A karakteri girilmiş ve ekrana A karakterinin kendisi değil ASCII tablosundaki numarası olan 65 değeri çıktılanmıştır.

Girilen karakterin ASCII kodunu değil de karakterin kendisini ekrana çıktılamak için `int` veri tipinde tutulan kullanıcı girdisi `Convert.ToChar()` metoduyla `char` veri tipine çevrilmelidir.

```
Console.WriteLine("Bir karakter giriniz:");
```

```
char ch = Convert.ToChar(Console.Read());
```

```
Console.Write("Girdiğiniz karakter: {0}", ch);
```



```
Microsoft Visual Studio Debug Console  
Bir karakter giriniz:  
A  
Girdiğiniz karakter: A
```

1.5.2.2. Console.ReadKey()

Kullanıcıdan tek bir tuş verisi almak için kullanılır. Genellikle programın yapacağı işler bittikten sonra programı sonlandırmak için kullanıcıdan bir tuşa basması istenir. Bu amaçla `Console.ReadKey()` metodu kullanılır.

```
Console.WriteLine("Bir tuşa basınız:");
```

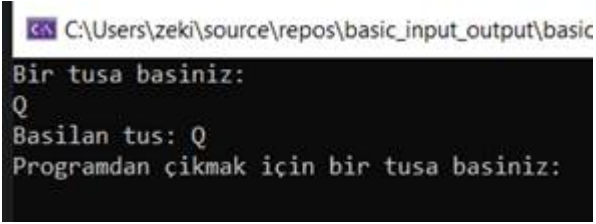
```
ConsoleKeyInfo tus = Console.ReadKey();
```

```
Console.WriteLine();
```

```
Console.WriteLine("Basılan tuş: " + tus.KeyChar);

Console.WriteLine("Programdan çıkmak için bir tuşa basınız:");

Console.ReadKey();
```



```
C:\Users\zeki\source\repos\basic_input_output\basic
Bir tusa basınız:
Q
Basılan tus: Q
Programdan çıkmak için bir tusa basınız:
```

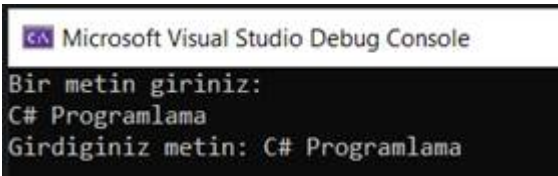
1.5.2.3. Console.ReadLine()

Kullanıcı Enter tuşuna basana kadar yazmış olduğu her şeyi alır ve string veri tipinde tutar. Kullanıcıdan çok karakterli metin ve sayı istemek için genellikle Console.ReadLine() metodu kullanılır.

```
Console.WriteLine("Bir metin giriniz:");

string metin = Console.ReadLine();

Console.WriteLine("Girdiğiniz metin: {0}", metin);
```



```
Microsoft Visual Studio Debug Console
Bir metin giriniz:
C# Programlama
Girdiğiniz metin: C# Programlama
```

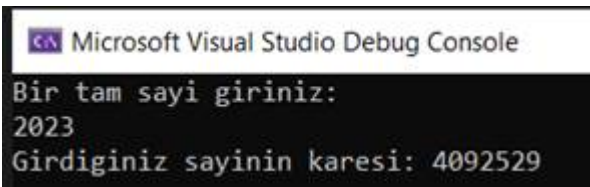
Console.ReadLine() metodu kullanıcı girdisini string veri tipinde tuttuğundan kullanıcıdan sayı girmesi istendiğinde girilen veri Convert metodu veya diğer veri tipi dönüşüm yöntemlerini kullanarak nümerik veri tiplerinden birine çevrilmelidir (Bkz: 3.2. Veri Tipi Dönüşümleri).

Aşağıda kullanıcıdan tam sayı girmesi istenmiş, Console.ReadLine() metoduyla alınan girdi Convert.ToInt32() metoduyla tam sayı (int) veri tipine çevrilmiş ve ekrana sayının karesi çıktıktanmıştır.

```
Console.WriteLine("Bir tam sayı giriniz:");

int sayi = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Girdiğiniz sayının karesi: {0}", sayi * sayi);
```



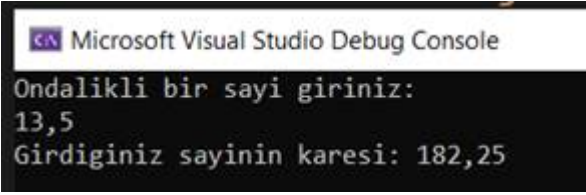
```
Microsoft Visual Studio Debug Console
Bir tam sayı giriniz:
2023
Girdiğiniz sayının karesi: 4092529
```

Aşağıda kullanıcıdan ondalıklı sayı girmesi istenmiş, Console.ReadLine() metoduyla alınan girdi Convert.ToDouble() metoduyla ondalıklı sayı (double) veri tipine çevrilmiş ve ekrana sayının karesi çıktıktanmıştır.

```
Console.WriteLine("Ondalıklı bir sayı giriniz:");

double sayi = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Girdiğiniz sayının karesi: {0}", sayi * sayi);
```



```
Microsoft Visual Studio Debug Console
Ondalikli bir sayi giriniz:
13,5
Girdiginiz sayinin karesi: 182,25
```

Bölüm Özeti

Visual Studio 2022 yazılımı, C# dilinin geliştiricisi Microsoft tarafından yayımlanan kodlama editörüdür. Derleyici ve koddaki hataları ayıklama işlevlerini sağlayan çeşitli araçlar bütünleşik halde editörle birlikte kurulmaktadır.

Çözüm en kapsamlı yapı olup içinde bir veya daha fazla proje barındırabilir. C# dosyaları .cs uzantısıyla kaydedilir.

Konsol ekranı, ilkel ve hiçbir görsel bileşen barındırmayan basit bir arayüzdür. Konsol ekranı aracılığıyla kullanıcıdan girdi alınmakta ve ekrana istenilen mesaj veya değişkenlerin değerleri çıktılanmaktadır.

Console.Read() metodu kullanıcıdan tek bir karakter alır. Girilen karakterin kendisini değil ASCII kodunu int veri tipinde saklar. Console.ReadKey() metodu ise kullanıcıdan tuş verisi alır. Console.ReadLine() metodu ise kullanıcının konsol satırında yazdığı her şeyi almak için kullanılmaktadır. Kullanıcıdan veri almak için genellikle Console.ReadLine() metodu tercih edilmektedir. Console.ReadLine() metodu kullanıcı verisini string veri tipinde sakladığı için girdinin nümerik veri tiplerine dönüştürülmesi için Convert sınıfı veya diğer veri tipi dönüşüm yöntemleri kullanılır.

Konsol ekranına bir şey yazdırılmasını sağlayan Console.Write() metodu imlecin en son kaldığı yerden itibaren istenilen ifadeyi ekrana çıktılar. Console.WriteLine() metodu ise aynı işi yapmakla birlikte ifadenin sonuna yeni satır ekler.

Kaynakça

Aktaş, V. (2017). *HER YÖNÜYLE C# 5.0* (5. bs.). İstanbul: Kodlab.

Microsoft. (2023a). Install Visual Studio. <https://learn.microsoft.com/en-us/visualstudio/install/install-visual-studio> adresinden erişildi.

Microsoft. (2023b). Console Class (System). <https://learn.microsoft.com/en-us/dotnet/api/system.console?view=net-7.0> adresinden erişildi.

Microsoft. (2023c). Console.Write Method (System). <https://learn.microsoft.com/en-us/dotnet/api/system.console.write?view=net-7.0> adresinden erişildi.

Microsoft. (2023d). Console.WriteLine Method (System). <https://learn.microsoft.com/en-us/dotnet/api/system.console.writeline?view=net-7.0> adresinden erişildi.

Microsoft. (2023e). Console.Read Method (System). <https://learn.microsoft.com/en-us/dotnet/api/system.console.read?view=net-7.0> adresinden erişildi.

Microsoft. (2023f). Console.ReadKey Method (System). <https://learn.microsoft.com/en-us/dotnet/api/system.console.readkey?view=net-7.0> adresinden erişildi.

Microsoft. (2023g). Console.ReadLine Method (System). <https://learn.microsoft.com/en-us/dotnet/api/system.console.readline?view=net-7.0> adresinden erişildi.