



**MANİSA**  
**CELAL BAYAR**  
**ÜNİVERSİTESİ**  
**KIRKAĞAÇ MESLEK YÜKSEKOKULU**

**BİLGİSAYAR TEKNOLOJİLERİ BÖLÜMÜ**  
**BİLGİSAYAR PROGRAMCILIĞI PROGRAMI**  
**PROGRAMLAMA TEMELLERİ DERSİ NOTLARI**

## YARARLANILABİLECEK KAYNAKLAR ve ARAÇLAR

- Algoritma ve Programlamaya Giriş. Ebubekir YAŞAR. Ekin Yayınevi.
- Algoritma ve C# Programlama. Erhan ARI. Seçkin Yayınevi.
- C# Programlama Dili ve Yazılım Tasarımı Cilt 1: Programlama İlkeleri. Ahmet KAYMAZ. Papatya Yayınları.
- C# Programlama Dili Yazılım Tasarımı Cilt 2: İleri düzey Programlama. Ahmet KAYMAZ. Papatya Yayınları.
- Algoritmalar. Süleyman UZUNKÖPRÜ. Kodlab Yayınları.
- Ders notları
- Konu ile ilgili internet kaynakları ☺

### Elektronik Kaynaklar:

<https://www.btkakademi.gov.tr/>

<https://bilgeis.net/>

[Freecomputerbooks.com](http://Freecomputerbooks.com)

<http://www.cizgi-tagem.org/>

<https://github.com/>

[code.org](http://code.org)

[codemonkey](http://codemonkey)

[gelecegiyazanlar.turkcell.com.tr](http://gelecegiyazanlar.turkcell.com.tr)

[Codecademy.com](http://Codecademy.com)

[www.eba.gov.tr](http://www.eba.gov.tr)

<https://scratch.mit.edu>

[www.edx.org](http://www.edx.org)

<https://www.edx.org/course/subject/computer-science>

[www.acikakademi.com](http://www.acikakademi.com)

<http://www.khanacademy.org.tr/>

<http://coderdojoturkiye.com/>

[https://docs.google.com/document/d/1b1Q4nq\\_11kA1JCyMWI0LWfNGPmNZsXUpXDbKa5ZqQwM/edit?usp=sharing](https://docs.google.com/document/d/1b1Q4nq_11kA1JCyMWI0LWfNGPmNZsXUpXDbKa5ZqQwM/edit?usp=sharing)

[www.udemy.com](http://www.udemy.com)

[www.scodeapp.com](http://www.scodeapp.com)

[www.istihza.com](http://www.istihza.com)

<http://php.net/manual/tr/getting-started.php>

<http://www.ruby-lang.org/tr/community/>

[www.muhammedmatar.com](http://www.muhammedmatar.com)

Codea? <http://twolivesleft.com/Codea/>

<https://codecombat.com/>

<https://www.codeschool.com/>

<https://itunes.apple.com/us/app/daisy-the-dinosaur/id490514278>

<http://www.gethopscotch.com/>

<http://movetheturtle.com/>

<http://railsforzombies.org/>

<https://www.udemy.com/programming-for-kids-how-to-make-coding-fun/>

**ARAÇLAR**

- **C#**
  - Derste herhangi bir MS C# Sürümü kullanılabilir. Bunlar MS Visual Studio’nun herhangi bir sürümünün içinde yer alan C# olabileceği gibi, MS ürünü olmayan herhangi bir C# IDE’si de olabilir.
  - Bir Microsoft hesabı ve üniversite e-posta hesabınız (öğrencinumaranız@ogr.cbu.edu.tr) ile imagine.microsoft.com’dan istediğiniz ürünü ücretsiz indirebilirsiniz.
  - Veya icsharpcode.net adresinden **SharpDevelop(#develop)** bağlantısından ücretsiz indirebilirsiniz.
  - **Flow Chart Visual Programming Language**→ basit algoritma akış diyagramı çizmek ve test etmek için.
- **Code Visual toFlowChart**→programlama dili kodlarını akış diyagramına dönüştürmek için.
- **Code Visual toFlowChart**→programlama dili kodlarını akış diyagramına dönüştürmek için.
- Linux ve MacOS altında MS Visual Studio kullanmak için Microsoft’un kendi desteği olan “Visual StudioCode” indirilebilir.
- Microsoft’un öğrenciler için imagine.microsoft.com portalından «Mac için Visual Studio» ücretsiz indirilebilir.

**Çevrim içi C# derleyiciler:**

- [https://www.onlinegdb.com/online\\_csharp\\_compiler](https://www.onlinegdb.com/online_csharp_compiler)
- [https://www.w3schools.com/cs/cs\\_compiler.php](https://www.w3schools.com/cs/cs_compiler.php)
- <https://dotnetfiddle.net/>
- <https://rextester.com/>
- <https://www.jdoodle.com/compile-c-sharp-online/>
- <https://onecompiler.com/csharp>

**Mobil C# derleyiciler:**

- C# Shell (C# Offline Compiler)
- C# Programming Compiler
- C# Programming Compiler Run C# (C Sharp) Scripts Ketan Appa (App Store)
- C# Compiler – OnePercent (App Store)

**Çevrim içi Akış Diyagramı:**

- <https://app.diagrams.net/> (draw.io)
- <https://www.lucidchart.com/pages/>
- <https://www.diagrameditor.com/>

**Ödev:**

- <https://github.com/> nedir? Ne amaçla kullanılır? Yaraları nelerdir? Araştırınız.
- Herkesin bir github.com hesabı olsun.
- Herkesin ad.soyad@epostasunucu.com şeklinde bir e-posta hesabı olsun.

## PROGRAMLAMA KAVRAMLARI

### Programlama

Türk Dil Kurumu, programı “belli bir çalışmanın amacını, bölümlerini, yöntemini ve süresini gösteren plan.” olarak tanımlamaktadır. Programlama ise program yapmak olarak düşünülebilir.

Bir bilgisayar programından söz edildiğinde ise bir problemi çözen yazılım anlaşılır. Programlama ise bu yazılımın üretilmesi işidir.

Bilgisayar programlama; sistem analizi ve tarasımı ile başlayan, ilgili problemin muhtemel çözümlerinin ortaya konması, çözüm için bir algoritma geliştirilmesi, uygun sistem ve uygun bir bilgisayar programlama dilinin seçilerek çözümün kodlanması, ortaya çıkan yazılımın test edilmesi, alınan geri dönüşlere göre iyileştirilmesi, bakımının yapılması ve güncellenmesi ile devam eden bir süreçtir.

Bilgisayar programlama, donanıma nasıl davranacağını bildirme, ona yön verme, aritmetik ve mantıksal işlemlerin ne olacağını ve nasıl işleyeceğini bildirme işidir.

Çoğunlukla çok iyi tanımlanmış bir sorunun çözümüne dair adımlar ile çözümün oluşturulup bunun bir programlama dili ile bilgisayar ortamına aktarılması programlama diye adlandırılabilir.

### Problem

Teoremler veya kurallar yardımıyla çözülmesi istenen soru, mesele. (TDK)

Bir işlemin, otomasyonun ya da bilimsel hesaplamanın bilgisayarla çözülmesi fikrinin ortaya çıkmasına problem denir. Bu tip fikirlerde insanların bu sorunları beyinle çözmeleri ya imkânsızdır ya da çok zor ve zaman alıcıdır. Bu tip bir sorunu bilgisayarla çözebilme fikrinin ortaya çıkması bir bilgisayar probleminin ortaya çıkmasına neden olmuştur. Bazen de bir işletme veya yönetimin otomasyonunu sağlamak amacı ile bu tip problemler tanımlanır.

### Problem Çözümü

Problemi Çözebilmek için öncelikle sorunun çok net olarak programcı tarafından anlaşılmış olması gerekir. Tüm ihtiyaçlar ve istekler belirlenmelidir. Gerekliyse bu işlem için birebir görüşmeler planlanmalı ve bu görüşmeler gerçekleştirilmelidir. Problemin Çözümüne ilişkin zihinsel alıştırmalar yapılır. Bu alıştırmaların Bilgisayar çözümüne yakın olması hedeflenmelidir. Bir sorunun tabii ki birden fazla çözümü olabilir. Bu durumda bilgisayar ile en uygun çözüm seçilmelidir. Çünkü bazen pratik çözümler bilgisayarlar için uygun olmayabilir. Oluşturulan Çözüm Algoritma dediğimiz adımlarla ifade edilmelidir. Bu algoritmanın daha anlaşılabilir olması için Akış Çizgesi oluşturulmalıdır. Uygun bir programlama dili seçilmeli ve oluşturulan algoritma ve akış çizgesi bu programlama dili aracılığı ile bilgisayar ortamına aktarılmalıdır. Oluşturulan program bir takım verilerle ve mümkünse gerçek ortamında test edilir. Oluşabilecek sorunlar ilgili kısımlar tekrar gözden geçirilerek düzeltilir. Bu adımlar defalarca gerçekleştirilmek zorunda kalınabilir.

Problem çözümü kısmında anlatılan adımlar uygulandıktan sonra ortaya çıkan ve sorunumuzu bilgisayar ortamında çözen ürüne program denir. Bazı durumlarda bu ürüne yazılım denebilir.

Problem çözümünde anlatılan adımların tümüne birden programlama denilebilir. Ancak gerçekte ilk paragrafta anlatılan kısım çoğunlukla sistem analizi veya sistem çözümleme olarak anlatılır. Diğer adımlar Programlama diye tanımlanabilir. Ancak son paragrafta anlatılan adıma kısaca test aşaması da denir.

### Algoritma

Algoritma sözcüğü Ebu Abdullah Muhammed İbn Musa el Harezmi adındaki dünyaca ünlü matematikçi âlimden kaynaklanır. Bu âlim 9. yüzyılda cebir alanındaki algoritmik çalışmalarını kitaba dökerek matematiğe çok büyük bir katkı sağlamıştır. "Hisab el-cebir ve el-mukabala (حساب الجبر والمقابلة)" kitabı dünyanın ilk cebir kitabı ve aynı zamanda ilk algoritma koleksiyonunu oluşturur. Latince çevirisi Avrupa'da çok ilgi görür. Analitik geometrinin kurucusu, ikinci dereceden bir ve iki bilinmeyenli denklemlerin çözümünü hem cebirsel hem de geometriksel olarak gösteren, matematiksel işlemlerde sıfır sayısının nasıl kullanılacağını ilk defa açıklayan âlimin ismini telaffuz edemeyen Avrupalılar "algorizm" sözcüğünü "Arap sayıları kullanarak aritmetik problemler çözme kuralları" manasında kullanırlar. Bu sözcük daha sonra "algoritma"ya dönüşür ve genel kapsamda kullanılır. (Literatür)

### Algoritma Nedir?

Bir sorunu çözebilmek için gerekli olan sıralı mantıksal adımların tümüne denir. Doğal dille yazılabileceği için fazlaca formaldeğildir.

Bir algoritma için aşağıdaki ifadelerin mutlaka doğrulanması gereklidir.

- Her adım son derece belirleyici olmalıdır. Hiç bir şey şansa bağlı olmamalıdır.

- Belirli bir sayıda adım sonunda algoritma sonlanmalıdır.
- Algoritmalar karşılaşılabilecek tüm ihtimalleri ele alabilecek kadar genel olmalıdır. (gokhandokuyucu.com)

Her algoritma aşağıdaki kriterleri sağlamalıdır. (Mehmet AKTAŞ – Mustafa AKSU)

- 1. Girdi:** Sıfır veya daha fazla değer dışarıdan verilmeli.
  - 2. Çıktı:** En azından bir değer üretilmeli.
  - 3. Açıklık:** Her işlem (komut) açık olmalı ve farklı anlamlar içermemeli.
  - 4. Sonluluk:** Her türlü olasılık için algoritma sonlu adımda bitmeli.
  - 5. Etkinlik:** Her komut kişinin kalem ve kağıt ile yürütebileceği kadar basit olmalıdır.
- Not:** Bir program için 4. özellik geçerli değil. İşletim sistemleri gibi programlar sonsuza dek çalışırlar.

Programlamanın temelinde çalışma akışını, izlenecek yolları belirleyen algoritmalar vardır. Bir iş yapılmaya başlanmadan önce nasıl planlanıyorsa, kodlamaya geçilmeden önce de bir çalışma planı belirlenmelidir. Yazılımlar, bu planda yazılan kodları belli bir sıra ile okur ve işler. Dolayısıyla algoritma yapısını çok iyi kurmak gerekir. Kurulan algoritmalar akış diyagramları ile görsel zenginlik kazanırlar.

Algoritma, bir işin hangi etaplardan geçilerek yapılacağını gösteren çalışma planıdır. Algoritma bir programlama dili değildir. Programlama dillerine yol gösteren bir yöntem dizisidir. Her dilde algoritma yazılıp uygulanabilir. Örneğin bir cep telefonunun el kitapçığında yazan, rehber kaydı girmek için izlenecek yollar, o işin algoritmasıdır.

Algoritma yazarken, uygulamanın çalışması için kullanılan kaynakların, yapılması gereken kontrollerin veya işlemlerin açıkça ifade edilmesi gerekir. Ayrıca iyi bir algoritmanın, tüm ihtimalleri kontrol edip istenmeyen durumlarda ne yapılması gerektiğini belirtmesi gerekir.

Örneğin, bir e-ticaret uygulamasında ürün satış algoritması çıkarılır. Satın alınacak ürün seçildikten sonra, kullanıcıdan adet miktarı bilgisi alınır. Uygulama yazılırken, bu değerin int veri tipinde olacağına karar verildiği düşünülürse; kullanıcının girdiği adet miktarı bu değişkene atanmadan önce kontrol edilmelidir. Eğer int veri tipinin tutamayacağı bir değer girilmişse, çalışma anında uygulamanın beklenmedik şekilde durduğu ya da istenmeyen sonuçların üretildiği gözlemlenir. Ayrıca sistemin verdiği hata, kullanıcının anlamayacağı bir mesaj içereceği için, uygulamanın imajını da kötü yönde etkiler.

### **Veri girişi**

Çalışma zamanında çoğu zaman, işleyişin tamamlanması için dışarıdan bir bilgi girilmesi gerekir. Algoritmanın çalışması için ihtiyaç duyduğu veriler, işlemi başlatan kişiden veya belirtilen bir kaynaktan alınabilir. Bu bilgiler sağlanmadan işlem devam etmez.

### **Kararlar**

Karar ve kontrol yapıları algoritmanın akışını yönlendiren en önemli kavramlardır. Girilen veya işlem sonucunda elde edilen veriler, işlemin amacına göre kontrol edilir ve sonuca göre algoritma akışı istenilen yere yönlendirilir.

### **İşlemler**

Algoritmanın akışı boyunca veriler üzerinde değişiklikler, yeni değer atamaları gibi işlemlere ihtiyaç duyulur. Algoritmalar kurulurken, yapılan işlemlerin yalın halde, tek tek yazılması okunabilirliği artırır.

Algoritmalar adım sırası ile çalışır ve karar yapıları sonucunda farklı bir yere yönlendirilmediği müddetçe, bir sonraki adım ile işlemeye devam eder.

**Örnek:** Telefon kulübesinden telefon açmak için örnek bir algoritma

1. Telefon kulübesine git
2. Telefon kartı al
3. Telefon sırasında kaç kişi olduğuna bak
4. Kişi sayısı sıfırdan fazlaysa 3 e dön
5. Kapı kapalıysa kapıyı aç
6. İçeri gir, kapıyı kapat
7. Telefon kartını telefona yerleştir
8. Ahizeyi kaldır
9. Numarayı çevir
10. Konuşmanın bitip bitmediğine bak
11. Konuşma bittiyse kartı al, bitmediyse 10 a dön
12. Bir daha konuşma yapılacaksa 7'ye dön
13. Kapıyı aç, dışarı çık

Bu algoritmanın işlenmesi için, her ihtimal gözden geçirilerek, algoritma akışı gerekli yerlere yönlendirilir. Örneğin kapının kapalı olması durumunda kapıyı açmak için gerekli komutlar verilmelidir. Bu algoritmanın ihtiyaç duyduğu veriler, ya kullanıcı tarafından verilir ya da işlem başlamadan önce belirlidir. Sıradaki kişi sayısı, telefon kartı gibi veriler kullanıcı tarafından sağlanmış; çevrilecek numara, algoritma başlamadan önce belirlenmiştir.

(Ayrılmaz, 2008, sf:89-90)

#### **Akış Çizelgesi Nedir?**

Bir algoritmanın daha görsel gösterimidir. Çizgiler, Dörtgen, daire vb. geometrik şekillerle algoritmanın gösterilmesini sağlar. Doğal dille yazılmadığı için daha formal olduğu düşünülebilir.

#### **Programlama Dili Nedir?**

Bir Problemin Algoritmik çözümünün Bilgisayara anlatılmasını sağlayan, son derece sıkı-sıkıya kuralları bulunan kurallar dizisidir. (gokhandokuyucu.com) Yeni yazılımlar üretmeye yarayan yazılımlara programlama dili denir.

##### **1.1. Algoritma Yazım Aşamaları (yaşar, 2011, sf:5 - 8)**

###### **1.1.1. Problemi Tanımlama**

Çözüme kavuşturulacak problemin eksiksiz ortaya konması gerekir. Problem ne kadar tümüyle ortaya konursa program da o kadar tam olacaktır. Problemdaki eksiklikler programı doğrudan etkileyerek eksik olmasına neden olacaktır.

###### **1.1.2. Problemi Geliştirme**

Özellikle otomasyon programlarında, problemin ortaya konması farklı branş uzmanı kişilerle çalışmakla mümkündür. Problem tanımlanırken birkaç kişi ile sınırlı kalmamak önemlidir. Bu bize yazılacak programda daha **genel** bir yapının doğmasını sağlayacaktır. Problem ne kadar genel ve geçer olursa program da o kadar genel ve geçer olur. Böyle bir program daha çok kişi tarafından benimsenecektir.

###### **1.1.3. Çözümün Sisteme Uyumluluğunu Tespit Etme**

Problem ve istekler doğrultusunda meydana gelecek çözüm, programın gücünü gösterecektir. Sistemin nasıl çalıştığını bilerek, ortadaki probleme getirilen çözümleri belirli bir sıraya sokmak gerekecektir. Programı değerli kılan ihtiyaçlar, istekler ve bunlara getirilen çözümdür. Algoritmayı kurarken kesinlikle, problemi yaşayan işin uzmanı kişilerle sıkı bir diyalog içinde olunmalıdır. **Kendi başınıza getireceğiniz çözümler elbette kendinize göre olacaktır ve sadece sizin işinize yarayacaktır!**



**1.1.4. Çözümü Kâğıt Üzerinde Gösterme**

Probleme getirilen çözüm birden fazla olabilir. Bu çözümlerden en uygunu, en basiti ve en etkili olanı seçilmelidir. Bazen problem tek veya iki – üç aşamada çözülmeyebilir. Bunlar, alt gruplara ayrılarak bütünlüğü korunacak şekilde çözüme kavuşturulmalıdır. Bu gruplar arası bütünlük asla bozulmamalıdır. Probleme getirilen çözümler çalışma kâğıdına dökülerek problemin ve getirilen çözümün tamamına hâkim olmaya çalışılmalıdır. Problemin bir kısmına çözüm getirirken, getirilen çözümler bazen farklı başka problemleri doğurabilir.

**1.1.5. Çözümü Deneme**

Üretilen ve kâğıda aktarılan çözüm farklı senaryolar ile ve hatta mümkünse farklı kullanıcı grupları ile denenmeli ve oluşabilecek tüm olasılıklar araştırılmalıdır.

**1.1.6. Çözümü Geliştirme**

Başlangıçta göz ardı edilmiş veya gözden kaçmış durumlar da ortaya çıkarsa yeniden çözüm üretilmelidir.

**1.1.7. Oluşabilecek Hatalar**

Bu aşamada ortaya çıkabilecek hatalar kodlama ile ilgili değil, mantıksal hatalar olacaktır. Zaten programlama diline ait deyimsel ve kelimesel hatalar (syntaxerrors) düzeltilmeden yazılan program çalışmaz. Eğer probleme getirilen çözüm mantık dışı (hatalı) ise; bu durumda hatanın kaynaklandığı yere dönülerek algoritmayı değiştirmek gerekir. Hata genellikle algoritmadan olabileceği gibi problemin tanımından da kaynaklanabilir.

**Algoritma Hazırlarken Dikkat Edilecek Hususlar:**

- 1) Algoritmada çözüm tamamıyla ifade edilmelidir.
- 2) Algoritma basit olmalıdır.
- 3) Yapılmak istenenler belirgin ve açık olmalıdır.
- 4) Algoritma genel olmalıdır.
- 5) İşlemler yapılma sırasına göre baştan sona doğru olmalıdır.

**Bilgisayarın Çalışma Mantığı (yaşar, 2011, sf:1-2)**

Bilgisayarda bilginin işlenmesi söz konusudur. Bilgisayar, kendisine girilen verileri, amaca uygun bir şekilde işleyerek veya işlemeyen saklayabilen, daha sonra istendiğinde bunların çıktısını istenen birime verebilen elektronik bir cihazdır. Yani bilgisayarda verinin veya bilginin işlenmesi söz konusudur.

Bilgi işlem ise üç aşamadan oluşan bir süreçtir. Bilgi işlem sürecinin aşamaları:

GİRDİ → İŞLEM → ÇIKTI

şeklinde özetlenebilir.

Bilgisayarın iki temel bileşeni vardır denebilir. Bunlar yazılım ve donanımdır.

Bilgisayarda tüm donanım birimleri anakart üzerine doğrudan veya dolaylı bağlanırlar. Doğrudan bağlananlara internal(dâhili), dolaylı(kablo ile veya kasanın dışından) bağlananlara external(harici) donanım denir. Dâhili donanımlar kasanın içinde, dolaylı bağlanan donanımlar ise genelde bir kablo yardımıyla kasanın dışında yer alır. İşlemci kendisine hafıza veya giriş çıkış portlarından gelen verileri, üzerinde çalıştırılan programlar yardımıyla işler ve elde edilen verileri yine program doğrultusunda, gerekirse hafıza birimlerine veya giriş çıkış portlarına gönderirler. Bilgisayardaki işlenen veriler istenirse dosya denilen veri kümeleri halinde kalıcı hafızalara da kaydedilebilir. Bilgisayardaki komut işleme performansı veriyollarının bant genişliğine, veriyollarının hızına ve disk erişimini azaltan RAM büyüklüğüne bağlıdır. Disk erişimi performansı ise daha çok sabit diskin erişim ve yazma hızına bağlıdır.

Bilgisayara verilen komutları çalıştıran donanım birimi işlemcidir. İşlemci(CentralProcessingUnit) üzerinde çalışacak kodlar, mutlaka 16'lık (hexadecimal) yapıdadır. Bu biçimdeki işlemci kodlarına **makine kodları** denir. Her bir kod işlemci üzerinde ayrı ayrı tanımlıdır. Programlar sadece işlemciden, işlemci üzerinde tanımlanmış komutları çalıştırmasını isteyebilirler. Zaten programlar, işlemcide tanımlı kodların belirli bir mantık yoluyla problemleri çözebilecek şekilde sıralanmasıyla oluşan, komut bloklarıdır.

**Bir işlemcinin çalıştırabileceği 4 temel komut grubu:**

- Aritmetiksel işlemler (toplama, çıkarma, çarpma, bölme)
- Mantıksal işlemler (ve, veya, değil...)
- Atama (veri) işlemleri (hafıza, register arası veya kendi aralarında veri transferleri)
- Program kontrol işlemleri (belirli şartlar oluştuğunda programı istenen koda yönlendirmek)

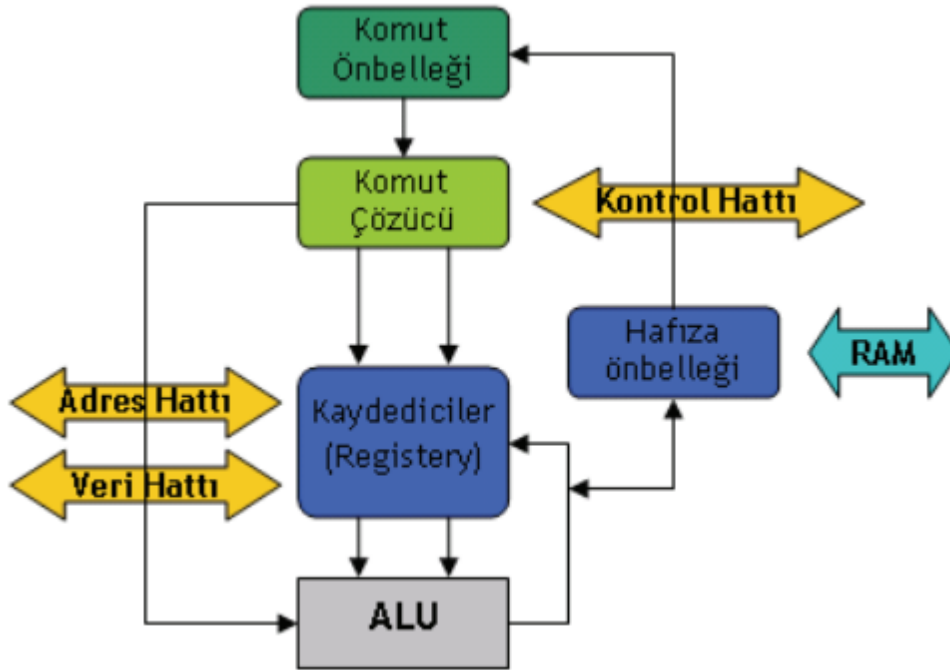


İşlemci, tüm donanım birimleri ile dolaylı ya da dolaysız etkileşim halindedir. Tüm donanım birimleri işlemcinin isteklerini yerine getiren birer hizmetçi olarak çalışır. Bazı donanımlar kendilerine has işlemciye sahiptir. Bu tip donanımlara akıllı donanım denmektedir. Bunlar işlemciye daha az yük olmaktadır. İşlemci artakalan zamanda başka görevlerini yaparak zaman kazanmaktadır.

### İşlemcinin Temel Bileşenleri:

- 1. ALU(Aritmetik ve Mantıksal İşlem Birimi):** Toplama çıkarma, çarpma, bölme, mantıksal ve, veya, değil komutları ve kaydırma komutları.
- 2. Komut Çözücü(InstructionDecoder):** İşlemcinin yapması gereken kodların icrası için gerekli işlemleri başlatır ve komutun çalıştırılması için gerekli işlemleri belirler.
- 3. Kaydediciler(Register):** İşlemci içerisinde sayıları depolamak için kullanılan hafıza çeşididir. İşlemci veri uzunluğu kadar genişliğe(32, 64 bit) sahiptirler. Literatürde test, EBX, EAX, BX, ES, IP gibi isimler alan kaydedici hafıza gözleri vardır.
- 4. Bayraklar(Flags):** İşlemlerin sonucuna göre 1 ya da 0 değerlerini alan 1 bit genişliğe sahip hafıza gözleridir. Sıfır, işaret, elde, eşlik, taşma gibi çeşitleri vardır. Örneğin bir çıkarma işleminde sonuç sıfır çıkarsa sıfır bayrağı 1 değerini alır.
- 5. Veriyolları(Buses):** İşlemcinin diğer donanım birimleri ile bağlantısını sağlayan iletken elektriksel yollardır. Üç adet veriyolu bulunur. Bunlar veri(data), adres(address) ve kontrol(control) veriyollarıdır.

İşlemciler komutları yürütürken öncelikle işletilen komut sırasını üzerinde tutan program sayıcının(PC=program counter) gösterdiği adresteki komut RAM den alınır(Fetch). Alınan komut, komut çözücü tarafından, nasıl yürütüleceği ve ne anlama geldiği belirlenir(Decode). Sonunda ise çözölen komut doğrultusunda ALU ya verilen direktifler yardımıyla istenen işlemler yaptırılır(Execute). Elde edilen sonuçlar istenen hafıza gözlerine yazılır(Write Back). Bu işlemler bir sonraki komut için benzer şekilde devam ederek işletilmesi gereken komutlar bitene kadar sürer.



İşlemcinin Temel Bileşenleri ve Çalışma Şekli (yaşar, 2011, s: 2)

## Makine Dili

Makine dili her bilgisayar sisteminin kendine has olan dilidir. Yani donanımın ana dili denilebilir. Temelde bilgisayarlar 1 ve 0'lerden oluşan ikilik sayı\* sistemindeki dili anlarlar. Buna “makine dili” denir:

```
0010101000011101
0011110010101111
0101011011010101
1101111100101001
```

Makine dilinin dezavantajları, kodları yazarken hata yapılma olasılığının fazla olması ve yazımının uzun sürmesidir. Makine dilinin daha rahat programlanması için 1950'li yıllarda “assembly dili” geliştirilmiştir. “Assembly dili” basit, hatırlanması kolay deyimlerden oluşur.

Programcılar 1 ve 0 ile program yazma yerine “assembly dili” ni geliştirmişlerdir, bilgisayarın bu yazılanlarla ilgili hiçbir fikri yoktur. Bu sebeple programcılar “assembly dili” komutlarını “makine dili” ne çeviren programlar yazmışlardır. Bu programlara “makine dili çeviricisi - assembler” denir. Böylece “assembly dili” ile yazılmış bir kod, bilgisayarın anlayabileceği “makine dili” ne dönüşmüş olur.

**Not:** Bir programı ilk denemede çalıştırmak zordur. Genel davranış olarak programı okunaklı yazarsanız, hatalara fırsat vermez, ileride bir eklenti yapmak istediğinizde zorlanmazsınız.

\*Bir çevrim örneği:

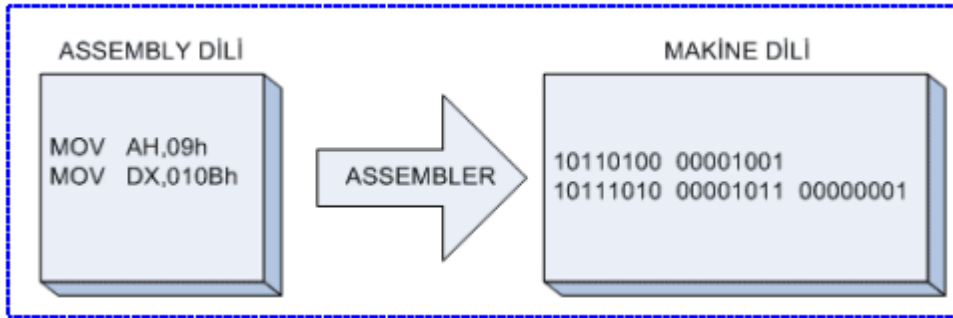
$$(1100)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 \\ = 8 + 4 = (12)_{10}$$

### Assembly dilinde kod örneği

TitleYaziProgrami

```
dosseg
.model small
.stack 100h
.data
my_message db 'Selam!',0dh,0ah, '$'
.code

    mainproc
    mov ax,@data
    mov ds,ax
    mov ah,9
    mov dx,offset my_message
    int 21h
    mov ax,4C00h
    int 21h
main endp
end main
```



Bilgisayarın anladığı dil; makine dili

**Not:** Intel uyumlu ve Microsoft’un MS DOS ve Windows işletim sistemlerine uygun bir “assembly dili” ile burada örnek verdik. Kişisel bilgisayarlar (PC), PowerPC (Macintosh), PIC ve 8051 gibi işlemci ve mikro denetleyicilerin kendilerine has dilleri ve çevirici programları vardır.

Temel olarak makine dilinde; mantıksal işlemler, aritmetik işlemler, dallanma işlemleri ve veri hareket işlemleri yapılabilir. Bu işlemleri kullanarak ister basit ister çok karmaşık programlar yazabilirsiniz.

“Assembly dili” makine dilinden daha rahat yazılmasına rağmen, hala bazı dezavantajlara sahiptir:

- Başka dile çevrilmeleri zordur.
- Çok uzun program yazımına elverişli değildir. (MEGEP, Programlama Temelleri, 2007, ss:6-7)

## İkili Sayı Sistemi ([www.gokhandokuyucu.com](http://www.gokhandokuyucu.com), ss:27-28)

Bir Bilgisayar sisteminde tüm bilgi kayıtları ve işlemleri elektriksel devreler üzerinden gerçekleştiği için tek bilinen gerçek elektrik akımının varlığı veya yokluğudur. Bu da matematiksel ve mantıksal olarak ikili sayı sistemine karşılık gelir. Çoğunlukla ikili sayı sistemindeki 0 değeri elektrik olmadığını, 1 değeri ise bir elektriksel gerilimin olduğunu anlatır. Bu iki sayısal değer (0 ile 1) ikili sayı sisteminin rakamlarıdır. Ve bilgisayarda oluşan tüm değer ve sonuçlar gerçekte bu rakamlar ile anlatılabilirler. Ancak bizim bu sayısal değerleri anlamamız zor olduğu için sayısal olarak onluk sayı sistemini kullanırız.

### Tabandan Tabana Çevrim

Böyle olunca sayıların gerektiği durumlarda tabandan tabana çevrilebilmesi gereklidir. İlköğretim düzeyinde görmüş olabileceğiniz yöntemlere burada bir değinmekte fayda bulunmaktadır. Bu rakamların her biri bilgisayarda bit denilen alanlarda tutulmaktadır.

### İkili Sayı isteminden onlu sayı sistemine çevrim:

Elimizdeki ikili sayının en sağındaki basamak sıfıncı basamak olmak kaydıyla tüm basamaklarımız sola doğru numaralandırılır. Sonra her basamaktaki sayısal değeri  $2^{\text{basamak}}$  değeri ile çarpıp ve bulunan tüm değerleri toplarız.

7	6	5	4	3	2	1	0
1	0	0	1	1	0	1	1

$$= 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 + 0 \cdot 2^6 + 1 \cdot 2^7$$

$$= 1 + 2 + 0 + 8 + 16 + 0 + 0 + 128$$

$$= 155$$

### Onlu Sayı isteminden ikili sayı sistemine çevrim:

Eldeki onlu sayı sürekli 2 değerine bölünerek işlem yapılır. Bölme işlemi en son bölümün 0 olduğu noktaya kadar devam eder. Elde edilen bölme tablosunda en son kalanlar sondan başa doğru yan yana yazılır ve sayının ikilik tabandaki karşılığı bulunmuş olur.

Örneğin elimizde 156 gibi sayısal bir değer olsun.

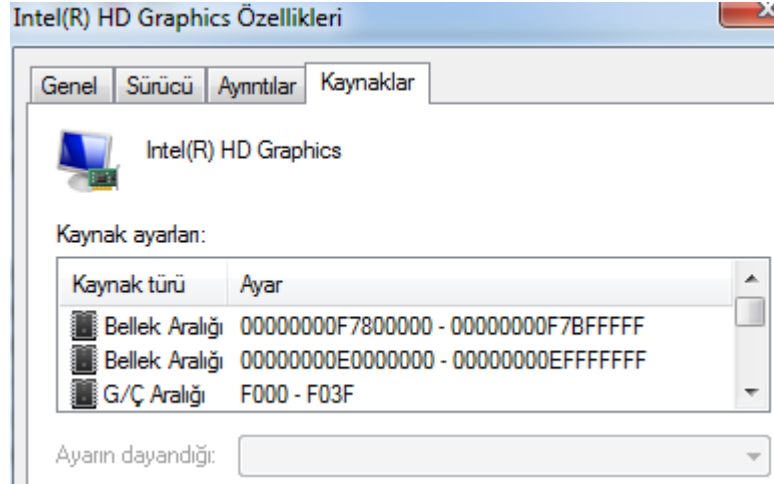
İşlem	Bölüm	Kalan
155 / 2	77	1
77 / 2	38	1
38 / 2	19	0
19 / 2	9	1
9 / 2	4	1
4 / 2	2	0
2 / 2	1	0
1 / 2	0	1

Sonuç Kalan sütunundaki değerlerin aşağıdan yukarı dizilmesi ile 1 0 0 1 1 0 1 1 olarak elde edilir.

## Onaltılı Sayı Sistemi

Sayı Sistemleri		
10'lu	16'lı	2'li
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Bilgisayar sistemleri ikili sayı sistemlerini anlasa da sayıları daha kısa bir şekilde ifade etmek için onaltı tabanındaki sayı sistemi de literatürde ve uygulamada sıklıkla kullanılır. Özellikle bilgisayar sisteminde kullanılan donanım birimlerinin ve bellek adreslerinin ifade edilmesinde onaltılı sayı sistemi kullanılır. Onaltılı sayı sisteminde bir sayının her bir basamağı 0 – 15 arasında bir değer alır ve 9'dan sonraki değerler sırası ile A, B, C, D, E ve F sembolleriyle gösterilir.



Onaltılı = Hexadecimal = Hex

Bir sayının onaltılı sayı sisteminde olduğunu ifade etmek için sayının sağ alt köşesine (16) yazılabileceği gibi (HEX), (hex), (H) veya (h) de yazılabilir.

Örneğin dokuz basamaklı  $123456789_{10}$  sayısı onaltılı sayı sisteminde yedi basamak olarak  $75BCD15_{16}$  şeklinde gösterilebilir.

### İkili Sayı Sisteminden Onaltılı Sayı Sistemine Dönüşüm

İkili sayı sistemindeki sayı en sağdan itibaren dörder basamak dörder basamak ayrılır ve her bir dört bitlik ikili sayının onaltılı sayı sistemindeki karşılığı yazılır.

**Örnek:**  $10110011101010_2 = ?_{16}$

En sağdan itibaren dört bit (kırmızı) :  $1010_2 = 10_{10} = A_{16}$

Sağdan ikinci dört bit (yeşil) :  $1110_2 = 14_{10} = E_{16}$

Sağdan üçüncü dört bit (mavi) :  $1100_2 = 12_{10} = C_{16}$

En sonda iki bit kaldı (siyah) :  $10_2 = 2_{10} = 2_{16}$

Sonuçta her bir dört bitlik grubun onaltılı sayı karşılığı sırası ile yerine yazıldığında cevap  $10110011101010_2 = 2CEA_{16}$  olacaktır.

**Onaltılı Sayı Sisteminden İkili Sayı Sistemine Dönüşüm**

Onaltılı sayı sistemindeki her bir basamak değeri ikili sayı sisteminde dört bit ile yazılır ve sonuç bulunur.

**Örnek:**  $5A2F1C_{16} = ?_2$

$5_{16} = 5_{10} = 0101_2 \rightarrow$  5 değerini ikili sayı sisteminde 3 bit ile ifade edebiliyoruz. Olsun. Biz her zaman dört bite tamamlayalım. Çünkü bu beş değeri en solda değil de arada bir yerde veya en sağda olsaydı sonuç yanlış çıkacaktı.

$$A_{16} = 10_{10} = 1010_2$$

$2_{16} = 2_{10} = 0010_2 \rightarrow$  2 değerini ikili sayı sisteminde 2 bit ile ifade edebiliyoruz. Olsun. Biz her zaman dört bite tamamlayalım. Çünkü bu iki değeri aradadır ve bunu dört bit yerine iki bit ile yazarsak sonuç yanlış çıkar.

$$F_{16} = 15_{10} = 1111_2$$

$1_{16} = 1_{10} = 0001_2 \rightarrow$  1 değerini ikili sayı sisteminde sadece bir bit ile ifade edebiliyoruz. Olsun. Biz her zaman dört bite tamamlayalım. Çünkü bu 1 değeri aradadır ve bunu dört bit yerine iki bit ile yazarsak sonuç yanlış çıkar.

$$C_{16} = 12_{10} = 1100_2$$

En sonunda bulduğumuz dört bitlik ikili değerleri basamak sırasına göre yazarız.

**5     A     2     F     1     C**  
**0101 1010 0010 1111 1100 1100**

Yani cevap  $5A2F1C_{16} = 0101\ 1010\ 0010\ 1111\ 1100\ 1100_2$

**NOT:** Altı basamaklı onaltılı sayının ikili sayı sisteminde yirmi basamakla ifade edilebildiğine dikkat edin. Demek ki neymiş? Onaltılı sayı sistemiyle sayılar ikili sayı sistemine göre daha sade bir şekilde ifade edilebiliyor muş! 😊

**RAM Bellek**

Bilgisayarın ana belleğidir ve CPU programları çalıştırırken bunu kullanır. RAM bellek örneğin her biri 1 Byte olacak şekilde küçük parçalara veya gözlere ayrılmış gibi düşünülebilir. RAM belleğin bu her bir gözü CPU ve kullanılan işletim sistemi tarafından adreslenir. Bilgisayar programları CPU tarafından işletilirken her bir veri RAM bellekte adresi belli olan bu hücrelere yazılır. Kullanılan işletim sistemine ve donanım mimarisinin 32 veya 64 bit olmasına göre RAM belleğe yazılacak verilerin kaç Byte yer tutacağı değişkenlik gösterir.


RAM Belleğin Hücreleri – Her bir hücrenin adresi vardır.

## Derleyici (Compiler)

Bir programlama dili ile bilgisayara aktarılan programın bilgisayarın anlayabileceği Makine Diline çevirmeyi sağlayan ve yazılan programda söz dizim (syntax) hatalarının (errors) olup olmadığını bulan yazılımlardır. Her Programlama dili için bir derleyici olması gerekmektedir. ([www.gokhandokuyucu.com](http://www.gokhandokuyucu.com), s:9)

Derleme hatalarını programlama dilinin kurallarına uymayan (Sözdizimi Hatası. SyntaxError) veya derleyicinin her durumda hata vermesi kesin olan (Derleyici Hatası. Compiler Error) kodlar yazıldığında gerçekleşir. Sözdizimi hatasına Örnek olarak şunu verebiliriz: (Ayrılmaz, ss:62-63)

## Bağlayıcı (Linker):

Program yazarının kütüphaneler kullandığı durumlarda veya programlarını parçalara böldüklerinde derleme işleminden sonra bu program parçacıklarını ve kütüphaneleri birbirine bağlayıp hedef işletim sisteminin anlayabileceği çalıştırılabilir hale getiren yazılımlara bağlayıcı deniyor. Kütüphaneler açısından 2 biçimde bağlama kullanılıyor. Bunlardan biri statik bağlama, diğeri dinamik bağlamadır. Statik bağlama işlemi kütüphanede bulunan program parçacıklarının bağlama esnasında hedef programın içine eklenmesidir. Dinamik bağlamada ise hedef programın içine kütüphanedeki kodlar eklenmez sadece gerekli sembolleri eklenir. Hedef işletim sisteminin yeteneklerine bağlı olarak kullanılan dinamik bağlamada kütüphaneler ayrı biçimde işletim sistemine yüklenir. ([www.ceturd.com](http://www.ceturd.com))

Bazı derleyiciler, kaynak kodu önce özel bir dosya türüne çevirir. “Object file – nesne dosyası” denilen bu dosya, kaynak dilden bağımsız hâle gelir. Farklı dilde oluşturulmuş nesne dosyaları, “linker - bağlayıcı” adı verilen programlar ile birleştirilir. Sonuçta ise çalıştırılabilir tek dosya oluşur.

Microsoft Windows, dillerin ortak kod kullanımı için DLL (DynamicLinkLibraries – Dinamik Bağ Kütüphanesi) yöntemi kullanır. Ara birimi olmayan bu dosyalar, farklı programlama dilleri kullanabilsin diye sisteme tanıtılmıştır.

Son bir yöntem, Microsoft’un .NET teknolojisidir. Aynı ara birimde ister C#, ister Basic veya başka bir dil ile program yazılabilir. Aynı veri tabanını ve çözümü ortak kullanarak birçok programcı bir arada çalışabilir. Her dilin kendine has avantajı kullanılarak, uygulamalar daha güçlü hâle gelebilir. (MEGEP, Programlama Temelleri, 2007, s:14)

## Yorumlayıcı Nedir? (Interpreter)

Derleyici gibi çalışan ancak yazılmış programları o anda Makine diline çeviren yazılımlardır. Bu tür bir yazılımda Programın Makine dili ile oluşturulmuş kısmı bilgisayarda tutulmaz. Programın her çalıştırılmasında her adım için Makine dili karşılıkları oluşturulur ve çalıştırılır. ([www.gokhandokuyucu.com](http://www.gokhandokuyucu.com))

## Nesne Kodu (Object Kod)

Derleme işleminden sonra oluşan kütüphaneler ile ve diğer nesne kodları ile bağlanmak üzere hazırlanan derlenmiş olan dosyanın makine kodlarını içeren dosyalardır. (.OBJ uzantılı dosya)

## Kütüphane (Library)

Program yazarları, işlerini yaparken bazı uzun işlemleri tekrar tekrar yaparlar. Ayrıca bazı işlemler farklı işletim sistemleri veya farklı platformlarda farklı biçimlerde yapılıyor. Bu yüzden bazı kodlamaların her ortamda aynı şekilde yapılabilmesini ve tekrarlanan uzun işlemlerin kısa çağrılarla tekrarlanabilmesini sağlayan yazılımlar bulunuyor. Bu yazılımlara kütüphane deniliyor.

## Sanal Makine (Virtual Machine)

Bu tür yazılımlar, genel olarak programların taşınabilirliklerini arttırabilmek için hazırlanan, işletim sistemi ile program arasındaki iletişimi düzenleyen yazılımlardır. Bu tür sanal makineler sayesinde bir yazılım bir kez derlenerek birden farklı platformda sanal makine ile çalıştırılabilir hale getirilir. Burada önemli olan istenen platform için özel olarak bu sanal makinenin bulunmasıdır. Bu sanal makinenin çalışmadığı bir platformda bahsedilen derlenmiş yazılım çalışmayacaktır. ([www.ceturd.com](http://www.ceturd.com))

## Yazılım

Bir işin, uygulamanın bilgisayarla yapılması için tasarlanarak uygulanan ve kaydedilen komutların tamamına yazılım denir. Yazılım, bilgisayar donanımdan yararlanmayı sağlayan, donanımın hangi durumda nasıl davranacağını bildiren bilgisayar programlarıdır. Kullanım amacına göre çok çeşitli yazılımlar vardır:

- İşletim Sistemleri (Windows, Unix, Novell Netware, Linux, MacOS, Android...)
- Paket Programlar (Autocad, MS Word, ...)
- Fayda Tabanlı (Utility) Programlar (Winrar, Nero Burning Rom, ...)
- Programlama Dilleri (Assembly, C++, C#, Java, ...)

Bir yazılım tasarlanırken ve kodlanırken bazı kalite özellikleri göz önünde bulundurulmalıdır.

### Bir Yazılımın Kalite Özellikleri:

#### Kullanıcı Açısından:

- İşini doğru yapmalı. Yararlı ve kullanışlı olmalı.
- Gerektiği kadar hızlı çalışmalı.
- Sistem kaynaklarını (işlemci, bellek, disk kapasitesi, ağ kapasitesi) çok fazla harcamamalı.
- Güvenilir olmalı. (Hassasiyet ve her durumda aynı davranış)
- Kolay güncellenebilmeli.
- Kullanım kılavuzu (user manual) yeterli olmalı.

#### Yazılım Geliştirici Açısından:

- Kaynak kod okunabilir ve anlaşılabilir olmalı.
- Yeni ihtiyaçlara göre bakımı ve güncellemesi kolay olmalı.
- Yazılımın bir bölümündeki hata, diğer bölümlerini etkilememeli.
- Yazılımın modülleri gelecekteki başka projelerde de kullanılabilirmeli.
- Bir yazılım projesi son teslim tarihinden önce bitirilebilmeli.
- Yazılım geliştirme ile ilgili yeterli açıklayıcı belgeye sahip olmalı.

## Programlama Dili

İster genel ister özel amaçlı olsun tüm uygulama ve sistem yazılımları programlama dilleriyle yazılır. Bir programlama dili, insanların bilgisayara çeşitli işlemler yaptırmasına imkân veren her türlü sembol, karakter ve kurallar grubudur. Programlama dilleri insanlarla bilgisayarlar arasında tercümanlık görevi yapar. Programlama dilleri, bilgisayara neyi, ne zaman, nasıl yapacağını belirten deyim ve komutlar içerir.

Bir programlama dili şunlardan oluşur.

- **Genel komutlar:** Programlama dilinin anlayacağı komutlardır.
- **Gelişmiş komutlar:** Genel komutları kullanarak oluşturulmuş komutlardır.
- **API(Application Programme Interface) komutları:** İşletim sisteminin sunduğu özellikleri kullanan komutlardır.
- **Derleyici komutları:** Komut içinde çalışmayıp derleme esnasında alınan bilgilere göre derleme yapılmasını sağlar.
- **Aktif nesneler:** 'Buton, Menü, Gösterge çubuğu ve Tabpanel' gibi bileşenlerin genel adıdır. (MEGEP,482BK0123,2011, ss:8-9)

## Programlama Dili Çeşitleri

Bir programlama dili ya insan ya da makine anlayışına yakındır. İnsan anlayışına daha yakın programlara dillerine yüksek seviyeli programlama dilleri, makineye yakın olanlara ise düşük seviyeli programla dilleri denir.

- Yüksek seviye programlama ile yazılan projelerin kaynak kodları kısa, derlenmiş hâlleri ise uzun olur. Çalışma hızları ise yavaştır.
- Alçak seviye programlama ile yazılan projelerin kaynak kodları uzun, derlenmiş hâlleri ise kısadır olur. Çalışma hızları ise en yüksek seviyededir.



Programlama dillerini seviyelerine göre 5 ana gruba ayırabiliriz:

- **Çok yüksek seviyeli diller ya da görsel diller**  
Access, Foxpro, Paradox, Xbase, Visual Basic, Oracle Forms, .NET platformları
- **Yüksek seviyeli diller (Bunlara algoritmik diller de denir.)**  
Fortran, Pascal, Basic, Cobol
- **Orta seviyeli diller**  
C, C++(C Plus Plus) , C#(C Sharp) Orta seviyeli diller daha az kayıpla makine diline çevrilebildiğinden daha hızlı çalışır.
- **Alçak seviyeli programlama dilleri**  
Sembolik makine dili (Assembly).
- **Makine dili**  
En aşağı seviyeli programlama dilidir (Saf makine dili tamamen 1 ve 0 lardan oluşuyor.).

Kuşak	Programlama Dili	Periyod
1	Makine dili	1940 – 1950 arası
2	Assembly dili	1950’li yıllardan itibaren
3	Yüksek seviyeli diller	1960’lı yıllardan itibaren
4	Çok yüksek seviyeli diller	1970’li yıllardan itibaren
5	Yapay zekâya yönelik diller	1980’li yıllardan itibaren

**Tablo : Programlama dillerinin tarihi gelişimi**

Kaynak: MEGEP,482BK0123,2011, ss:9-10

## Akış Diyagramı (Flow Chart)

Algoritma birden çok şekilde ifade edilebilir. Bunlar:

- Metinsel olarak düz
- Genel programlama dili deyimleriyle
- Akış diyagramları

### A) Metinsel Olarak Düz İfade

Algoritma adımları alt alta sıralı bir şekilde dizilirler. Başlarında satır numaraları olabilir.

Girilen sayının tek mi çift mi olduğunu bulan algoritma:

Birinci çözüm:

1. Başla
2. Bir sayı gir ve tamsayı A olarak al.
3. A'nın 2'ye bölümünden kalanı hesapla (kalan = A mod 2)
4. Kalan=0 ise ekrana "çift sayı" yaz
5. Kalan=1 ise ekrana tek sayı yaz
6. Bitir

İkinci Çözüm: (MOD ALMA İŞLEMİ YAPMADAN)

```

10 BAŞLA
20 Tamsayı A gir //0-3-4 - (-5) - yedi
22 EĞER A=0 ise "Sıfır değerlendirme dışıdır!" ve GİT 40
25 Eğer A>0 ise GİT ADIM 50
30 Eğer A<=0 ise ekrana "pozitif tamsayı giriniz" yaz.// A=|A| (veya A=-1*A) GİT 50
40 Başka deneme yapılacak mı? Evet ise adım 20'ye git. Hayır ise ADIM 98'E GİT.
50 B=A //B=3, B=4
60 B=B-2 //B=3-2=1, B=1-2=-1, B=4-2=2, B=2-2=0
70 EĞER B=0 İSE "A SAYISI ÇİFTTİR" YAZ. ADIM 40'A GİT.
80 EĞER B<0 İSE "A SAYISI TEKTİR" YAZ. ADIM 40'A GİT.
90 ADIM 60'A GİT.
98 BİTİR.

```

### B) Genel Programlama Deyimleri İfadesi (Sözde Kod, Pseudo kod, Kaba kod)

Programlama dillerinde kullanılan deyimleri fazla açık ve ayrıntılı olarak kullanmaksızın ifade şeklidir. Bazı kitaplarda sıkça karşılaşılan bir yöntemdir. Genelde Pascal ve C deyimleri kullanılarak ifade edilir. Algoritmanın programa dökülmesi bu ifade şeklinde daha kolaydır. Yarı deyim yarı metinsel olarak ifade edilir. İşlem akışı programlama dili deyimleriyle ifade edilir. Aşağıda Pascal programlama dilinde ifade edilmiş bir sayının tek veya çift olduğuna karar veren işlem basamakları gösterilmektedir.

Begin

Klavyeden A'yı oku

İf A%2=0 then

Ekrana çift yaz

Else

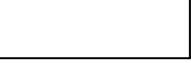
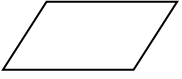
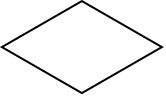
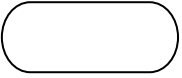

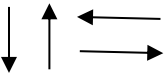

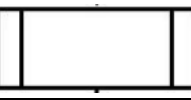

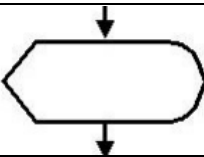

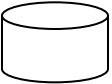
Ekrana tek yaz.

End

**C) Akış Diyagramları (FlowCharts) ile İfadesi**

Kendine has standart bazı şekillerle algoritmanın ifade biçimidir. Bu semboller uluslararası olup, geneldir. Dolayısıyla metinsel olarak ifadeden daha anlamlı ve daha açıktır. Bu yöntemle oluşturulan bir algoritma dünyanın her yerinde rahatlıkla anlaşılır. Bu sembolleri kullanma açısından kitaptan kitaba bazı farklılıklar olabilir.

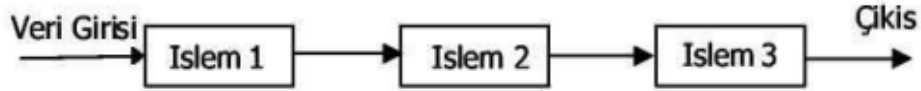
**Akış Diyagramı Şekilleri**

Şekil	Anlamı
	İşlemleri ve atamaları gösterir
	Giriş ve çıkış işlemleri için
	Karar işlemleri için
	Program başlangıcı ve sonu için
	Bağlama işlemleri için
	Olayların akış yönünü belirlemek için
	Tekrarlı işlemleri göstermek için. (Döngü)
	Önceden tanımlı işlemler için. İşlevler (fonksiyonlar – functions)
	Klavyeden Bilgisayara bilgi girilecek konumu belirten şekildir . Girilecek bilginin hangi değişkene okunacağını kutu içerisine yazabilirsiniz.
	Algoritmada bir bilginin ekrana yazılacağı konumu gösteren şekildir. Ekrana yazılacak ifade ya da değişken bu şekil içerisine yazılır.
	Bilginin Yazıcıya yazılacağı konumu gösteren şekildir.
	Kütük, dosya veya veri tabanı

## 2.1 Akış Diyagramının Kullanımı, Takibi ve Akış Diyagramında Hata Kontrolü

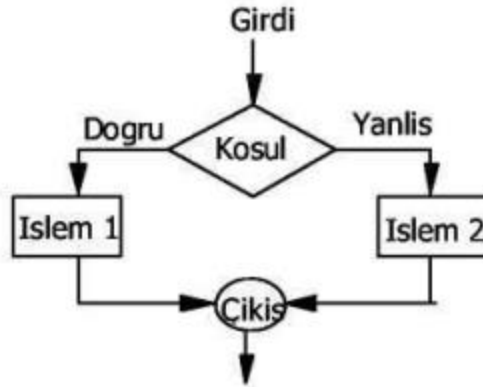
Ayrıntılı bir akış diyagramı, yazılımı oluşturan işlemleri ve ilişkilerini en küçük detayına kadar belirler.

Bir bilgisayar programının geliştirilmesinde kullanılan programlama dili ne olursa olsun bu programların akış diyagramlarında genel olarak yalnız üç basit mantıksal yapı kullanılır. Bu mantıksal yapılardan en basiti *sıralı yapıdır* (**Şekil 1.2**). Sıralı yapı, hazırlanacak programdaki her işlemin mantık sırasına göre nerede yer alması gerektiğini vurgular. Bu yapı sona erinceye kadar ikinci bir işlem başlayamaz.



**Sekil 1.2 Sıralı Yapı**

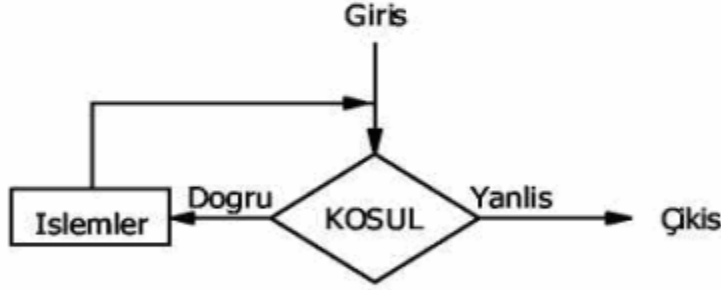
Mantıksal yapılardan ikincisi *Karar Verme* yapısıdır (**Şekil 1.3**). Programlama sırasında If...Then... Else yapısı ile tanıyacağımız bu mantıksal yapılar, birden fazla sıralı yapı seçeneğini kapsayan modüllerde, hangi şartlarda hangi sıralı yapının seçileceğini belirler.



**Sekil 1.3 Karar Verme Yapısı**

Üçüncü mantıksal yapı çeşidini *tekrarlı yapılar* (**Şekil 1.4**) oluşturmaktadır. Bu yapılara Pascal programlama dilinde *For* (**Şekil 1.4**), *While* ve *Repeat..Until* yapısı adı da verilir. Şartlara göre değişik işlem gruplarının yapılmasını sağlar. Bu yapı yukarıda sözü edilen iki yapının çeşitli kombinezonların tekrarlanmasından oluşmuştur.

Söz konusu üç değişik yapı, değişik kombinezonlarda kullanılarak istenilen işlevleri yerine getirecek programlar hazırlanabilir. Programların bu üç basit yapı ile sınırlandırılması program modüllerinin daha kolay tasarlanmasını sağlar.



**Şekil 1.4.** Tekrarlı yapılar

Bazen Bir takım algoritmaların ne işe yaradığını anlamak veya algoritmanın doğru çalışıp çalışmadığını test etmek için algoritmayı çalıştırmak gereklidir. Algoritmayı çalıştırmak demek algoritmanın adımlarını sıra ile uygulamak, oluşan değişken değerlerini bir tablo üzerinde göstermek demektir.

1. BAŞLA
2. A OKU
3. B OKU
4. C OKU
5. TOP=0
6. SAY=A
7. TOP = TOP+SAY
8. SAY=SAY+C
9. EĞER SAY<=B İSE 7. ADIMA GİT
- 10.TOP YAZ
- 11.SON

Şeklinde verilmiş bir algoritmamız olsun. Bu algoritma için  $A \rightarrow 3$ ,  $B \rightarrow 12$  ve  $C \rightarrow 2$  değerleri girilince SAY ve TOP değişkenlerinde hangi değerlerin oluşacağını algoritmayı adımlayarak gösterelim.

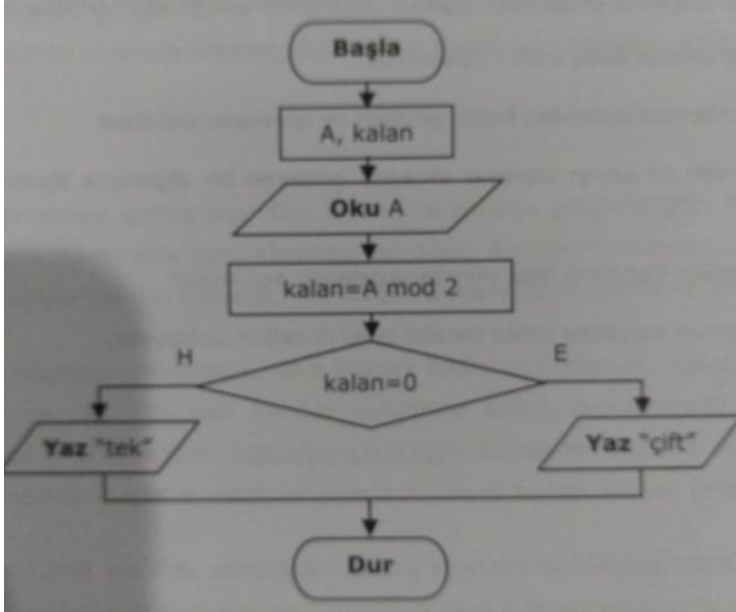
Değişkenlerin Her birinin Değeri					Açıklama
A	B	C	TOP	SAY	
3	12	2	0	3	6. adıma kadar programın ilk çalıştırılışında değişkenlerin elde ettiği değer
			3	5	7. ve 8. adımların çalıştırılmasından sonraki değerler
			8	7	7. ve 8. değerler tekrar çalıştırılıyor
			15	9	9<=12 olduğu için 7. ve 8. tekrar çalıştırılıyor.
			24	11	11<=12 olduğu için 7. ve 8. tekrar çalıştırılıyor.
			35	13	13<=12 olmadığı için algoritma 10. satırdan çalışmaya devam edecektir. Ve 10. satırdaki ifadededen dolayı ekrana 35 değeri yazılacaktır.

Değişkenlerin Her Birinin Değeri					AÇIKLAMA
A	B	C	TOP	SAY	
3	12	2	0	3	6. ADIM
3	12	2	3	3	7. ADIM
3	12	2	3	5	8. ADIM
3	12	2			9. ADIM 7. ADIMA GİT
3	12	2	8	7	2. 7. VE 8. ADIM
3	12	2			2. 9. ADIM 7'YE GİDER
3	12	2	15	9	3. 7. VE 8. ADIM
3	12	2			3. 9. ADIM 7'YE GİDER
3	12	2	24	11	4. 7. VE 8. ADIM
3	12	2			4. 9. ADIM 7'YE GİDER
3	12	2	35	13	5. 7. VE 8. ADIM
3	12	2			5. 9. ADIM 10'A GİDER
					10. ADIM SONUNDA EKRANA 35 YAZAR
					11. ADIM SONUNDA BİTER

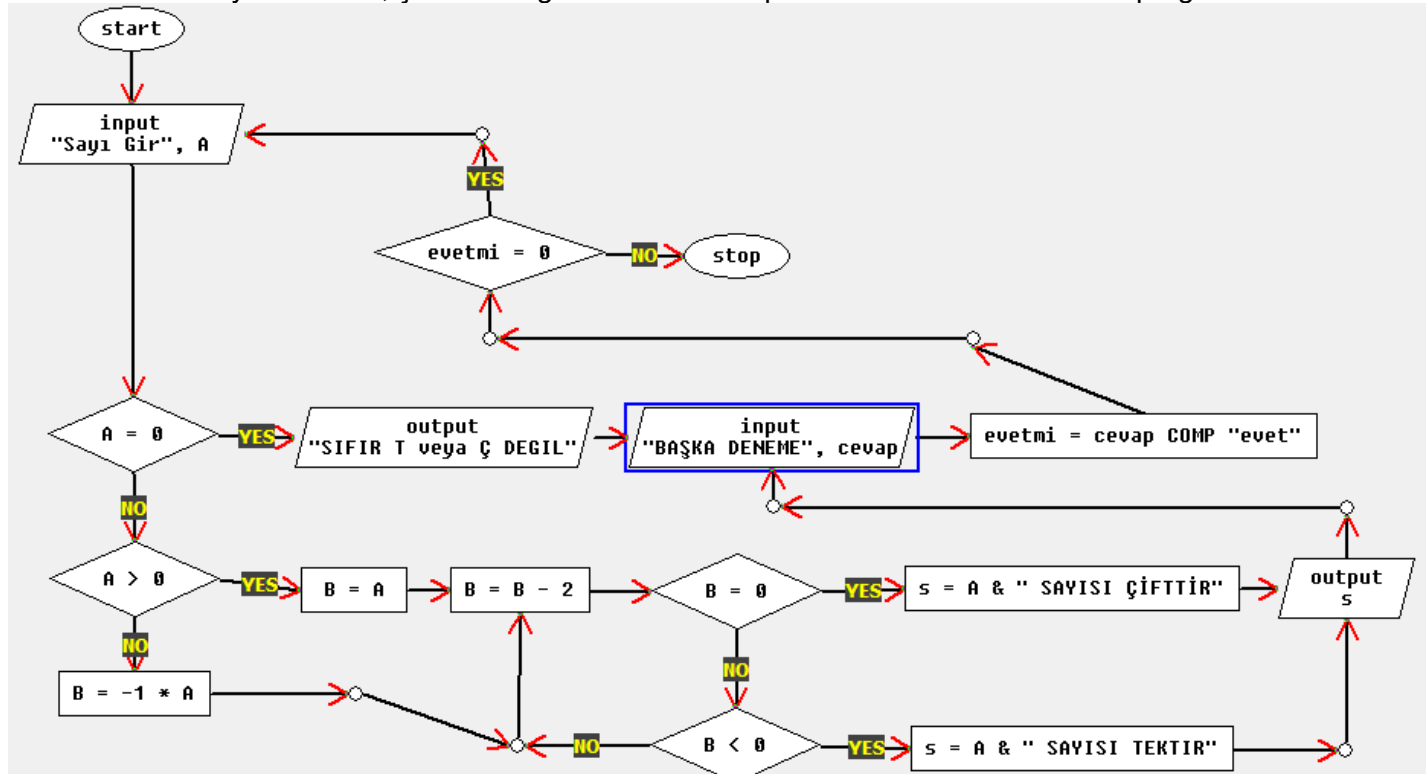


## PROGRAM ÇALIŞTIRMA ve GÜNLÜK HAYATTAN ALGORİTMA ÖRNEKLERİ

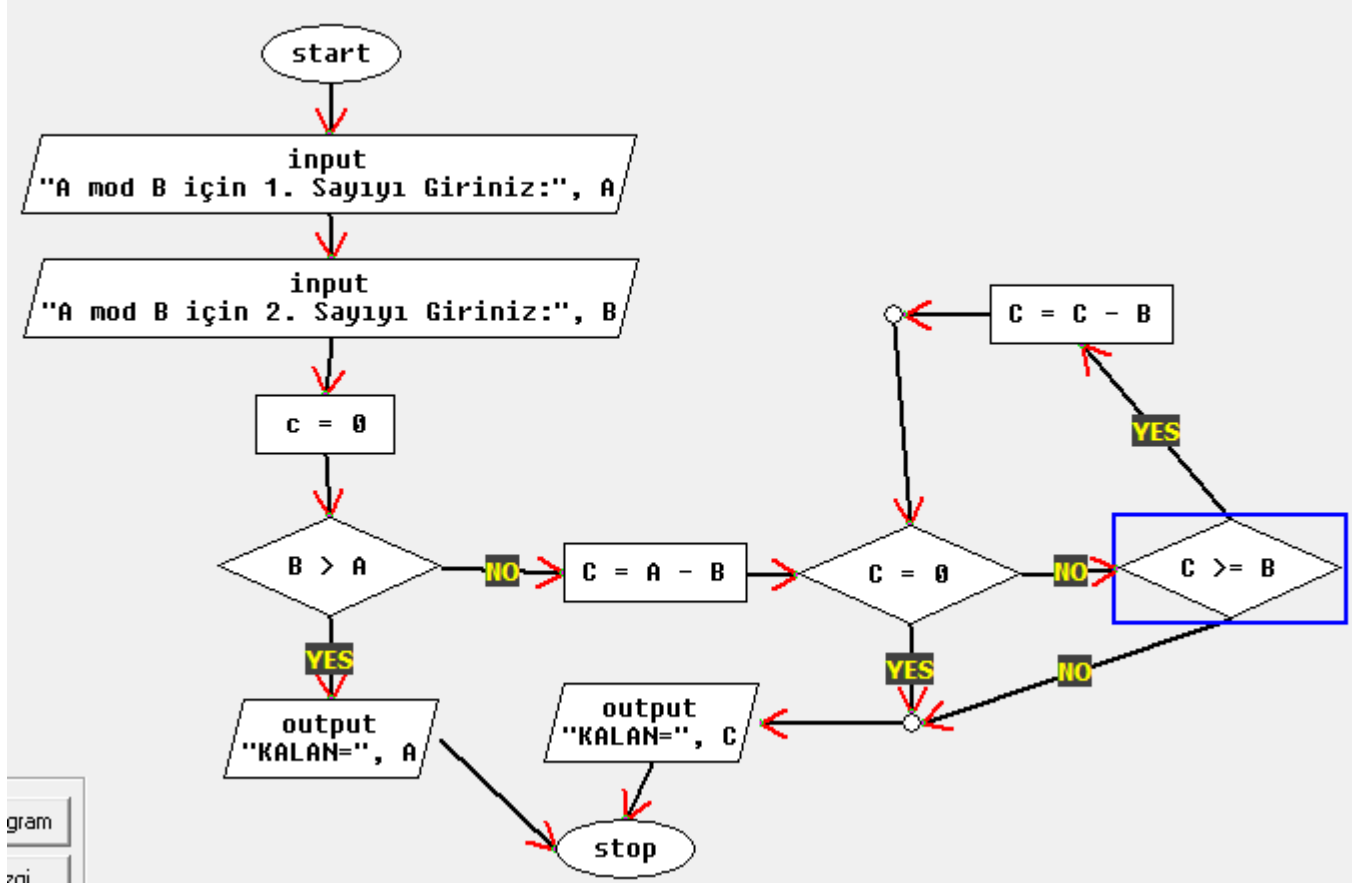
**Örnek:** Girilen sayının tek mi, çift mi olduğunu mod alma operatörü kullanarak bulan program.



**Örnek:** Girilen sayının tek mi, çift mi olduğunu mod alma operatörü kullanmadan bulan program.

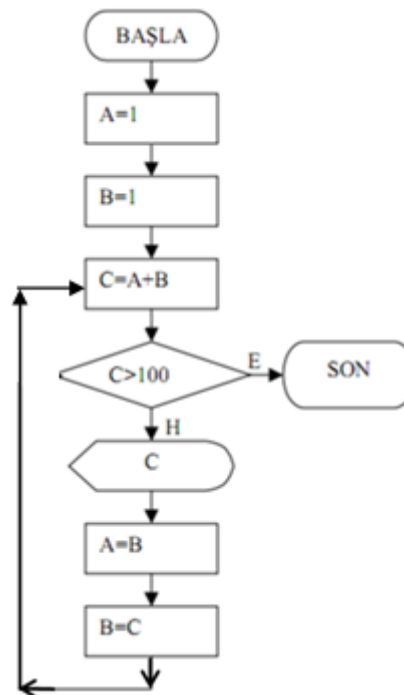


**Örnek:** Pozitif A sayısının pozitif B sayısına bölümünden kalanı MOD operatörsüz hesaplayan akış diyagramı.



**Örnek:** 100'e kadar sayılardan Fibonacci dizisinin elemanlarını listeleyen akış diyagramı. Not: Fibonacci dizisi iki adet bir ile başlar ve dizinin bir sonraki elemanının değeri kendinden önceki iki elemanın değerinin toplamı olur.  $F(n) = F(n-1) + F(n-2)$

1	1	2	3	5	8	13	21	34	.....
---	---	---	---	---	---	----	----	----	-------



1'DEN 10'A KADAR OLAN FİBONACCİ SAYILARINI EKRANA YAZAN ALGORİTMA:

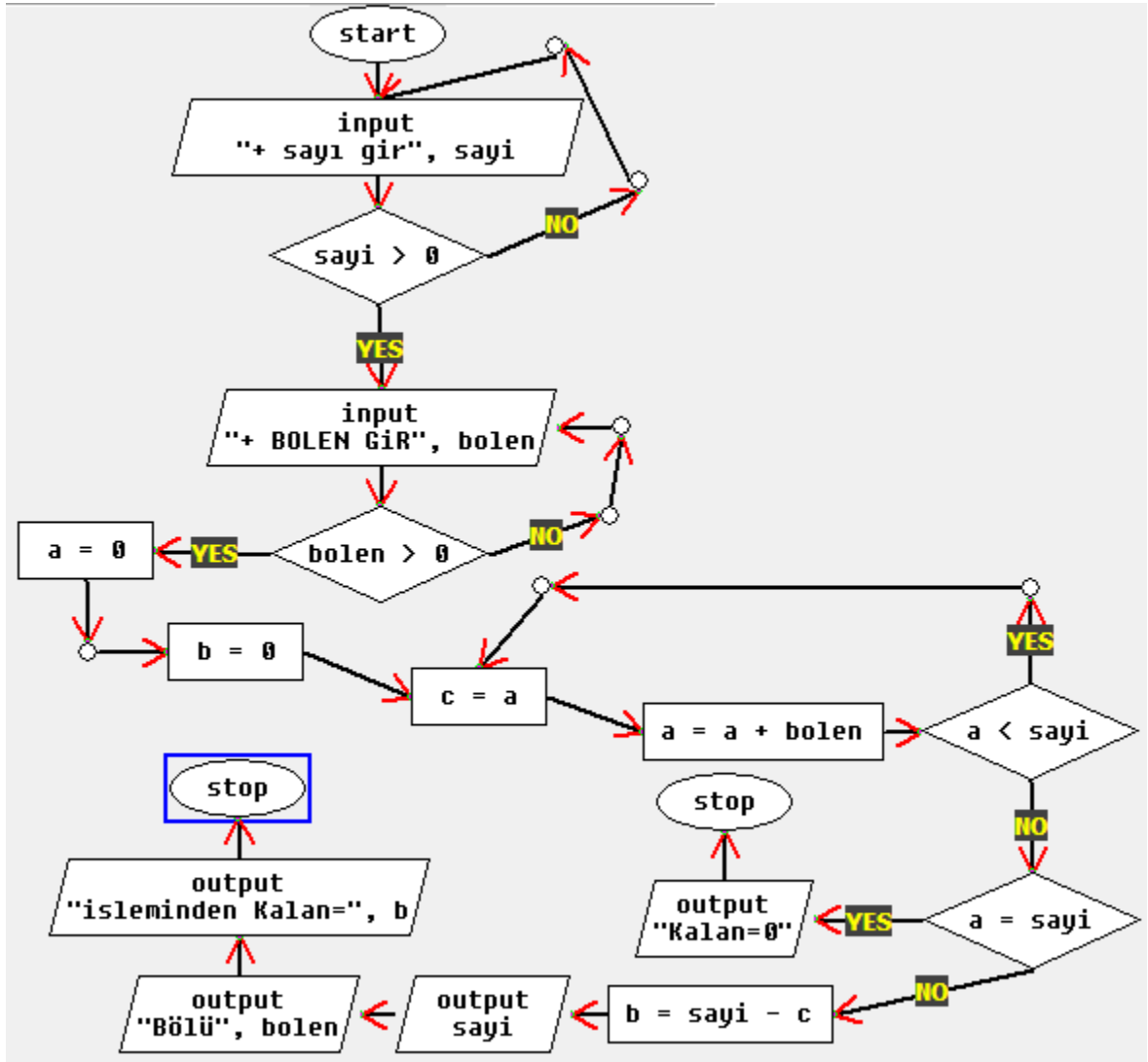
1. BAŞLA
2. A=1
3. B=1
4. EKRANA YAZ 1-1
5. C=A+B
6. EĞER C>10 BİTİR
7. EKRANA YAZ “-” E.YAZ C
8. A=B
9. B=C
10. ADIM 5'E GİT

Çevrim	A	B	C	Açıklama	ekran	
1.	1	1	2		1-1	-2
2.	1	2	3		-3	
3.	2	3	5		-5	
4.	3	5	8		-8	
5.	5	8	13	BİTİR		

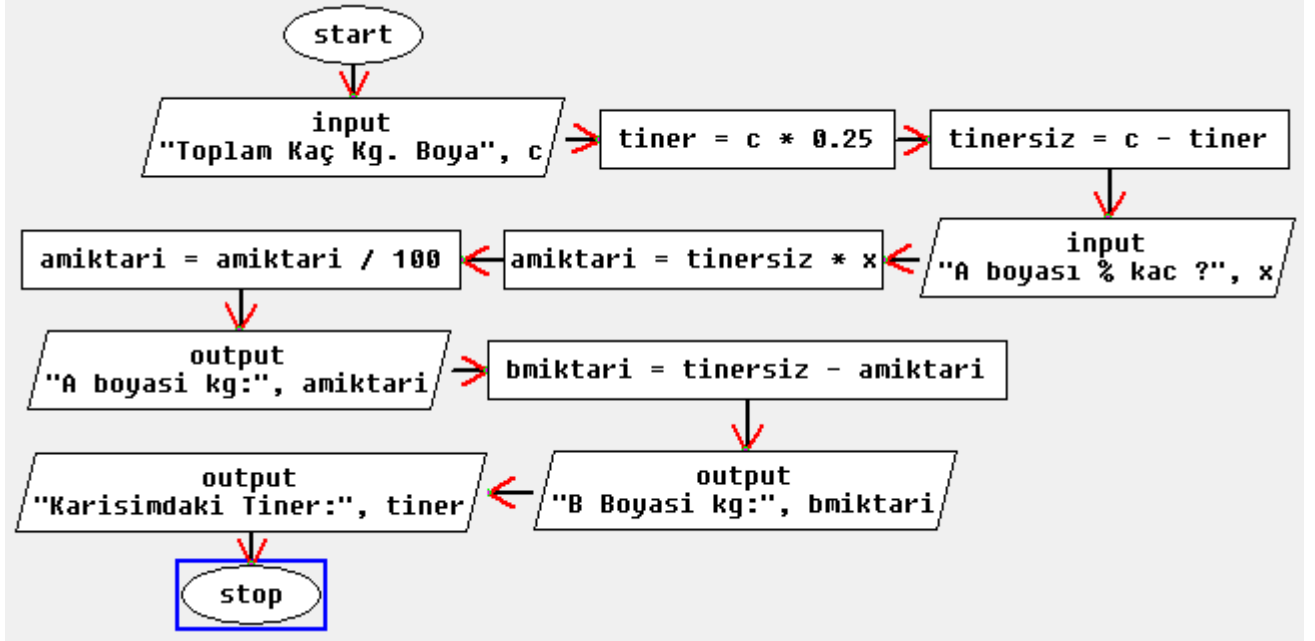
Ders Kitabı, yaşar, ss: 97 - 109

(Ders kitabı: yaşar, sf.105 → Günlük hayattan algoritma örnekleri, sf.127-160 arası → Tamsayı problemleri,)

Mod alma operatörü kullanmadan kalan bulan programın algoritma akış diyagramı



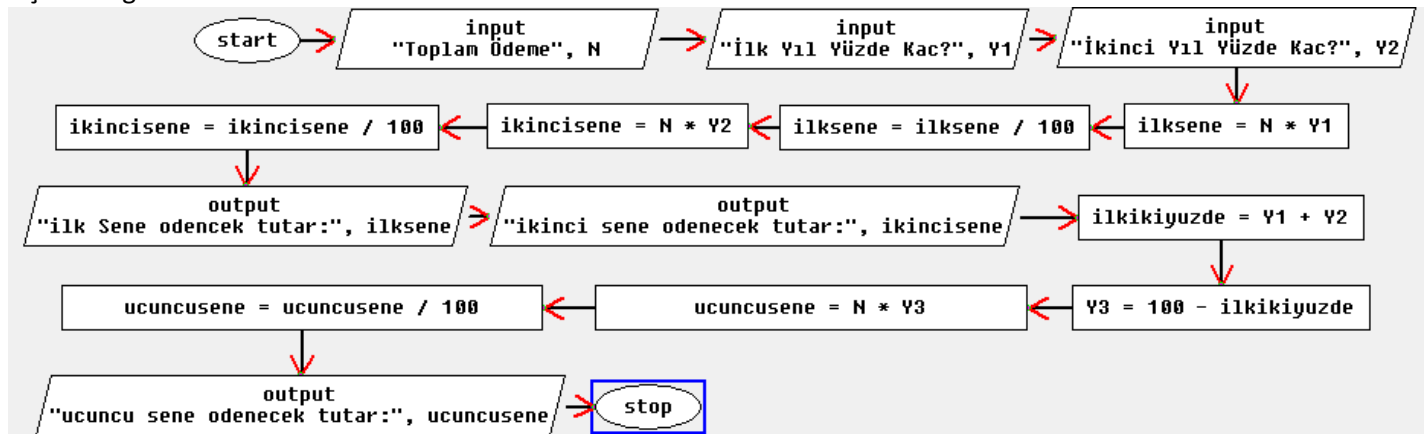
**SORU:** A ve B boyları karıştırılarak C boyası elde ediliyor. C boyası; A'dan %X birim, B'den %Y birim ve toplam karışımın ¼'ü oranında tiner katılarak elde ediliyor. Örneğin toplam 100 kg C boyası elde edilmek isteniyor ise bunun yüzde kaçının A boyası, yüzde kaçının B boyası olacağı kullanıcıya soruluyor.  $100\text{kg} * \frac{1}{4} = 25\text{ kg}$ . tiner kullanılacağı ise zaten baştan bellidir. Tiner hariç toplam karışım ise  $100 - 25 = 75\text{ kg}$ . olacaktır. Örneğin X için %40 ve Y için %60 girilmiş ise bu durumda  $75 * 0,40 = 30\text{kg}$  A boyası ve  $75 * 0,60 = 45\text{ kg}$  B boyası kullanılacaktır. Buna göre C, X ve Y değerleri kullanıcı tarafından girildiğinde kullanılması gereken A, B ve tiner miktarlarını hesaplayarak ekrana yazan bilgisayar programının algoritma akış diyagramını çiziniz. (Ders kitabı sayfa 107)



**SORU:** Elektrikli bir çaydanlıkta bulunan sıvıyı X°C'de Y dakika boyunca ısıtmak için gerekli algoritmanın akış diyagramını çiziniz. **NOT:** X ve Y değerleri kullanıcı tarafından girilecektir. (Ders kitabı sayfa 108)

**SORU:** Bir gaz ocağına konan X (örneğin 3) kg. yemek önce Y (örneğin 3) dakika kısık ateşte devamlı karıştırılarak pişiriliyor. Sonra kabın içine yemeğin %Z'si (örneğin 1/3'ü) kadar su ilave ediliyor. Daha sonra ise ağız kapalı halde ve karıştırılmadan W (örneğin 10) dakika pişiriliyor ve gaz kesiliyor. Bu işlemlerin algoritma akış diyagramını çiziniz. (Ders kitabı sayfa 109)

**SORU:** Bir futbol takımı yeni transfer ettiği oyuncu için 3 yıllık sözleşme karşılığı N lira ödeme yapacaktır. Ödemenin yüzde olarak Y1 oranındaki miktarı ilk yıl, Y2 oranındaki miktarı ikinci yıl ve kalanı üçüncü yıl yapılacaktır. Futbolcuya her yıl ödenecek para miktarını hesaplayan programın algoritma akış diyagramını yazınız. **NOT:** N, Y1 ve Y2 değerleri dışarıdan girilecektir.



**Ödev Soru:**

Otomobil için 2 saat: 5tl sonraki her saat 1 tl 10 saatten sonrası günlük sabit ücret...

Kamyon için 2 saat 8 tl, sonraki her saat 2 tl 8 saatten sonra günlük sabit ücret

İş makinası 2 saat 12 tl sonraki her saat 3 tl 8 saatten sonra günlük sabit

Otoparka gelen bir aracın kalma süresi ve araç türüne göre çıkış ücretini hesaplayan programın algoritmasını yazınız veya çiziniz veya kodlayınız.

Otomobil 18 saat kalıyor. İlk2saat=5, sonraki saatlerin 10'a tamamlayana kadarına saat başı 1 TL=(10-2)\*1=8TL TOPLAM=13 TL öder.

Otomobil 32 saat kalırsa? 24 saat için 13 TL, sonraki günün ilk 2 saati için 5TL, geri kalan 6 saat için saat başına 1 TL'den 6 TL, TOPLAM=13+5+6=24 TL ÖDER.

İPUCU: Kaç saat hesap edileceği için MOD 24 işlemini kullanabilirsiniz. Kaç gün kaldığını bulmak için de /24 yapabilirsiniz. Çünkü **programlamada tamsayı/tamsayı her zaman tamsayıdır.**

```
int sure = 47;
Console.WriteLine(sure+" saat kalan bir araç:");
Console.WriteLine(sure/24+" gün");
Console.WriteLine(sure%24+" saat\nkalmıştır.");
Console.ReadLine();
```

**Ödev Soru:** Boy, kilo ve cinsiyet bilgileri girildiğinde ideal kiloda olup olmadığını ekrana yazan programın algoritmasını yazınız veya çiziniz veya kodlayınız.

- Kadınlarda ideal kilo:  $45.5 + 2.3 \times (\text{İnç cinsinden boy} - 60)$
  - Erkeklerde ideal kilo:  $50 + 2.3 \times (\text{İnç cinsinden boy} - 60)$
- 1 inç = 2,54 cm .....

## PROGRAMLAMANIN TEMEL YAPI TAŞLARI

- Sabitler (Constants)
- Değişkenler (Variables)
- Karar yapıları (Conditions)
- Döngüler (loops)
- Diziler (Arrays)
- Fonksiyonlar (Functions – Methods)

## VISUAL STUDIO ve C# GENEL YAPISI

### .NET FRAMEWORK NEDİR?

Bir platformdur. Masaüstü, web ve mobil uygulama geliştirirken birçok programlama dilini destekleyen derleyicileri ve hazır programlama dili kütüphanelerini içinde barındırır.

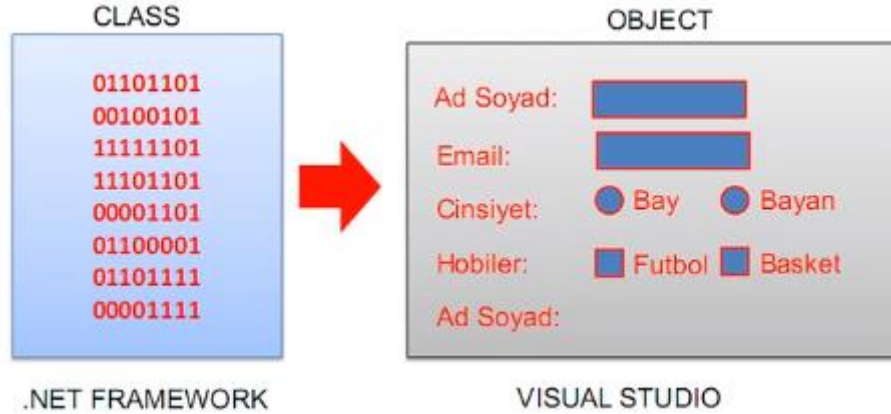
Programcının daha kolay ve hızlı program yazmasını sağlar.

Düğmeler, metin kutuları, formlar ve daha birçok nesne hazır olarak gelir.

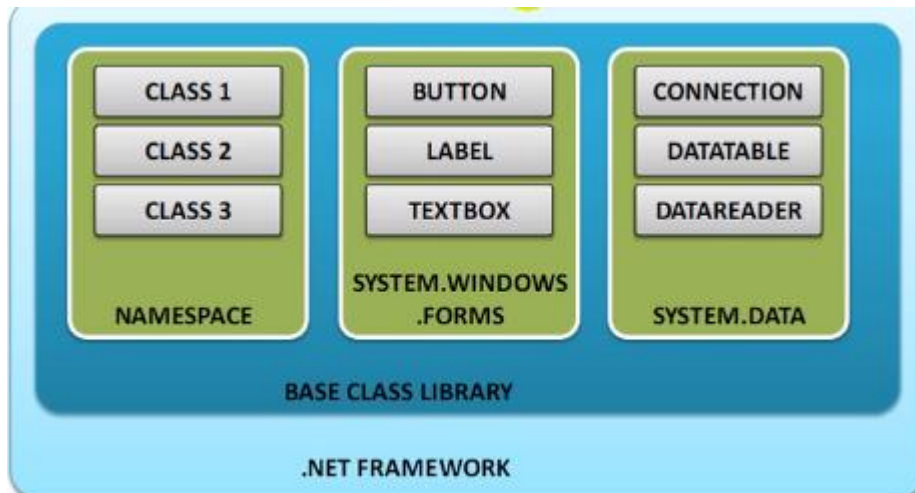
Nesne yönelimli bir platformdur.

**Sınıf (Class):** Soyut bir kavramdır. Ortak özellikleri olan nesnelerin tanımlandığı kodlar denebilir. Örneğin insan sınıfı, masa sınıfı gibi. Yazılımda ise örneğin düğme (button), checkbox, radio button. .NET Framework'te bulunan bir kod topluluğudur. Biz bunu alır ve programdaki formun üzerine koyar ve kullanırsak nesne olur. Sınıf, .NET Framework'te kullanılmayı bekleyen hazır kodlar olarak düşünülebilir.

**Nesne (object),** ait olduğu sınıfın somut bir örneğidir. Nesnelerin kendilerine özgü özellik (property), metot (function – method) ve olayları (event) vardır.



**Namespace,** sınıfları kategorize eden sanal klasörlerdir. Sınıflar yaptıkları işe göre kategorize edilmiş ve namespace'leri içine konmuştur.



**Using** komutu ile yazılacak kodun özelliğine göre namespace ler programın içine çağırılır ve bunların içindeki sınıflar kullanılabilir. Örneğin program içerisinde veri tabanı komutları kullanılacaksa system.data namespace'i çağırılmalıdır.

Sık kullanılan namespace'ler:

- SYSTEM
- SYSTEM.COLLECTIONS
- SYSTEM.WEB
- SYSTEM.WEB.UI
- SYSTEM.FORMS
- SYSTEM.DATA

**Özellik (Property):** Nesnelerin renk, boyut, konum vb. niteliklerini değiştirmeye yarayan parametre. Visual studiodaki buton gibi nesnelerin özellikleri Properties bölümündendeğiştirilebilir.

```
Nesne.özellik=değer
Arsa.genişlik=100 metre
Arsa.derinlik=50 metre
Öğrenci.No=182238027
Öğrenci.Ad=Murat
```

**Metot (Method):** Nesnelerin yapabileceği faaliyetler, işler. Nesnelere iş yaptırmak için kullanılırlar.

```
Öğrenci.DersÇalış()
Öğrenci.DerseGir()
Öğrenci.KayıtYenile()
```

**Olay (Event):** Nesnelerin dışarıdan gelen etkilere gösterdiği tepki. (sağ tık, sol tık, çift tıklama, vb.)

Nesneye yönelik program geliştirme ortamlarında yazılan tüm kodlar **metot** içerisine yazılır. Metot ise eskiden alt program veya fonksiyon(işlev) olarak bilinen yapılara karşılık gelir. Metotlar ise bir araya gelerek **sınıf(class)** denen yapıları oluşturur. C#'ta sınıflar **namespace (ad alanı)** denen yapılarda bulunur ve bunlar **sınıf kütüphaneleri** olarak görev yapar. C#'ta **using** (kullanım) sözcüğü ile bu tanımlamalar yapılabilir.

C#'ta komut satırı parametrelerini yakalamak için main metodunda giriş parametresi olarak **args** String değişkeni kullanılır.

Örnek(yaşar, sayfa 35)

```
/*
    Bu program şu amaçla yazılmıştır, bu amaçla yazılmıştır.
*/
using System; // using ile Kullanılacak namespace tanımlamaları yapılır.
namespace PROGTEMEL
{
    //sınıf tanımlama alanları
    class Program
    {
        //alt metot tanımlama alanları
        static void Main(string[] args) //komut satırı parametreleri için args değişkeni
        {
            //program kodları bu alana yazılır.
            int b; //b adında tamsayı veri tipinde bir değişken tanımlanıyor.
            for (b = 1; b<=5; b++)//Aşağıdaki kodu 5 kere çalıştırır
            {
                Console.WriteLine("MCBÜ MYO");//ekrana MCBÜ MYO yazdırılıyor ve imleç bir alt satıra geçiyor.
            }
            Console.ReadLine(); //Klavyeden enter tuşuna basılana kadar yazılan karakterleri okuyor.
        }
    }
}
```



```
//temel sistem kütüphanesi
using System;
//proje sanal klasörü.
namespace merhaba
{
//program sınıfı. wrapper
    class Program
    {
//main metodu.
//public : metodun namespace içindeki tüm sınıflardan ulaşılabilmesini sağlar
//static: metodun sınıfı nesnesi oluşturulmadan dışarıdan çağırılabilmesini sağlar
        //void: metodun dönüş türünü belirler. burada void = boş tür demektir.
        //metod isminden sonra gelen parantez içerisinde belirtilen parametrelere
        //metodun imzası (method signature) denir.
public static void Main(string[] args)
    {
        Console.WriteLine("Merhaba!");
        Console.ReadKey();//kullanıcının bastığı tuşu okur.
    }
}
}
```

## VERİ TİPLERİ

C# dilinde her sınıf bir veri tipidir, her veri tipi bir sınıftır.

Klâsik dillerde *tamsayılar*, *kesirli sayılar*, *karakterler*, *boolean değerler* gibi ilkel veri tipleri dile gömülü öğelerdir; herbiri bir anahtar sözcükle belirtilir. Oysa Nesne Yönelimli Programlamada her sınıf soyut bir veri yapısıdır. O nedenle, C# dilinde klâsik dillerdeki gibi ilkel veri tipleri yoktur. Her veri tipi bir sınıftır. Ancak, programcının işini kolaylaştırmak için, .NET bu sınıfları ve sınıf öğelerini hazır tanımlamıştır. .NET Framework ortamını anlatırken C++, C#, VB, J# dillerinin temel veri tiplerinin .NET'in CTS (Common Type System) denilen veri tiplerine dönüştürülür ve böylece bu diller için ortak bir platform yaratılmış olur. C# dilinin temel veri tiplerinin .NET 'deki karşılıkları *Veri Tipleri ve Değişkenler* bölümünde ayrıntılı olarak ele alınacaktır. Bu sınıflar sanki klâsik dillerdeki ilkel veri tipleriymiş gibi kullanılabilirler. Aşağıda ayrıntıları verilecek olan *temel veri tiplerinin (built-in-types)* birer sınıf olduğunu, dolayısıyla klâsik dillerdeki ilkel veri tiplerine göre çok daha işlevsel olduklarını daima anımsamalıyız.

Bir programda farklı veri tipleriyle işlem yapmamız gerekebilir. Örneğin, tamsayılar, kesirli sayılar, karakterler (harfler ve klavyedeki diğer simgeler), metinler (string), mantıksal (boolean) değerler (dogru=true, yanlış=false) ilk aklımıza gelen farklı veri tipleridir. Bu farklı veri tiplerinin büyüklükleri (bellekte kaplayacakları yer) ve onlarla yapılabilecek işlemler birbirlerinden farklıdır. Örneğin, sayılarla dört işlem yapabiliriz, ama metinlerle yapamayız. O nedenle, C# ve başka bazı diller verileri tiplere ayırır. Değişken tanımlarken onun hangi tip veriyi tutacağını belirtir. Böylece, **ana bellekte (RAM)** o değişkene yetecek bir yer ayırır ve o veri tipine uygun işlemlerin yapılmasına izin verir.

### Sayısal Veri Tipleri

Birçok işlemi tamsayılarla ve kesirli sayılarla yaparız. O nedenle, sayılar her programlama dilinde önemlidirler. Farklı dillerde yazılan derleyiciler sayıları alt sınıflarına ayırır. En basit derleyiciler bile, sayıları tamsayılar (integer) ve kesirli sayılar (floating numbers) diye ikiye ayırır. Bazan küçük sayılarla, Bazan da büyük sayılarla uğraşırız. Dolayısıyla, belleği etkin kullanabilmek için, birçok dilde, sayılar, alt-gruplarına ayrılır. C# dili, bu yönde çok ayrıntılı bir gruplama yapmıştır.

### Tamsayı Veri Tipi

C# dili tamsayıları bellekte kendilerine ayrılan büyüklüğe ve işaretli olup olmadıklarına göre gruplara ayırır.

Değişken tipinin başında s harfi varsa bu, o değişkenin işaretli (signed) olduğunu gösterir.

Değişken tipinin başında u harfi varsa bu, o değişkenin işaretli (unsigned) olduğunu gösterir.

Bazı değişken tiplerinin başında ne s ne de u harfi vardır. Bu durumda o değişkenin kullanılan programlama dilindeki varsayılan durumu geçerlidir. Örneğin byte tipinde bir değişken C# programlama dilinde varsayılan olarak işaretlidir ve başına u harfi getirilmez. Short tipindeki bir değişken ise varsayılan olarak işaretlidir ve işaretli olarak kullanılmak istenirse ushort olarak tanımlanmalıdır.

Sbyte → signed byte demektir.

uint → unsigned integer → işaretli tamsayı

int → -2.147.483.648 – 2.147.483.647 ( $2^{32}$  farklı değer alır. Bellekte 32 bit yer kaplar.)

uint → 4.294.967.295+1=4.294.967.296 ( $2^{32}$  farklı değer alır. Bellekte 32 bit yer kaplar.)

sbyte → -128 - +127 ( $2^8 = 256$  farklı değer alır. Bellekte 8 bit yer kaplar.)

byte → 0 – 255 ( $2^8 = 256$  farklı değer alır. Bellekte 8 bit yer kaplar.)

### *Kesirli Sayı Veri Tipi*

#### **Float;**

bellekte 4 byte yer kaplar ve basamak hassasiyeti 7 dir.

float sayı=11.2f;

şeklinde tanımlanır.

#### **Double;**

bellekte 8 byte yer kaplar. Basamak hassasiyeti 16 dır.

double sayı=11.2d; veya double sayı=11.2;

şeklinde tanımlanır.

C# ta ondalıklı bir sayı yazınca varsayılan olarak o sayı double tipindedir.

#### **Decimal;**

bellekte 16 byte yer kaplar hem tamsayı hemde ondalıklı sayı barındırabilen bir tipidir. Basamak hassasiyeti 29 karakterdir.

decimal sayı=11.2m;

şeklinde tanımlanır.

Aralarındaki farka gelince float bellekte az yer kapladığı için performans açısından en uygun değer gibi gözüksede 7 basamağa kadar değer alabildiği için kullanım amacı daha küçük sayılarla işlem yapmak içindir.

Örneğin

"float sayı=12.456781f;"

gibi bir tanımlamanın ekran çıktısı 12.45678 olacaktır,

son basamak 8. Basamağa denk geldiği için ve 5 ten küçük olduğu için dikkate alınmamıştır.

Eğer 5 ten büyük olsaydı son sayı bir üst sayıya yuvarlanarak 12.45679 olacaktı.

double tipinde tanımlanmış bir değişkenin float tipinde tanımlanmış bir değişkenden farkı bellekte kapladığı alan 8 byte ve basamak sayısı 16 dır, diğer özellikleri floatla aynıdır ve C# ondalık sayılar için aksi belirtilmedikçe varsayılan değer olarak double tipini kabul eder.

decimal tipi ise bellekte 16 byte gibi büyük bir yer kaplar, ondalık sayıları 29 basamağa kadar hassiyetle kullanabildiği için ve tamsayı tipleride içinde barındırabildiği için genellikle finansal hesaplamalarda kullanılır.

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	$\pm 1.5 \cdot 10^{-45}$ , ..., $\pm 3.4 \cdot 10^{38}$ (reel sayı)	float a=5F; veya float a=5f;
double	8 bayt	$\pm 5.0 \cdot 10^{-324}$ , ..., $\pm 1.7 \cdot 10^{308}$ (reel sayı)	double a=5; veya double a=5d; veya double a=5D;
decimal	16 bayt	$\pm 1.5 \cdot 10^{-28}$ , ..., $\pm 7.9 \cdot 10^{28}$ (reel sayı)	decimal a=5M; veya decimal a=5m;

### Sözel Veri Tipleri

Programda kullanılan değerleri sayısal olarak değil de karakter veya karakter katarı olarak tutmak için kullanılan veri tipleridir.

### Char Veri Tipi

Character sözcüğü, kaynak programda kullanılan bütün harf, rakam ve diğer simgelerin kümesidir. C# dili, karakterleri unicode diye adlandırılan ve uzunluğu 2 byte (16 bit) olan sistemle belirler. Bu nedenle 1 byte (8 bit) lık sistemlerde olduğu gibi yalnızca İngiliz alfabesini değil, Türkçe, Arapça, Çince, Japonca, Rusça, İbranice vb. bütün dillerin alfabelerindeki harfleri içine alır.

Burada şuna dikkat çekmeliyiz. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 rakamların her birisi sayısal değerlerinin dışında, ayrıca birer character olarak character veri tipinde yer alırlar. Zaten bütün sayıların ekrana ya da kagıda gönderilen görüntüleri 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 karakterleriyle temsil edilirler.

char değerler daima tek tırnak ( ' ' ) içinde yazılır. Örneğin,

'b', '0', '7', '\$', '=', '?', ''

birer karakterdir, ama derleyici

b, 0, 7, \$, =, ?, ' ,

simgelerini birer karakter olarak algılamaz.

Bütün dillerde olduğu gibi, C# dili de klavyede tek bir simgeyle temsil edilemeyen bazı karakterleri kullanır. Bunlar, özellikle, yazıcıya ya da ekrana çıktı alırken kullanılan karakterlerdir. Bazı harflerin önüne ( \ ) konularak yazılırlar, ama derleyici onları birer karakter olarak algılar:

\n New Line (yeni satıra geç)

\t Tab (tab yaz)

\0 Null (boş)

\r Carriage Return (satırbaşına git)

\\ Backslash (ters bölü çizgisi ( \ ) yazar)

\b Backspace (geri silme)

**String Veri Tipi**

string veri tipi, String sınıfının takma adıdır. Tek bir karakter yerine bir sözcük, bir tümce, bir paragraf ya da bir roman yazmak istedigimizde char veri tipi yeterli olmayacaktır. Modern dillerin çoğunda olduğu gibi, C# dili de metinleri içerecek bir veri tipi yaratmıştır. Bu tipe string denilir. string veri tipi (sınıfı), uzun ya da kısa her metni içerebilir. Tabii, burada metin derken, yalnızca alfabenin harf, rakam ve işaretleriyle yetinmiyor, char tiplerinden oluşabilecek her diziyi kastediyoruz.

String tipleri çift tırnak ( " ") içine yazılırlar. Char tipler tek tırnak ( ' ' ) içine yazıldığı için, C# derleyicisi 'b'

verisini char olarak algılar, ama "b"

verisini string olarak algılar.

C# dilinde string tipli bir değişken bildirimi aşağıdakilere benzer olarak yapılır:

string birMetin;

ya da bildirim anında ilk değeri verilmek istenirse

string birMetin = "Merhaba, dünya!";

string değerler daima çift tırnak ( " ") içinde yazılır. Örneğin,

"b", "0", "Ankara", "&\*\$\*/?", "3 + 2", "123", "-123"

birer string'dir. Ama derleyici

b, 0, Ankara, &\*\$\*/?, 3 + 2, 123, -123

simgelerini birer string olarak algılamaz.

**Uyarı:**

C# dilinde string değeri birden çok satıra yazılamaz. Metin satıra sığmadığı ya da satırlara bölmek gerektiğinde, yeni satıra geçmeyi sağlayan (\n) karakteri kullanılır. Örneğin,

string birSiir = "Neler yapmadık bu vatan için \n Kimimiz öldük, \n Kimimiz nutuk söyledik. \n O.V.Kanık";

Bunu ekrana yazdırmak için

System.Console.WriteLine (birSiir);

metodunu kullanırsak, ekrana şu görüntü gelir:

Neler yapmadık bu vatan için

Kimimiz öldük,

Kimimiz nutuk söyledik.

O.V.Kanık

**Boolean Veri Tipi**

Matematiksel Mantığın kurucusu sayılan George Boole'un adına izafe edilen ve Boolean diye adlandırılan mantıksal veri tipi iki tanedir: true (dogru), false (yanlış). Bütün programlama dillerinde boolean veri tipi çok önem taşır. Özellikle, program akışının yönlendirilmesinde ve döngü yapılarında olmazsa olmaz tiplerdir. bool anahtar sözcüğü System.Boolean'ın takma adıdır.

C# Tipi	Uzunluk (Byte)	Kapsam
bool	1 byte	true , false

Bool tipi değişken bildirimi, genel değişken bildirimi kuralına göre yapılır:

```
bool değişken_adı;
```

ya da ilk değer atanmak istenirse

```
bool değişken_adı =değişken_degeri;
```

yazılır. Bazı durumlarda true ya da false literal değerleri yerine, bir mantıksal (boolean) deyim de atanabilir. Örneğin,

```
bool mezun;
bool mezun = true;
bool emekli = false;
bool bFormül = (x > 15 && x < 30);
```

**Örnek:**

```
using System;
```

```
namespace mantıksal
```

```
{
    using System; class test
    {
        public static void Main()
        {
            bool a = true;
            Console.WriteLine(a ? "evet" : "hayır");
        }
    }
}
```

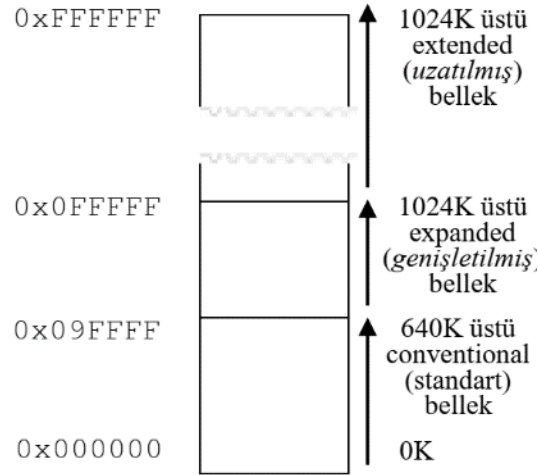
**Tarih ve Saat Veri Tipi**

Tarih ve saat bilgilerini tutar.

Tür	Boyut	Açıklama	Örnek
DateTime	8byte	Tarih ve Zaman Tutar	Datetime zaman = Datetime.now;

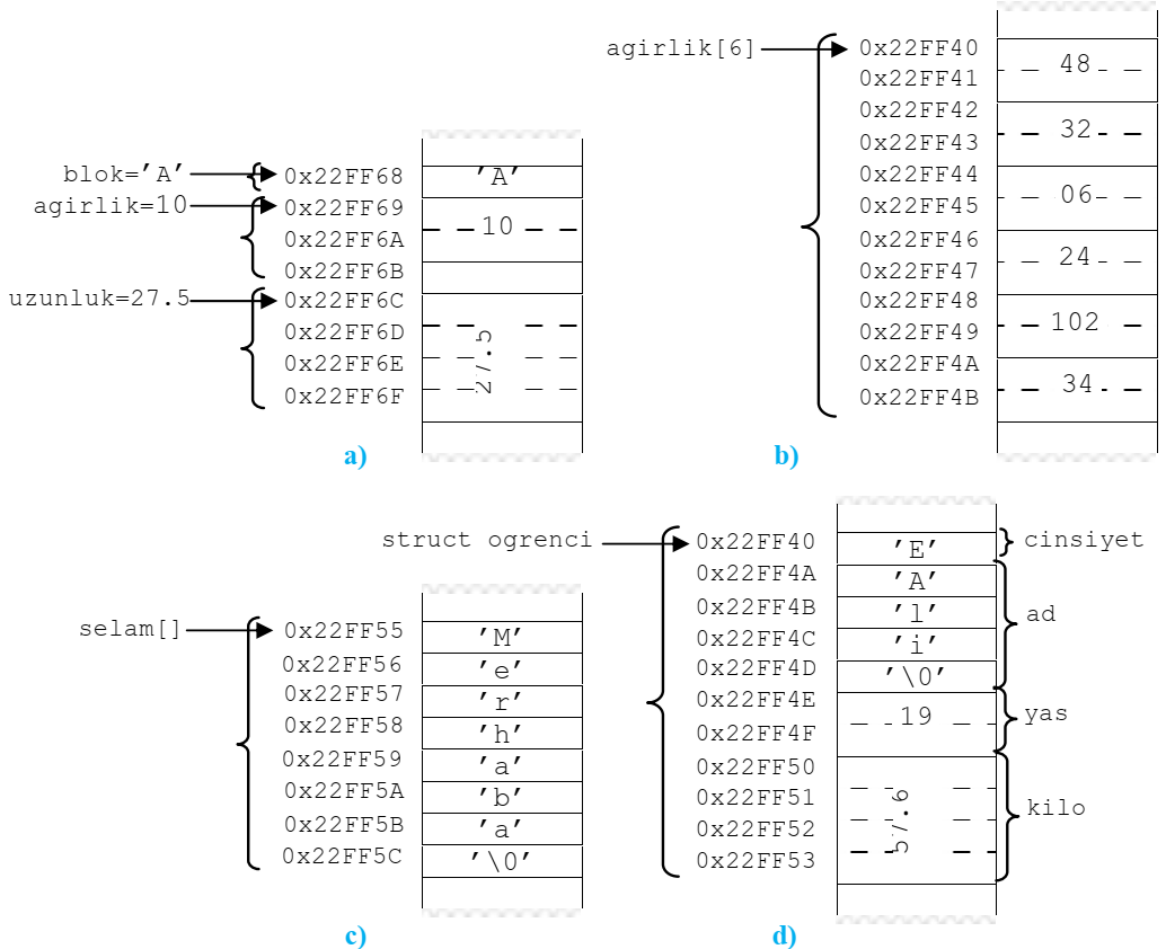
## BELLEK YAPISI

Yapısal olarak bellek, büyüklüğüne bağlı olarak binlerce, milyonlarca 1'er Byte (8 Bit)'lik veriler saklayabilecek biçimde tasarlanmış bir elektronik devredir. Bellek, aşağıdaki şekildeki gibi her byte'ı bir hücre ile gösterilebilecek büyükçe bir tablo olarak çizilebilir.



Belleğin adreslenmesi ve bellek haritası

Her bir hücreyi birbirinden ayırmak için hücreler numaralarla adreslenir. Program, işlemci ve işletim sistemi bu adresleri kullanarak verilere erişir (yazar/okur). Büyük olan bu adres numaraları 0'dan başlayarak bellek boyutu kadar sürer ve 16'lık (Hexadecimal) biçimde ifade edilir. İşletim sistemleri belleğin bazı bölümlerini kendi dosya ve programlarını çalıştırmak için, bazı bölümlerini de donanımsal gereksinimler için ayırır ve kullanır. Ancak belleğin büyük bölümü uygulama programlarının kullanımına ayrılmıştır. Aşağıdaki şekilde DOS işletim sistemi için bellek haritası görülmektedir.



Veri yapılarının bellek üzerindeki yerleşimleri

Temel/İlkel (Primitive) veri yapılarından birisinin tipi ile tanımlanan bir değişken, tanımlanan tipin özelliklerine göre bellekte yerleştirilir. Örneğin;

```
char blok = 'A';
```

```
int agirlik = 10;
```

```
float uzunluk = 27.5;
```

değişken tanımlamaları sonunda bellekte Şekil a)'daki gibi bir yerleşim gerçekleşir. İşletim sistemi değişkenleri bellekteki boş alanlara veri tiplerinin özelliklerine uygun alanlarda yerleştirir.

Basit (Simple) veri yapıları temel veri yapıları ile oluşturulur. Örneğin;

```
int agirlik [6];
```

tanımlamasındaki agirlik dizisi 6 adet int (tamsayı) veri içeren bir veri yapısıdır. Bellekte Şekil b)'deki gibi bir yerleşim gerçekleşir. İşletim sistemi dizinin her verisini (elemanını) ardı ardına, bellekteki boş alanlara veri tipinin özelliklerine uygun alanlarda yerleştirir. Örneğin;

```
char selam [] = "Merhaba";
```

veya

```
char selam [] = {'M','e','r','h','a','b','a','\0'};
```

tanımlamasındaki selam dizisi 8 adet char (karakter) tipinde veri içeren bir string veri yapısıdır. Bellekte Şekil c)'deki gibi bir yerleşim gerçekleşir. İşletim sistemi stringin her verisini (elemanını) ardı ardına, bellekteki boş alanlara veri tipinin özelliklerine uygun alanlarda yerleştirir.

Örneğin;

```
struct kayit{
```

```
    char cinsiyet;
```

```
    char ad[];
```

```
    int yas;
```

```
    float kilo;
```

```
}ogrenci;
```

tanımlamasında kayit adında bir structure (yapı) oluşturulmuştur. Bu veri yapısı dikkat edilirse farklı temel veri yapılarından oluşan, birden çok değişken tanımlaması (üye) içermektedir. ogrenci, kayit yapısından bir değişkendir ve ogrenci değişkeni üyelerine aşağıdaki veri atamalarını yaptıktan sonra, bellekte Şekil d)'deki gibi bir yerleşim gerçekleşir.

```
ogrenci.cinsiyet = 'E';
```

```
ogrenci.ad[] = "Ali";
```

```
ogrenci.yas = 19;
```

```
ogrenci.kilo = 57.6;
```

İşletim sistemi kayit veri yapısına sahip ogrenci değişkeninin her bir üyesini ardı ardına ve bir bütün olarak bellekteki boş alanlara üyelerin veri tiplerinin özelliklerine uygun alanlarda yerleştirir.

**Birleşik (Compound) veri yapıları** basit veri yapılarından **dizi** veya **structure** tanımlamaları **ile** oluşturulabileceği gibi, nesne yönelimli programlamanın veri yapılarından **class** (sınıf) tanımlaması **ile** de oluşturulabilir.

## Bellek (Memory) Alanları

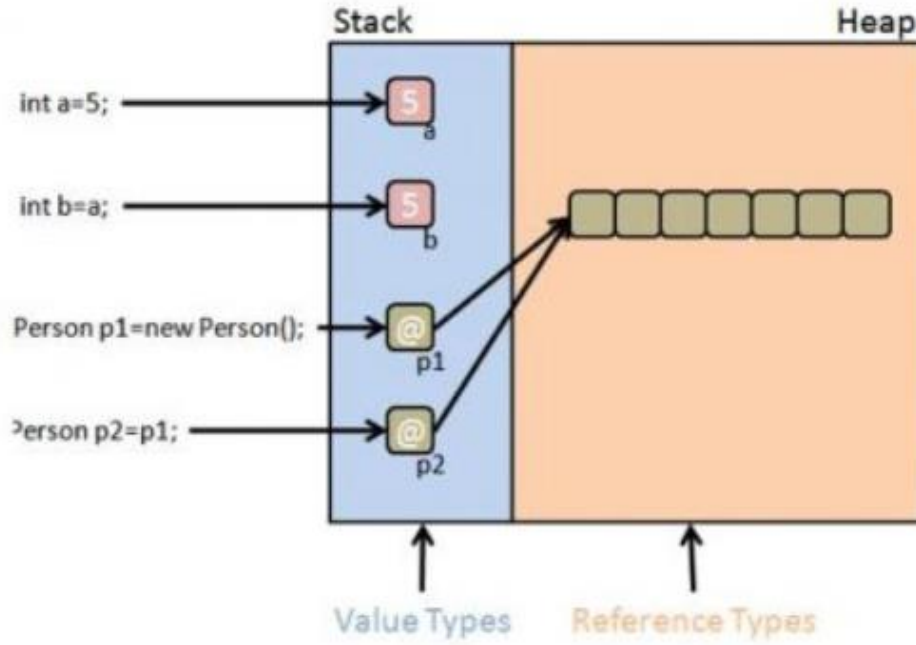
### Stack (Yığın)

- RAM'de yer alır.
- İlkel (primitif) veri tipleri burada tutulur. (byte, short, int, long, double, char, ..)
- Hızlı ulaşılır.
- Tutulan değer Stack kapsamından çıkınca otomatik silinir.
- Stackoverflow olabilir. ☺
- Derleme anında oluşturulur.
- Programcı kullanacağı bellek alanının boyutunu tam olarak biliyorsa stack kullanmak daha mantıklıdır.
- Boyutu belli olmayan verileri tutmaz. Onların adreslerini tutar. O veriler ise belleğin Heap bölgesindedir.



**Heap (Küme)**

- RAM’de yer alır.
- Gelişmiş veri tipleri burada tutulur. (string, metotlar, nesneler, eleman sayısı belirsiz her türlü dizi, { } içerisinde oluşturulan her türlü şey, vs. )
- Erişim stack’e göre daha yavaş.
- Güncel programlama dillerinde (C#, Java) tutulan değerlerin stack ile ilişkisi kesilince otomatik silinir. (GarbageCollector). Eskiden programcı silmek zorundaydı.
- Çalışma anında (run-time) oluşturulur.
- Programcı kullanacağı bellek alanının boyutunu tam olarak bilmiyorsa heap kullanmak daha mantıklıdır.
- Boyutu belli olmayan verileri tutar. Stack onların adreslerini tutar.

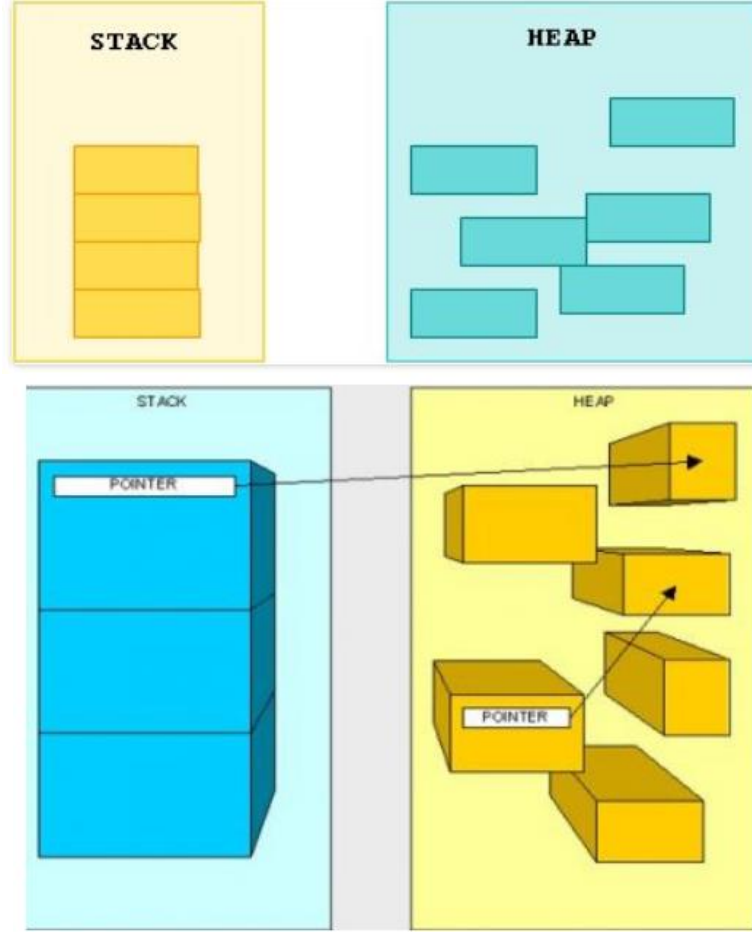
**STACK ve HEAP İLİŞKİSİ**

```
int i=13;
byte b=23;
string mesaj;

mesaj="Selam";

Ekrana_Yaz(mesaj);
```

STACK			HEAP		
i	13	00 H	mesaj	S	A0 H
		01 H		e	A1 H
		02 H		l	A2 H
		03 H		a	A3 H
b	23	04 H		m	A4 H
mesaj	A0H	05 H		\0	A5 H
		06 H			A6 H
		07 H			A7 H
		08 H			A8 H
		09 H			A9 H



### STATIC Bellek Alanı

- İçinde bulunduğu sınıftan nesne oluşturulmadan veya hiç bir nesneye referans olmadan kullanılabilen sınıf üyeleri **static** olarak nitelendirilir. Metotlar ve alanlar (fields) **static** olarak tanımlanabilir.
- **Static** veya **const** olarak oluşturulan değişkenler saklanır.
- **Const** olarak tanımlanan bir değişken aslında sabittir. Bu değer daha sonra değiştirilemez ve aynı sınıftaki tüm nesneler tarafından aynı değer ile kullanılır.
- **Static** olarak tanımlanan bir metod veya değişken ise tanımlandığı sınıfa ait olur ve bu sınıf dışından da çağırılabilir. Ancak tanımlandığı sınıftan türetilen tüm nesneler için içeriği aynı olur. Nesnelerden biri aracılığı ile **static** elemanın içeriği değiştirilirse tüm nesneler için de değiştirilmiş olur.

### Bellek İhtiyacı

Programın bellek gereksinimi, programın veya fonksiyonun yürütülmesi sırasında gerekli bellek miktarıdır; üç kısımdan oluşur. Biri, program kodu, ikincisi programın üzerinde çalıştığı veri ve üçüncüsü de fonksiyon çağırımlarda kullanılan yığın (stack) alanıdır. Dolayısıyla bir programın bellek gereksinimi hesaplanırken her üçü ayrı ayrı göz önüne alınır:

- Program Kodu
- Veri
- Yığın

Eğer, program içerisinde rekürsif özelliği olan fonksiyon yok ise üçüncü kısım olan yığın alanı ihmal edilebilir. Çünkü fonksiyon çağırımlarında parametre aktarımı, geri dönüş adresinin tutulması gibi bilgiler yığında tutulurlar. Dolayısıyla çok fazla iç içe fonksiyon çağırımları yığın bellek alanını ihmal edilemeyecek kadar şişirebilir.



Program kodu, Veri ve Yığın aynı anda bilgisayar belleğini kullanırlar.

Eğer program, proseslere ayrılmamışsa, yani hepsi bir bütünsü program kodu, yaklaşık olarak, doğrudan program kodunun saklandığı dosya büyüklüğündedir. Ancak, program, prosesler şeklinde tasarlanmışsa, bu durumda program kodu, bellekte, aynı anda çalışması gereken proseslerin işgal ettiği alan büyüklüğündedir. Veri ise, program içerisinde kullanılan değişken, dizi veya veri yapılarının toplamıdır; bunlar için gerekli bellek alanı **veri için gerekli bellek** diye adlandırılır.

### Program kodunun bellek ihtiyacı

Program kodu için bellek gereksinimi, programın tasarımına, derleyici türüne, program içerisinde işletim sistemi çağrılarını kullanılıp kullanılmamasına, fonksiyonların rekürsüf veya iterasyonlu olmasına gibi birçok tasarım yaklaşımına göre farklılık gösterir. Bu, program tasarımcısının bilgi, görgü ve deneyimine bağlı olarak değişir.

Program kodunun bellekteki kapladığı alan programın tasarım şekline ve diğer parametrelere bağlı olarak değişir.

### Veri için bellek ihtiyacı

Veri için bellek gereksinimi, doğrudan program içerisinde kullanılan değişken/sabit sayısı ve türüne, veri yapısına bağlıdır. Bellek gereksinimi kendi içerisinde biri statik diğeri dinamik bellek olmak üzere iki sınıfa ayrılır. Statik bellek gereksinimi program içerisindeki global değişken ve veri yapılarından hesaplanabilir; dinamik bellek gereksinimi ise programın yürütülmesi anında o anki gereksinime göre belirlenir. Dolayısıyla programın çalışması sırasındaki parametrelere bağlıdır. Ancak, örnek olaylar için, en kötü durumda gerekli bellek gereksinimi hesaplanabilir. C dili gibi fonksiyon içerisinde yerel değişkenler bildirimi yapılan programlama dillerinde, yerel değişkenlerden en fazla yer işgal edenin bellek gereksinimi de göz önüne alınmalıdır.

İSİM	SOYAD	ŞEHİR
Ahmet	Öztürk	Ankara
Mehmet	Yılmaz	İstanbul
Ayşe	Sönmez	Mersin
İlhan	Çiftçi	Rize
Fatma	Ünal	Konya
Sinan	Kudoğlu	Gaziantep
Ali	Kılıç	Adana
Veli	Cevahir	Edirne
Bülent	Faruk	Bolu
Cem	Özcan	Van

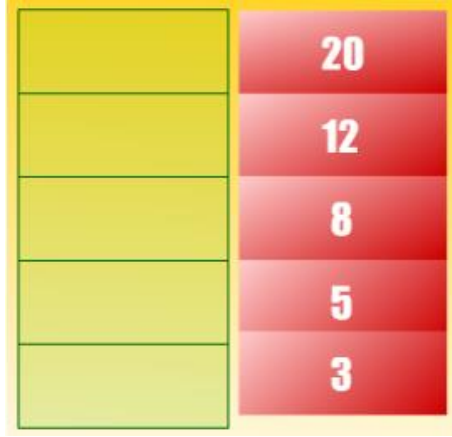
Veri için bellek gereksinimi, doğrudan program içerisinde kullanılan değişken/sabit sayısı ve türüne, veri yapısına bağlıdır.

**Yığın Bellek İhtiyacı**

Hemen hemen her sistemde yığın olarak adlandırılan bellek alanı bulunur; burası son giren ilk çıkar (LIFO) davranışında olup bilgilerin geçici olarak saklandığı bellek alanıdır. Burası işletim sistemde koşan her program tarafından bilinir; dolayısıyla programlar veya fonksiyonlar bu alan üzerinden birbirlerine parametre aktarabilirler.

Yığında genel olarak, fonksiyonlara parametre aktarılmasında, geri dönüş adreslerinin saklanması ve fonksiyonların dönüş değerlerinin aktarılmasında kullanılır.

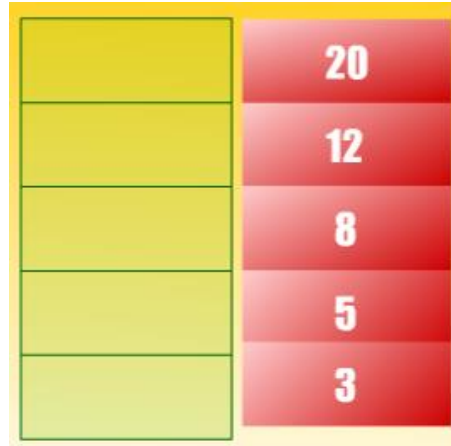
Kesme hizmet programları da yığını kullanılır. Sisteme bir kesme isteği geldiğinde, işlemci o anda yaptığı işi bırakır ve o andaki durum bilgilerini (saklayıcı içeriklerini, dönüş adresini vs.) yığına atar. Kesme hizmet programından döndüğünde yığından eski bilgilerini alır ve kaldığı yerden devam eder.



Boş Yığın



Yığın dolduruldu



Yığın boşaltıldı. Son giren ilk çıkar (last in firstout).

**Bellek Miktarı Hesabı**

**Örnek:** Aşağıda 20 elemanlı bir dizinin eleman değerlerinin aritmetik ortalamasını hesaplayan bir C programı verilmiştir; bu programın yürütülmesi sırasında gerekli bellek miktarını hesaplayınız.

```
main()
{
    int A[20]={7, 65, 96, 17, 15, 2, 5, 43, 33, 99,
              112, 37, 64, 87, 3, 5, 8, 21, 27, 1};

    int k, toplam=0;
    float ortalama;
    for(k=0; k<20; k++)
        toplam+=A[k];

    ortalama= toplam/20.0;

    printf("Ortalama Değer=%f\n", ortalama);
}
```

**Çözüm:** Bu programda veri olarak adları A, k, toplam ve ortalama olan dört tane değişken kullanılmaktadır. Her şeyden önce veri için gerekli bellek alanı bunların işgal ettiği toplam bellek alanından hesaplanır. Dolayısıyla ilgili değişkenlerin bellekte ne kadar yer işgal ettikleri bilgisi gereklidir. Tamsayı (int) ve tek duyarlı gerçel (float) sayıların 4 Byte yer işgal ettiği varsayılırsa, veri için bellek alanı,  $4*20+4*1+4*1+4*1=92$  Byte olarak hesaplanır.

Programda rekürsif yapıda fonksiyon kullanılmadığı için yığın pek fazla kullanılmamaktadır; yalnızca printf() fonksiyonu bir kez çağırılmıştır. Dolayısıyla yığın kullanımı ihmal edilebilir.

Programın işgal ettiği bellek ise, programın derlenip makine kodunun (.EXE) elde edilmesiyle öğrenilebilir veya programın birleştirici dildeki karşılığının belirlenmesi gerekir.

**SABİTLER ve DEĞİŞKENLER (Ders kitabı – Yaşar, 2011, ss:32 - 55)****Sabitler (Constants)**

Sabitler veri tutan sınıf öğeleridir. Dolayısıyla her sabite, ana bellekte tutacağı veri tipine yetecek kadar bir yer ayrılır. Program çalışırken sabitlere atanan değerler değiştirilemez. Ayrıca, sabit bildirimi yapıldığı anda değeri atanmalıdır. Örneğin,

```
const int    yıllıkFaizOranı = 8;
const float  amortismanPayı = 12.50;
```

Sabit kullanmadan, sabitin işleme girdiği her yerde onun değerini koyarak program yazmak mümkündür. Örneğin, yukarıda bildirimi yapılan yıllıkFaizOranı bir bankanın mudilerinin alacağı faizleri hesaplayan programda kullanılıyor olsun. Yıllık faizin hesaplandığı her işlemde yıllıkFaizOranı yerine 8 kullanılabilir ve program aynı sonucu verir. Ancak faiz oranını değiştirip 9 yapmak istediğinizde, kaynak programın bütününde 8 yerine 9 yazmak gerekecektir. Bu hem uzun zaman alıcı hem de bazıları unutulabileceği için, hataya açıktır. Onun yerine yıllıkFaizOranı adlı bir sabit kullanılırsa, kaynak programda onun değerini 9 ile değiştirmek, bütün programı güncellemeye denktir. Bunu yapmak büyük bir zaman kaybını önleyeceği gibi, programda hata oluşmasının da önüne geçmiş olacaktır.

### Sabit Tanımlama

Sabitlerin değeri değiştirilemez!

Sabit tanımlama söz dizimi(syntax):

Tanımlayıcı+Tip+Sabit\_Adi=="Aldığı Değer"

### Değer Atama

const String s="Kırkağaç MYO"; //#\$@}{?+\*/ 1s 123 boşluk \_ avagadro\_sabiti -> string içine tüm karakter yazılabilir!

const double pi\_sayisi=3.14159262;

Örnek(yaşar, sf:38):

---

```

using System;
namespace sabitler
{
class Program
{
static void Main(string[] args)
{
const String a = "Kırkağaç MYO"; //a adında değeri Kırkağaç MYO olan string veri tipinde
bir sabit tanımlanıyor.
const string A="Celal Bayar Üniversitesi";
const double pi_sayisi = 3.14;//pi_sayisi adında değeri 3.14 olan ondalıklı veri tipinde
bir sabit tanımlanıyor.
const char tek_karakter='E';//tek_karakter adında char tipinde, değer E karakteri olan
sabit tanımlanıyor.
Console.WriteLine(a);//a sabitinin değeri ekrana yazdırılıyor ve imleç bir alt satıra
geçiyor.
Console.WriteLine(pi_sayisi);//pi_sayisi sabitinin değeri ekrana yazdırılıyor ve imleç
bir alt satıra geçiyor.
Console.ReadLine();//satırdaki verileri Klavyeden Enter tuşuna basılana kadar okuyor.
}
}
}

```

---

### Değişkenler

Verileri belleğe almak ve programda işlemek için değişkenler kullanılır. Değerleri program çalışırken değişebilir. Sabitlerden farkı budur.

Değişken tanımlama söz dizimi(syntax):

Değişken\_Tipi Değişken\_Adi;

Programlama diline özgü kelimeler de değişken adı olamaz! Örneğin int, string, char, double, if, for, while gibi kelimeler değişken veya sabit ismi olamaz!

/\*-+?@%& karakterleri değişken adında olmaz.

int a,b,c,d;//tek bir değişken tipi ile aralarına virgül koyarak birden fazla değişken tanımlanabilir.

musteri\_adi;//değişken isimlerinde boşluk karakteri yerine \_ karakteri kullanılabilir.

int i=82;//değişkenler tanımlanırken başlangıç değeri atanabilir.

int i=75,x,k=12,m,n=36;//

Kaynak programlarımızda C# veri tipleri yerine .NET 'teki karşılığını kullanabiliriz. Örneğin,

```
short n ;
int m ;
long r ;
bildirimleri yerine sırasıyla,
```

```
Int16 n;
Int32 m ;
Int64 r ;
```

bildirimlerini yapabiliriz. C# derleyicisi bunları denk deyimler olarak algılar.

### *Değişkenlere değer atama*

```
using System;
namespace gc1Bor042{
class program{
static void Main(string [] args){
int a,b=9;
double c=2.5,d;
    a=15;
    d=c*3;//d=7,5
    System.Console.WriteLine(d);//7,5
    a=b+9;//a=18
Console.WriteLine(a);//18
Console.ReadLine();
    }
}
```

**Grup Uyumu:** değişkenler aynı gruptan sabitlere ve değişkenlere doğrudan atanabilirler. Ondalıklı sayılar tamsayıları kapsadığı için farklı gruplar arası atamalarda, tamsayılar ondalıklı değişkenlere atanabilirler. Ancak ondalıklı sayılar tamsayı değişkenlere atanamazlar.

**Boyut Uyumu:** grup uyumu sağlanıyorsa kapasite aralığı kapsanan değişken diğer kapsayan değişkene atanabilir.

```
int a=3;
byte b=5;
a=b; //geçerli.
B=a; //geçersiz. Çünkü a değişkeni daha büyük bir değişkendir.
int x=1250;
double y=3.14;
y=x; //geçerli. Çünkü y değişkeni ondalıklı sayıdır ve ondalıklı sayılar tam sayıları kapsar.
X=y; //geçersiz. Çünkü grup uyumu yok.
```

En büyük ve En küçük değerlerin kodla bulunması:

```
//Yöntem 1:
using System;
namespace gzlAor043
{
class program
{
static void Main(string[] args)
{
byte enkucukbyte, enbykbyte;
enkucukbyte = Byte.MinValue;
enbykbyte = Byte.MaxValue;
System.Console.WriteLine("Byte veri tipinin en küçük
değeri="+enkucukbyte); //0
System.Console.WriteLine("Byte veri tipinin en büyük değeri="+enbykbyte);
Console.ReadLine();
}
}}
```

Değişkenlerin en büyük ve en küçük değerlerinin kodla bulunması: (Yaşar, sf:43)

```
//Yöntem 2:
using System;
namespace gclBor043b
{
class program
{
static void Main(string[] args)
{
/*byte enkucukbyte, enbykbyte;
enkucukbyte = Byte.MinValue;
enbykbyte = Byte.MaxValue;*/
System.Console.WriteLine("En Küçük Byte Veri Tipi Değeri="+Byte.MinValue);
System.Console.WriteLine("En Büyük Byte Veri Tipi Değeri=" + Byte.MaxValue);
Console.ReadLine();
}
}
}
```

### Değişken İsimlendirme Kuralları

Yaşar, sf: 39

/\*-+?@%&

### Sözel Değişkenler

String → birden fazla karakter.

Char → \u0000 - \uFFFF aralığında 16 bit uzunluğunda tek karakter.

String s="KMYO"; //string çift tırnak karakteri içinde!

Char cinsiyet='E'; //char tek tırnak karakteri içinde!



**Örnek (Yaşar, sf:45)**

```

using System;
namespace ornek045
{
    class Program
    {
        public static void Main(string[] args)
        {
            string s, adi;
            char k;
            s = "Emin";
            adi = s;//adi="Emin"
            k = 'a';
            Console.WriteLine(s);//Emin
            Console.WriteLine(k);//a
            Console.ReadKey();
        }
    }
}

```

**Not:** String ve char tipleri birbirlerine doğrudan atanamazlar.

**Not:** Stringler ile matematiksel işlem yapılamaz, ama iki string toplanabilir. Toplamak için + işareti kullanılır. Toplama işleminin değişme özelliği yoktur.

**Mantıksal Değişkenler**

Bool → True veya False değerini alır.

Örnek 046:

```

using System;
namespace or046 //Calisma
{
    class Program
    {
        public static void Main(string[] args)
        {
            bool cinsiyet;
            cinsiyet = true;
            if(cinsiyet==true)
                Console.WriteLine("Erkek");
            else
                Console.WriteLine("Bayan");
            Console.ReadKey(true);//
        }
    }
}

```

**Değişken Kapsama Alanı**

İki küme parantezi arasına **{...Kapsam...(Scope)...}** kapsam (scope) adı verilir.

Değişken kapsama alanı dışındaysa hiç tanımlanmamış gibi hata verir!

C# için:

Cannot use local variable 'not' before it is declared! (yerel '**not**' değişkeni tanımlanmadan önce kullanılamaz!)

The name 'not' does not exist in the current context! ('**not**' diye bir değişken geçerli içerikte yok!)

**Kapsamlı/Genel/Küresel (Global) Değişkenler**

Tüm alt programların dışında en üst seviyede "**static**" sözcüğü ile tanımlanırlar ve tüm programda her yerde geçerli olurlar.

Söz dizimi → **static** değişken\_tipi değişken\_adı;

Örnek 048:

```
using System;
namespace or048{
    class Program {
        static int numara;//genel değişken tanımlanıyor!
        static void degistir(){ //degistir adında bir alt program(işlev) tanımlanıyor!
            numara=5;
        }
        public static void Main(string[] args)
        {
            numara=7;//numara değişkeni ana programda da kullanılıyor!
            Console.WriteLine(numara);//numara değişkeninin değeri (7) ekrana yazdırılıyor!
            degistir();//degistir isimli fonksiyon ana programa çağırılıyor! Numara=5
            Console.Write(numara);//numara değişkeninin değeri (5) ekrana yazdırılıyor!
            Console.ReadKey(true);//bir tuşa basana kadar bekliyor!
        }
    }
}
```

**Kısıtlı/Yerel (Local) Değişkenler**

Sadece tanımlı oldukları alt program içinde geçerlidirler. Belleği daha etkin kullanmak için tercih edilirler.

**Not:** İki küme parantezi ( { } ) arası **blok(scope -> kapsam)** olarak değerlendirilir.

**Kural 1:** Tanımlanmadan önce kullanılamazlar. Bir blok içerisinde tanımlanan değişkenler diğer bloktan çağrılmaz, ama aynı blok içerisinde çağrılabilir.

**Kural 2:** Bir alt programda, programlamanın yapı taşlarına ait iç içe bloklar olabilir. Bir blok içerisinde tanımlanan değişkenler diğer üst bloklarda kullanılamazlar ama alt bloklar içerisinde kullanılabilirler. Blok, herhangi bir yapıya ait küme parantezleri ( { } ) arasındadır.

Örnek 050:

```

using System;
namespace or050
{
    class Program
    {
        static void Main(string[] args)
        {
            int numara;
            numara=7;
            not=200;//not isimli değişken burada kullanılamaz. Çünkü daha önce bildirilmedi!
            Console.WriteLine(numara);
            int not;
            not=100;
            Console.WriteLine(not);
            Console.ReadLine();
        }
    }
}

```

---

Örnek 051:

```

using System;
namespace or051
{
    class Program
    {
        static void Main(string[] args)
        {
            int numara=7;
            if(numara==7)
            {
                int not=100;
                Console.WriteLine(not);//100
                numara++;//numara=numara+1=7+1=8;
                Console.WriteLine(numara);//8
            }
            not++; //not isimli değişken burada kullanılamaz! Çünkü bu blokta bildirilmedi!
            Console.WriteLine(not); //not isimli değişken burada kullanılamaz! Çünkü bu blokta bildirilmedi!
            Console.ReadLine();
        }
    }
}

```

---

**Tip Dönüşümleri****Sayısal – Sayısal Dönüşüm:**

- 1- Aynı veya farklı gruplara ait küçük (kapsanan) tip, büyük(kapsayan) tipe doğrudan dönüştürülebilir. Byte a=100; int b=500; double d=2.5; bildirimleri yapılmış iken d=b=a atamaları geçerlidir ancak a=b=d atamaları geçersizdir.
- 2- Aynı gruba ait büyük tip, küçük tipe çevrilirken (örneğin int tipi byte tipine çevrilirken) veya farklı gruplar arasında (örneğin double tip int tipine çevrilirken) dönüşüm uygulanır. Bu dönüşüm parantez içine çevrilecek tip((byte)(int)(double)(short)(long)(float)(sbyte)...) yazılarak yapılır. Örnek: byte a=100; int b=500; bildirimleri yapılmış iken küçük tip olan “a” değişkenine büyük tip olan “b” değişkeninin değeri atanacak ise bu atama “a=(byte)b;” şeklinde yapılmalıdır. Böylece tip dönüşümü yapıldığından atama geçerli olur. **Ancak** byte tipinde olan “a” değişkeninde saklanabilecek en büyük değer 255 olduğundan bu atama sonucunda veri kayıpları olacaktır. Benzer durum ondalıklı bir sayıyı bir tamsayıya atarken de ortaya çıkacaktır.

**Örnek:**

```
using System;
namespace sayi2sayi
{
    class Program
    {
        public static void Main(string[] args)
        {
            byte a=255;
            int b=200;//b değerini bir de 256 yaparak deneyiniz!
            double d=2147483647.999;//d değerini bir de 2147483649 yaparak deneyiniz!
            Console.WriteLine("b değeri a ya dönüşümle atanınca a=");
            Console.WriteLine(a=(byte)b);//200
            Console.WriteLine("ve d değeri b ye dönüşümle atanınca b=");
            Console.WriteLine(b=(int)d);// 2147483647
            /*Console.WriteLine("b değeri ikinci kez a ya dönüşümle atanınca a=");
            Console.WriteLine(a=(byte)b); */
            Console.ReadLine();
        }
    }
}
```

**Sayısal – Sözel Dönüşüm:****1. YOL**

Sayısal bir değeri stringe dönüştürmenin en kısa yolu onu boş bir string değerle toplamaktır.

Float d=10.33;

String s=" "+d;

**2. YOL**

Çevrim için özel fonksiyon kullanılabilir.

Söz dizimi:

String s=b.ToString(); //b sayısal değişkeninin string değere dönüştürülüp sonucun s isimli string değişkene aktarılması sağlanıyor.

**Sözel – Sayısal Dönüşüm (Ders kitabı – Yaşar, 2011, s:54)**

Özellikle dışarıdan girilen tüm veriler sözel değişkenlere alınır ve daha sonra matematiksel işleme alınacak olanlar için sözel- sayısal dönüşüm uygulanır. Bunun için her programlama diline özel sözel-sayısal dönüştürme metotları kullanılır.

C# için hazır sözel-sayısal dönüşüm metotları:

Byte a=byte.Parse(s); //s adındaki **string** sayısal dönüştürülerek **byte** tipindeki a değişkenine atanıyor.

short a=Int16.Parse(s); //s adındaki **string** sayısal dönüştürülerek **short** tipindeki a değişkenine atanıyor.

ushort a=UInt16.Parse(s); //s adındaki **string** sayısala dönüştürülerek **ushort** tipindeki a değişkenine atanıyor.  
 int a=Int32.Parse(s); //s adındaki **string** sayısala dönüştürülerek **int** tipindeki a değişkenine atanıyor.  
 long a=Int64.Parse(s); //s adındaki **string** sayısala dönüştürülerek **long** tipindeki a değişkenine atanıyor.  
 float a=Single.Parse(s); //s adındaki **string** sayısala dönüştürülerek **float** tipindeki a değişkenine atanıyor.  
 double a=Double.Parse(s); //s adındaki **string** sayısala dönüştürülerek **double** tipindeki a değişkenine atanıyor.

**Örnek:** sözel veri tiplerini sayısal veri tiplerine dönüştürme örnekleri.

```
namespace sozel2sayisal_dnsm
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayisal1 = 45,sayisal2,sayisal3;
            string il = "Manisa", plaka, mlaka;
            plaka = sayisal1.ToString();//sayisal veri tipinden sözele dönüşüm-1

            Console.WriteLine(il + " --> " + plaka);

            mlaka = sayisal1 + " ";//sayisal veri tipinden sözele dönüşüm-2
            Console.WriteLine("işte size sonunda bir boşluk karakteri olan mlaka: " + mlaka);
            //Sözel veri tipinden sayısal veri tipine Parse metodu ile dönüşüm
            // hedefveritipi.Parse(kaynakDeğer);
            sayisal2 = int.Parse(plaka);
            Console.WriteLine("Manisa nın plaka numarasının 2 katı="+ (2*sayisal2));

            //sözel veri tipinden sayısal veri tipine Convert metodu ile dönüşüm
            // Convert.HedefVeriTipi(KaynakDeğer);
            sayisal3 = Convert.ToInt32(mlaka);
            Console.WriteLine("Manisa plaka numarasının 20 eksiği="+ (sayisal3-20));
            Console.ReadKey();

        }
    }
}
```

**Sözel – Sözel Dönüşümü**

Char veri tipi string veri tipine, string veri tipi de char veri tipine doğrudan atanamaz. Bunun için çeşitli yöntemler vardır. Bir örnek ile açıklayalım:

```
using System;
namespace SozelToSozel
{
    class Program
    {
        static void Main(string[] args)
        {
            Char c = 'A';
            string s = "Merhaba";
            s = c.ToString(); //Tek karakteri string'e dönüştürme!
            Console.WriteLine("s stringinin yeni değeri=" + s);

            //c=s; --> olmaz. Çok karakterlik bir değer tek karakterlik bir alana yazılamaz!
            //string değer char değişkene atanamaz ama char bir diziye atanabilir.
            //şöyle ki:
            s = "Merhaba"; //s'yi yeniden merhaba yaptık!
            char[] ch_dizi = s.ToCharArray(); //s stringindeki her bir karakter ch_dizi'ye ayrı bir eleman olarak kopyalanır!
            foreach (char ch in ch_dizi) //ch_dizi'nin her bir elemanı tek tek okunarak ch'ye kopyalanır.
            {
                Console.WriteLine(ch); //ch'nin güncel değeri ekrana yazılır.
            }

            //Eğer bir karakter dizisinin karakterleri tek bir string olsun istenirse:
            char[] krk = new char[5];
            krk[0] = 'S';
            krk[1] = 'e';
            krk[2] = 'l';
            krk[3] = 'a';
            krk[4] = 'm';
            string cumle = new string(krk);
            Console.WriteLine("krk dizisinin karakterleri birleşip string oldu (cumle)= " + cumle);

            //Eğer tek karakterlik bir string char veri tipine dönüştürülecekse:
            string str = "A";
            char krk1 = char.Parse(str);
            Console.WriteLine("str stringi char veri tipine Parse() ile dönüştü (krk1)=" + krk1);

            //VEYA
            string str2 = "B";
            char krk2 = str2.ToCharArray()[0];
            Console.WriteLine("str2 stringi char'a ToCharArray ile dönüştü (krk2)=" + krk2);
            //CTRL+F5 ile deneyin!

        }
    }
}
```

**OPERATÖRLER (Ders kitabı – Yaşar, 2011, ss:56 - 65)**

Eşitliklerde çeşitli görevleri olan sembollere **operatör (işleç)** adı verilir. Hangi işlemin yapılacağını gösteren işarete **operatör (işleç)** denir.

Nelerin işleneceğine ise **operand** denir. Örneğin **a+b** söz diziminde (syntax) **+** işareti operatör, **a** ve **b** karakterleri ise operanddır.

**Matematiksel Operatörler**

Operatör	İşlevi	Örnek Kullanım
=	Atama	A=b
+	Toplama	A=a+b
-	Çıkarma	A=a-b
*	Çarpma	A=a*100
/	Bölme	A=a/2
%	Kalan	A=B%7 veya B%C veya 11%3
++	Bir arttırma	A++ → A=A+1, b=a++ → b=a ve a=a+1
--	Bir azaltma	b-- → B=B-1, b=-a → a=a-1 ve b=a

**Not 1:** Atamalarda eşittir (=) işaretinin soluna her zaman tek bir değişken veya sabit gelir.

**Not 2:** Atamalarda öncelikle eşitliğin sağ tarafı hesaplanır ve sonra sol tarafa atama yapılır.

**Not 3:** Eşitliğin sağındaki değişkenlerin değeri **++** ve **--** operatörleri dışında hiçbir zaman değişmez.

**Not 4:** **++** ve **--** operatörlerine atama ifadelerinde dikkat edilmesi gerekir. Bu operatörlerin sağda ve solda kullanılmaları arasında fark vardır. Bu operatörler değişkenin sağında ise önce değişken değeri işleme girer ve sonra değeri değişir. Solunda yer alıyorsa bu takdirde önce değişkenin değeri değişir ve sonra değişken yeni değeri ile işleme girer.

**Azaltma Arttırma Operatörleri**

Azaltma operatörü **--** ve arttırma operatörü ise **++** operatörüdür.

a++ → Eşdeğeri: a=a+1. a değişkeninin değeri bir artar.

a-- → Eşdeğeri: a=a-1. A değişkeninin değeri bir azalır.

Bu operatörlerin sağda ve solda kullanılmaları arasında fark vardır. Bu operatörler değişkenin sağında ise önce değişken değeri işleme girer ve sonra değeri değişir. Solunda yer alıyorsa bu takdirde önce değişkenin değeri değişir ve sonra değişken yeni değeri ile işleme girer.

int a=10, b=5,c; değişkenleri tanımlanmış iken;

**Örnek a:**

C=b\*(++a); //önce a değişkeninin değeri bir artar ve sonra bu yeni değer ile b değeri çarpılır. a=11, C=5\*(11)=55

**Örnek b:**

C=b\*a++;//a değişkeni mevcut değeri ile işleme girer ve sonra değeri bir artar. C=5\*10=50, a=10+1=11

**Örnek c:**

C=b\*(--a);//önce a değişken değeri bir azalır ve sonra bu yeni değer ile b değeri çarpılır. a=10-1=9, C=5\*9=45

**Örnek d:**

C=b\*a--;//a değişkeni mevcut değeri ile işleme girer ve sonra değeri bir azalır. C=5\*10=50, a=10-1=9

**Örnek 058:**

```

using System;
namespace or058
{
    class Program
    {
        public static void Main(string[] args)
        {
            int k,a=10,b=100;
            k=50+20;//k=70
            Console.WriteLine("Toplam Sonucu: k=50+20= "+k);//70
            k=b;//k=100
            Console.WriteLine("k=b Atama Sonucu: "+k);//100
            Console.WriteLine("İlk Değer a="+a);//10
            a=999;
            k = a / 10;//k=99 --> programlamada tamsayı/tamsayı sonucu her zaman tamsayıdır.
            Console.WriteLine("Bölme Sonucu k="+k);//99
            k=b%7;//k=2
            Console.WriteLine("b/7 Bölümünden Kalan:"+k);//2
            k=b++;//k=b=100; b=b+1=101;
            Console.WriteLine("Sağdan Atama k="+k+" ve b="+b);//100 ve 101
            k--a;//a=a-1=100-1=99; k=a=99
            Console.WriteLine("Soldan Atama k="+k+" ve a="+a);//99 ve 99
            Console.ReadLine();
        }
    }
}

```

### String Operatörleri (Yaşar, s:245)

String olarak tanımlanan bir değişkenin bellekteki başlangıç adresi o değişkene ait ilk karakterin bellek adresidir ve stringin elemanları sıfır sayısından başlayarak numaralandırılır.

String s="KIRKAĞAÇ";

Olarak tanımlanan s değişkeninin elemanlarının sırası ve bellekteki yerleşimi aşağıdaki tabloda gösterilmektedir.

Eleman Sırası	0	1	2	3	4	5	6	7	
Eleman İçeriği	K	I	R	K	A	Ğ	A	Ç	← S string değişkeni bellek yerleşimi

Örneğin s stringinin 5. Elemanı dendiğinde bu eleman **Ğ** karakteridir ve bu elemana ulaşmak veya bu elemanı ifade etmek için C# programlama dilinde **s[5]** yazılır.

Stringler üzerinde işlem yapmak için çok sayıda metot kullanılır. Ancak stringleri toplarken normal toplama operatörü kullanılır, ancak stringlerde toplama işleminin değişme özelliği yoktur!

S1="Kırkağaç";

S2="MYO";

SB=" ";

S3=S1+SB+S2; //S3= Kırkağaç MYO

S4=S2+SB+S1; //S4=MYO Kırkağaç;

Stringler üzerinde işlem yapan metotlar hakkında ayrıntılı bilgi için bkz. Yaşar, s:247

Örnek StringOpOr01:

```

using System;
namespace StringOpOr01{
    class Program {
        public static void Main(string[] args) {
            string s1="KIRKAĞAÇ", s2="MYO",s3;//s3=s1+s2
            Console.WriteLine("s1+s2="+s1+" "+s2); Console.WriteLine("s2+s1="+s2+" "+s1); Console.WriteLine(s1[5]);
            Console.Write("Çıkış İçin Enter Tuşuna Basınız . . !"); Console.ReadLine(); }
    }
}

```



**Atama Operatörleri**

Atamaları kolaylaştırmak için bazı özel atama operatörleri vardır. Bunların yerine eşdeğerleri de kullanılabilir.

**Söz dizimi(syntax):** *Değişken1"operatör"=değişken2* → eşdeğeri: *Değişken1=değişken1"operatör"Değişken2*

**NOT:** Değişken2 yerine sabit bir değer veya bir sabit adı yazılabilir!

Operatör	Kullanımı	Eşdeğeri
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b
<<=	a<<=b	a=a<<b a değerini b kadar sola kaydır ve sonucu a'ya yaz.
>>=	a>>=b	a=a>>b a değerini b kadar sağa kaydır ve sonucu a'ya yaz.
=	a =b	a=a b a ve b değerlerini ikili veya'la sonucu a'ya yaz.
&=	a&=b	a=a&b a ve b değerlerini ikili ve'le sonucu a'ya yaz.
^=	a^=b	a=a^b a ve b değerlerini ikili özel veya'la sonucu a'ya yaz.

**Örnek 062:**

```
using System;
```

```
namespace or062
```

```
{
```

```
    class Program
```

```
    {
```

```
        public static void Main(string[] args)
```

```
        {
```

```
            int a=10, b=100;
```

```
            a+=b; //a=a+b=110
```

```
            Console.WriteLine("a nın son değeri="+a); //110
```

```
            a/=2; //a=a/2=55
```

```
            Console.WriteLine("a nın son değeri="+a); //55
```

```
            Console.ReadLine();
```

```
        }
```

```
    }
```

```
}
```

## Mantıksal Operatörler

Karar ifadelerinde veya şartlı döngülerde kullanılırlar. Değişken değerlerinde herhangi bir değişikliğe yol açmazlar. Bu operatörlerin yaptığı işlemin sonucu True(evet/doğru/sıfırdan farklı bir değer) veya False(hayır/yanlış/sıfır) olur.

&	VE
	VEYA
	Kısa Devre VEYA
&&	Kısa Devre VE
!	DEĞİL

Mantıksal operatörler ise iki bool değerini karşılaştırmak için kullanılır.

Bunu parola giriş ekranında Kullanıcı Adı ve Parola'nın sınıandığı bir örnekte açıklayalım.

```
1 string KullaniciAdi = txtKullaniciAdi.Text;
2 string Parola = txtParola.Text;
3
4 if( KullaniciAdi == "admin" & Parola == "123" )
5     MessageBox.Show("Tebrikler, giriş başarılı.");
6 else
7     MessageBox.Show("Üzgünüm, giriş başarısız.")
```

4 nolu satırda görüldüğü gibi iki adet ilişkisel operatör kullanılmış. KullaniciAdi == "admin" ve Parola == "123". Daha önce ilişkisel ve mantıksal operatörlerin her zaman bool tipinde değer döndürdüğünden bahsetmiştik. Dolayısıyla bu iki ilişkisel sınavanın bool tipindeki sonuçlarını & mantıksal operatörü ile sinayarak her iki değerinde doğrulunu kontrol etmiş olduk.

### Mantıksal Operatörlerin Sonuçları

p	q	p & q	p   q	!p
false	false	false	false	true
true	false	false	true	false
false	true	false	true	true
true	true	true	true	false

### Kısa Devre Operatörler

Kısa devre operatörlerinin normal VE ( & ) ve VEYA ( | ) operatörlerinden çalışma mantığı biraz farklıdır. Normal operatörlerde ne olursa olsun her iki operatörde kontrol edilir. Örneğin normal VE ( & ) operatöründe ilk operatörün sonucu yanlış ( false ) ise ikinci operatörde ne olursa olsun sonuç yanlış ( false ) çıkacaktır. Aynı şekilde VEYA ( | ) operatörü kullanıldığında ilk operatörün sonucu doğru ( true ) ise ikinci operatör ne olursa olsun sonuç doğru ( true ) çıkacaktır. Bu iki durumda da ikinci operatörün kontrol edilmesi gereksizdir. Kısa devre operatörleri ( && , || ) kullanıldığı zaman bu durumlarda ikinci operatör kontrol edilmez. Bu performans artışı sağlayacağı gibi bazı istisnai durumlarda programın hata vermesini de engelleyecektir.

```
1 string KullaniciAdi = txtKullaniciAdi.Text;
2 string Parola = txtParola.Text;
```

3

```

4 if( KullaniciAdi == "admin" & Parola == "123" )
5     MessageBox.Show("Tebrikler, giriş başarılı.");
6 else
7     MessageBox.Show("Üzgünüm, giriş başarısız.")

```

Örnekte kullanıcı adı yanlış girilse dahi parola da kontrol edilmektedir. Hâlbuki kullanıcı adı yanlış girilirse parola doğru girilse dahi sonuç yanlış çıkacaktır.

```
false & true = false
```

Dolayısıyla parolayı ikinci bir kez kontrol etmeye gerek yoktur. 4. satırdaki VE ( & ) operatörünü Kısa Devre VE ( && ) yaparsak kullanıcı adının yanlış girilmesi halinde Parola kontrolü yapılmadan sonuca daha kısa sürede ulaşarak yanlış ( false ) sonucunu döndürecektir.

```

1 string KullaniciAdi = txtKullaniciAdi.Text;
2 string Parola = txtParola.Text;
3
4 if( KullaniciAdi == "admin" && Parola == "123" )
5     MessageBox.Show("Tebrikler, giriş başarılı.");
6 else
7     MessageBox.Show("Üzgünüm, giriş başarısız.")

```

Bu yüzden C#'ta normal VE ( & ) ve VEYA ( | ) operatörlerinden daha çok Kısa Devre ( && ) ve Kısa Devre VEYA ( || ) kullanılmaktadır.

## ? Operatörü

? Operatörü C#'ta oldukça kullanışlıdır. Bu operatör üç parçadan oluşur. Bu yüzden üçlü ( ternary ) operatör de denir. ?: operatörü bazı durumlarda ilerleyen bölümlerde göreceğimiz if-else deyimi yerine kullanılır. ?: operatörü genellikle bir atama operatörü ile beraber kullanılır.

Yazımı şu şekildedir:

```
değişken = koşul ? doğru kısım : yanlış kısım ;
```

koşul bölümünde bir değişkenin değeri ilişkisel ve mantıksal operatörler ile sınanır. Sonuç doğru ( true ) ise doğru kısım değeri değişkene atanır, sonuç yanlış ( false ) ise yanlış kısım değeri değişkene atanır.

Bir sayının mutlak değerini alacağımızı varsayalım. Önce sayının sıfırdan küçük olup olmadığını kontrol etmemiz gerekir. Eğer sıfırdan küçük ise sayıyı eksi ile çarparsak, sıfırdan büyük ise sayının kendi değerini alacağız.

```

1 int sayi = -5;
2 int mutlakDeger;
3 mutlakDeger = ( sayi < 0 ) ? -1 * sayi : sayi;

```

Burada mutlakDeger değişkeni sonuç olarak 5'tir. 3 nolu satırda atama yapılmadan evvel sayi'nin sıfırdan küçük olup olmadığı sınanmış sonuç olarak doğru ( true ) çıkmıştır ve ?: operatörünün doğru kısmı çalıştırılmıştır.

Kaynak: Ayrılmaz, ss:98-100

**Karşılaştırma Operatörleri**

Ürettikleri sonuçlar ve gösterdikleri etkiler mantıksal operatörler gibidir. Karar ifadelerinde veya şartlı döngülerde kullanılırlar. Değişken değerlerinde herhangi bir değişikliğe yol açmazlar. Bu operatörlerin yaptığı işlemin sonucu True(evett/doğru/sıfırdan farklı bir değer) veya False(hayır/yanlış/sıfır) olur.

Operatör	Anlamı	Kullanımı
==	Eşit mi? / Eşitse	a==b
<	Küçük mü? / Küçükse	a<b
>	Büyük mü? / Büyükse	a>b
<=	Küçük ya da eşit mi? Soldaki değer en fazla sağdaki değer kadar mı?	a<=b
>=	Büyük ya da eşit mi? Soldaki değer en az sağdaki değer kadar mı?	a>=b
!=	Eşit değil mi? (<>) / Eşit değilse / Farklıysa	a!=b

Örnek:

<b>GELİRİ EN FAZLA 1000TL OLUNCA SONUÇ TRUE ÇIKSIN İSTENİYORSA</b>
<b>!(GELİR&lt;=1000));//değilin değili kendisidir!</b>
<b>GELİR&lt;1001;</b>
<b>1000&gt;=GELİR;</b>
<b>1001&gt;GELİR;</b>
<b>!(GELİR&gt;1000);</b>

**Bit İşlem Operatörleri**

Sıfır ve bir düzeyinde ikilik sayı sistemine göre işlem yapmak için kullanılan operatörlerdir.

Operatör	İşlevi	Örnek Kullanım
	Veya	a=a   b
&	Ve	a=a & b
^	XOR(eşitsizlik – özel veya) Aynı ise sıfır, farklı ise 1	a= a ^ 30
~	~→ Tilda işareti. Değil	a=~a
<<	Sola kaydırma	a=a<<2
>>	Sağa kaydırma	a=a>>2

A	B	A & B	A   B	A ^ B	~A	~B
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

**NOT:** Sağa ve sola kaydırmalarda kaydırma yönüne göre sondaki bitler başına alınmaz. Bir bytelik bir verinin baş ve sonlarındaki bitler 0. Ve 7. Bitlerdir. Aşağıdaki örnekte 10011101(157) sayısını bir byte(8 bit) uzunlukta kabul edip inceleyiniz.

10011101>>1=01001110  
 <<10011101<<2=01110100

Örnek 060:

```
using System;
namespace or060{
    class Program {
        public static void Main(string[] args)    {
            int A=10, B=30, C;
            C=A^B;      Console.WriteLine("XOR "+C); //20
            C=A&B;      Console.WriteLine("AND "+C); //10
            C=A|B;      Console.WriteLine("OR "+C); //30
            C=A>>2;     Console.WriteLine("SAĞA KAYDIRMA "+C); //2
            C = A<<2;
            Console.WriteLine("SOLA KAYDIRMA " + C); //40
            Console.ReadLine();
        }
    }
}
```

	32	16	8	4	2	1
B=30	0	1	1	1	1	0
A=10	0	0	1	0	1	0
A^B=20	0	1	0	1	0	0
B=30	0	1	1	1	1	0
A=10	0	0	1	0	1	0
A&B=10	0	0	1	0	1	0
B=30	0	1	1	1	1	0
A=10	0	0	1	0	1	0
A B=30	0	1	1	1	1	0
A=10	0	0	1	0	1	0
A>>2=2	0	0	0	1	0	1
	0	0	0	0	1	0

### Operatörlerde İşlem Önceliği

Operatörlerde işlem önceliği ile ilgili genel kurallar şöyledir:

- Önce parantez içindeki işlemler yapılır. İç içe parantezler var ise önce en içteki parantezin içindeki işlem tamamlanmalıdır.
- Sonra üslü ifadeler yapılır. Köklü ifadeler de üslü ifadeler gibi düşünülebilir.
- Eşittir (=) operatörünün sağ tarafında işlem önceliği eşit olan operatörler var ise işlemler soldan sağa doğru yapılır. Örneğin toplama ve çıkarma operatörlerinin işlem önceliği aynıdır. Bu durumda a=b+c-d işleminde işlem eşittir operatörünün sağ tarafında soldan sağa doğru yapılır. (önce b+c işlemi yapılır ve daha sonra (b+c)-d işlemi yapılır.)
- Eşittir operatörünün sağ tarafındaki sonuç sol tarafındaki değişkene aktarılır. Değeri değişen değişken her zaman eşittir operatörünün sol tarafındaki değişkendir. **Not:** arttırma ve eksiltme operatörleri hariç.
- Aşağıdaki tabloda herhangi bir aynı satırda yer alan operatörlerin birbirlerine göre önceliği yoktur. Bu durumlarda soldan sağa doğru gidilirken ilk önce gelen operatörün işlemi yapılır. Örneğin 3. Satırda \*, / ve % operatörleri sırayla yazılmasına rağmen, 5+10%7\*8/2 gibi bir denklem varsa soldan yaklaşıp gelen ilk operatör olan + en son yapılır çünkü diğer operatörlerin önceliği daha yüksektir. Yani bu örnekte önce %, sonra \* ve sonra / işlemi ve en sonunda da + işlemi yapılır ve sonuç 17 bulunur.

İşlem Önceliği	Operatör (Öncelik numarası küçük olanın, önceliği yüksek)
1	X++, X--, ., [ ]
2	++X, --X, ~, -X, !
3	*, /, % (5+10%7*8/2) -> sonuç= 17
4	+, -
5	<<, >>
6	<, >, <=, >=
7	==, !=
8	&
9	^
10	
11	&&
12	
13	?:
14	=, +=, -=, *=, /=, %=, <<=, >>=,  =, &=, ^=

((5+10)%7\*8/2)

Örnek 064:

```
using System;
namespace or064
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a;
            a=3+2*4;// önce 2*4=8 ve sonra 3+8=11
            Console.WriteLine(a);//
            a=(3+2)*4;// önce 3+2=5 ve sonra 5*4=20
            Console.WriteLine(a);//
            Console.ReadLine();
        }
    }
}
```

---

### Operatör İşlemleri Hata Mesajları

Operatörler yukarıda açıklanan kurallara aykırı kullanıldığında hata mesajlarının ortaya çıkması doğaldır.

Örnek:

```
using System;
namespace OpHataOr01{
    class Program {
        public static void Main(string[] args) {
            int a=5, b=10, c=20;
            double d=7.8;
            string s1="Kırkağaç", s2="MYO";
            char karakter = 'Z';
            Console.WriteLine(s1-s2);
            Console.WriteLine(s1*s2);
            Console.WriteLine(s1/s2);
            Console.WriteLine(5 = b);//literal değerlere başka değer atanamaz.
            Console.WriteLine('A' = karakter);//literal değerlere başka değer atanamaz.
            Console.WriteLine(a&d);
            Console.WriteLine(++s1);
            Console.WriteLine(b%0);
            Console.WriteLine("Devam Etmek için Enter Tuşuna Basınız. . . !");
            Console.ReadLine(); } } }
```

**NOT:** Programlama dili içerisinde “Merhaba”, ‘K’, 3, 5, 2.718 gibi yazılan ve program içerisinde aynen yazıldığı gibi kullanılan değerlere **literal** değer denir.

---

**GİRİŞ – ÇIKIŞ (INPUT/OUTPUT) İŞLEMLERİ**

Ders kitabı (Yaşar, ss: 119 – 126)

Konsol programın çıktılarını ekrana göndermek ve programa değer almak için giriş – çıkış metotları kullanılır.

**Çıkış İşlemleri****Write / Writeline**

Write metodu ile ekrana yazılan ileti bitince imleç aynı satırın sonunda bekler.

Writeline metodu ile ekrana yazılan ileti bitince imleç bir sonraki (alt) satıra geçer.

Söz dizimi(syntax):

Console.Write(“ileti buraya yazılır”+değişken\_a+(toplam=b+c+d)+ `Byte.MinValue`);Console.WriteLine(“ileti buraya yazılır”+değişken\_a+(toplam=b+c+d)+ `Byte.MinValue`);

- Sabit iletiler, sayısal ve sözel değerler ile programlama dili içerisinde tanımlanan değişkenlerin ve sabitlerin değerlerini birleştirmek için **+** operatörü kullanılır.
- Sabit iletiler mutlaka çift tırnak işaretleri arasında yazılmalıdır.
- Metot içerisinde parantez içinde başka metotlar veya işlemler de yazılabilir.

C# tarafından kullanılan bazı özel karakterleri de ekrana yazmak ve ekrana yazılacak olan iletilere biçim kazandırmak için **karakter çıkış dizisi (character escape sequence)** denen bir takım işaretler kullanılır. Bu işaretler ve anlamları aşağıdaki gibidir:

İşaret	Anlamı
\'	Tek Tırnak
\"	Çift Tırnak
\\	Sola eğik(ters) bölü
\0	Null (Null)
\a	Uyarı (Alert)
\b	Geri silme (BackSpace)- ardından bir boşluk bırakılmalı!
\f	Sayfa İleri(Form Feed). Yazıcı için bir sonraki kâğıdın başı.
\n	Satır Sonu(New Line)
\r	Satır Başı(Carriage Return)
\t	Yatay Sekme (Tab)
\v	Dikey Sekme (Vertical Tab). Yazıcı için belirtilen satır.
\u	Xxxx Hexadecimal Unicode karakterleri

Örnek 120:

```
using System;
namespace or120
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a=2;char c=' ';
            Console.WriteLine("Sabit iletiler çift tırnak(\" \") karakterleri arasındadır!");
            Console.WriteLine("murat.albayrak\u0040bayar.edu.tr");
            Console.WriteLine("D:\\ALBAYRAKLAB3\\PROGTEMELLERİ\\PT02.DOCX");
            Console.WriteLine(a+"'nın karesi = "+a*a);
            Console.WriteLine("Pi sayısı =\t"+3.14);//bir tab kadar sağa değer yazılacak
            Console.WriteLine(784+"\n");//imlec bir alt satıra gider
            Console.WriteLine("BU YAZININ SONDAN\b BİR KARAKTERİ SİLİNECEK\b ");
            Console.ReadKey(true);
            if (c=='\0') Console.WriteLine("c değişkeni NULL karakter . . .!");
            else Console.WriteLine("c değişkeni NULL karakter DEĞİL . . .!");
            Console.WriteLine("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ . . .\a \r");
            Console.ReadKey(true);
        }
    }
}
```

**Not:** Sayısal veri tipleri ile Tarih ve Zaman veri tipleri için özel biçim belirleyici karakterler de kullanılır. Buna ilişkin tablo aşağıdaki gibidir:

SAYISAL TİPLER İÇİN		TARİH ve ZAMAN TİPLERİ İÇİN	
Karakter	Anlamı	Karakter	Anlamı
C[n]	Para birimi ve biçimi	d	MM/dd/yyyy
D[n]	Desimal – Onlu sayı sistemi	g	MM/dd/yyyy HH:mm
E[n], e[n]	Ondalık sayı bilimsel gösterimi	s	yyyy-MM-dd- HH:mm:ss
F[n]	Ondalık sayı	t	HH:mm
G[n]	Genel biçim	u	yyyy-MM-dd- HH:mm:ss
N[n]	Sayı	U	dddd,MMMM dd,yyyy HH:mm:ss
X[n],x[n]	Onaltılık gösterim	F	dddd, MMMM dd, yyyy HH:mm:ss

Sayısal veri tipleri için n parametresi isteğe bağlıdır.

Tarih ve zaman veri tipleri için: M=ay, d=gün, y=yıl, H=saat, m=dakika, s=saniye için kullanılmıştır.

Tablodaki karakterler küme parantezleri {değişken\_sıra\_no :Biçimkarakter[n]} içinde yazılır. Küme parantezleri arasındaki ilk sayı değişken sıra numarasını gösterir ve sıfırdan başlar.



Örnek 121:

```

using System;
namespace or121
{
    class Program
    {
        public static void Main(string[] args)
        {
            double i=6543.124;
            int a=8762;
            Console.WriteLine("A değişkeni ={0} dir ",a);//8762
            Console.WriteLine("{0:C5}",i);//Para birimi 6543,12400 TL
            Console.WriteLine("{0:D6}",a);//Desimal Tamsayı 008762
            Console.WriteLine("{0:E}",i);//Bilimsel Ondalıklı Sayı 6,543124E+003
            Console.WriteLine("{0:F4}",i);//Ondalıklı Sayı 6543,1240
            Console.WriteLine("{0:G3}",a);//genel biçim 8,76E+03
            Console.WriteLine("{0:N1}",a);//sayı 8762,0
            Console.WriteLine("{0:X}",a);//büyük harf onaltılık 223A
            Console.WriteLine("{0:x}",a);//küçük harf onaltılık 223a
            Console.WriteLine("a değişkeninin değeri={1:F6} ve i değişkeninin değeri={0:N4}",a,i);
            Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ . . . ");
            Console.ReadKey(true);
        }
    }
}

```

birazdan çift tırnak kapanacak virgöl konacak ve virgülden sonra sıfırıncı sıradaki de eri buraya yaz.

virgülden sonra 5 basamak

Örnek 122:

```

using System;
namespace or122
{
    class Program
    {
        public static void Main(string[] args)
        {
            double i=6543.124;
            int a=8762;
            Console.WriteLine("A değişkeni ={0} dir ",a);
            Console.WriteLine("{0:E2}",i);//Bilimsel Ondalıklı
            Console.WriteLine("{0:F1}",a);//Ondalıklı sayı
            Console.WriteLine("{0:G6}",i);//genel sayı
            Console.WriteLine("{0:N6}",i);//sayı
            Console.WriteLine("{0:X6}",a);//büyük harf onaltılık
            Console.WriteLine("{0:x4}",a);//küçük harf onaltılık
            Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ . . . ");
            Console.ReadKey(true);
        }
    }
}

```

Örnek 123:

```

using System;
namespace or123
{
    class Program
    {
        public static void Main(string[] args)
        {
            DateTime tz=DateTime.Now;//güncel tarih ve saat tz'ye atanıyor
            Console.WriteLine("Tarih Saat {0}",tz);
            Console.WriteLine("Tarih {0:d}",tz);
            Console.WriteLine("Uzun Tarih Saat {0:f}",tz);
            Console.WriteLine("Gün ay {0:m}",tz);
            Console.WriteLine("Saat {0:T}",tz);
            Console.WriteLine("Ay ve Yıl {0:y}",tz);
            Console.WriteLine("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ . . .");
            Console.ReadKey(true);
        }
    }
}

```

---

## Giriş İşlemleri

### Read/ReadLine

Read metodu ile dışarıdan girilen ilk karakterin ASCII karşılığı elde edilir.

ReadLine metodu ile ise dışarıdan girilen birden fazla karakter, yani string programa alınır.

Eğer istenen değer de string ise alınan değer doğrudan program içerisinde kullanılır, ancak başka tiplere ait veriler isteniyorsa tip dönüştürme metotları kullanılır. Bakınız: değişkenler konusu, sözel-sayısal tip dönüşümleri.

Ders kitabı sf:124 tabloları.

Örneğin dışarıdan istenen veri tipi int ise ve girilen değer s değişkenine atanıyorsa:

String s=Console.ReadLine();//klavyeden girilen en son stringi s değişkenine alır.

int b=Int32.Parse(s);//s değişkenindeki değeri int'e çevirerek b değişkenine atar.

veya

string s=Console.ReadLine();//klavyeden girilen en son değeri s stringine alır.

int b=Convert.ToInt32(s);//s stringindeki değeri int'e çevirerek b değişkenine aktarır.

**NOT:** Char tipinde veri almak için **ReadKey** metodu da kullanılabilir. **ReadKey** metodu karakteri yazdıktan sonra enter tuşuna gerek duymadan, anında değişkene atar. Kullanım için **ConsoleKeyInfo** sınıfından bir nesne tanımlanmalıdır. Bu nesnenin **KeyChar** metodu basılan karakteri verir.

**ConsoleKeyInfo c=Console.ReadKey();**

Veya

ConsoleKeyInfo c;

...;

İşlemler;

....;

Console.WriteLine("bir karakter giriniz");

c=Console.ReadKey();//klavyeden girilen ilk karakter c değişkenine alınır.

Console.WriteLine(c.**KeyChar**);//c değişkeninin değeri ekrana yazdırılıyor.

**Örnek charDENEME:**

```

using System;
namespace charDENEME
{
    class Program
    {
        public static void Main(string[] args)
        {
            Console.Write("Bir tuşa basınız: ");
            ConsoleKeyInfo c = Console.ReadKey();
            Console.WriteLine("\nBasılan ilk tuş:" + c.KeyChar);
            ConsoleKeyInfo c2;
            Console.Write("\nBaşka bir tuşa basınız: ");
            c2 = Console.ReadKey();
            Console.WriteLine("\nBasılan Diğer Tuş: " + c2.KeyChar);
            if (c.KeyChar == c2.KeyChar)
                Console.WriteLine("Girilen ilk iki karakter birbirine eşit..!");
            else
                Console.WriteLine("Girilen ilk iki karakter EŞİT DEĞİL!");
            Console.WriteLine("Press ENTER key to continue . . . ");
            Console.ReadLine();
        }
    }
}

```

---

int tipinde sayı için → Console.Write("Bir tamsayı giriniz:");

int n=Convert.ToInt32(Console.ReadLine());//klavyeden girilen değer okunarak int'e çevriliyor ve n değişkenine aktarılıyor.

double tipinde sayı için → Console.Write("Bir ondalıklı giriniz:");

double d=Convert.ToDouble(Console.ReadLine());//klavyeden girilen değer okunarak double'a çevriliyor ve d değişkenine aktarılıyor.

**Örnek 125:**

```

using System;
namespace or125
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a;
            Double x;
            String s;
            Console.Write("Tam Sayı Gir :");
            s=Console.ReadLine();//klavyeden girilen değer okunarak s değişkenine alınıyor
            a=Convert.ToInt32(s);//s değeri int'e dönüştürülüyor ve a değişkenine aktarılıyor.
            Console.WriteLine("Girilen Tam Sayı="+a);
            Console.Write("Adınız? ");
            s=Console.ReadLine();
            Console.Write("Merhaba "+s+".\nTanıştığımıza memnun oldum.\n");
            Console.Write("Pi sayısı nedir? ");
            s=Console.ReadLine();
            x=Double.Parse(s);
            Console.WriteLine("Girdiğiniz Pi Sayısı="+x);
            Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ . . . ");
            Console.ReadKey(true);
        }
    }
}

```

**ŞART İFADELERİ (Ders kitabı – Yaşar, 2011, ss:66 - 79)**

Belirli koşullara göre belirli işlemler yaptırarak programı dallandırmak için kullanılan ifadelere şart veya koşul ifadeleri denir. Üç gruba ayrılırlar:

- a) If yapısı
- b) Switch yapısı
- c) ?: yapısı

**5.1 If Yapısı**

if(koşul)

koşul doğru ise yapılacak işlem;

. //koşul doğru da olsa yanlış da olsa if deyimi işini bitirince program if deyiminden sonra

. //kaldığı yerden devam eder.

.

.

.

.

**NOT:** eğer if deyiminden sonra yapılacak birden fazla işlem var ise bu işlemler blok içine alınır.

if(koşul)

{ //if bloğu açılıyor.

Koşul doğru ise yapılacak işlem1;

Koşul doğru ise yapılacak işlem2;

...

...

Koşul doğru ise yapılacak işlem\_N;

} //if bloğu kapatılıyor!

./if bloğu bittikten sonra program kaldığı yerden devam eder!

.

.

.

**If/Else Yapısı**

if için yazılan koşul doğru ise yapılacaklar farklı, yanlış ise yapılacaklar farklı olduğunda ise if ile birlikte else kelimesi de kullanılır.

if(koşul)

koşul doğru ise yapılacak işlem;

else

koşul yanlış ise yapılacak işlem;

./if – else yapısından sonra program kaldığı yerden devam eder!

.

.

.

.

.

**NOT:** if için yazılan koşul doğru olduğunda yapılacak birden fazla işlem var ise, aynı zamanda yanlış olduğunda da yapılacak birden fazla işlem var ise tüm bu işlemler de blok içerisinde yazılır!

```

if(koşul)
    { //koşul doğru olduğunda yapılacak işlemler bloğu açılıyor.
      Koşul doğru ise yapılacak işlem1;
      Koşul doğru ise yapılacak işlem2;
      ...
      ...
      Koşul doğru ise yapılacak işlem_N;
    } //koşul doğru olduğunda yapılacak işlemler bloğu kapatılıyor!
else
    { //koşul yanlış olduğunda yapılacak işlemler bloğu açılıyor.
      Koşul yanlış ise yapılacak işlem1;
      Koşul yanlış ise yapılacak işlem2;
      ...
      ...
      Koşul yanlış ise yapılacak işlem_N;
    } //koşul yanlış olduğunda yapılacak işlemler bloğu kapatılıyor!
} //if – else yapısından sonra program kaldığı yerden devam eder!
.
.
.
.
.

```

---

**Örnek (Yaşar, 2011, 68):**

```

using System;
namespace or068
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = -100, mutlak_deger;
            mutlak_deger = a;
            if (a < 0)
                mutlak_deger = (-1) * a;
            Console.WriteLine(mutlak_deger);
            Console.ReadLine();
        }
    }
}

```

---

**Örnek (Yaşar, 2011, 69):**

```

using System;
namespace or069
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = -100;
            if (a > 0)
                Console.WriteLine("a değeri pozitif");
            else
                Console.WriteLine("a değeri negatif");
            Console.WriteLine("PROGRAMI SONLANDIRMAK İÇİN ENTER TUŞUNA BASINIZ..!"); //bu satır koşul
            //doğru da olsa, yanlış ta olsa çalışır!
            Console.ReadLine();
        }
    }
}

```

---

**If-Else if – else if... else (iç içe if-else) Yapısı**

Birden fazla koşula bağlı olarak farklı işlemler yaptırmak için kullanılır.

```

if(koşul1)
    { //koşul1 doğru olduğunda yapılacak işlemler bloğu açılıyor.
      Koşul1 doğru ise yapılacak işlem1;
      Koşul1 doğru ise yapılacak işlem2;
      ...
      ...
      Koşul1 doğru ise yapılacak işlem_N;
    } //koşul1 doğru olduğunda yapılacak işlemler bloğu kapatılıyor!
Else if(koşul2)
    { //koşul2 doğru olduğunda yapılacak işlemler bloğu açılıyor.
      Koşul2 doğru ise yapılacak işlem1;
      Koşul2 doğru ise yapılacak işlem2;
      ...
      ...
      Koşul2 doğru ise yapılacak işlem_N;
    } //koşul2 doğru olduğunda yapılacak işlemler bloğu kapatılıyor!
Else
    { //koşul1 ve koşul2 her ikisi de yanlış olduğunda yapılacak işlemler bloğu açılıyor!
      İşlem1;
      İşlem2;
      .
      .
      İşlem_N;
    } //koşul1 ve koşul2 her ikisi de yanlış olduğunda yapılacak işlemler bloğu kapatılıyor!
.//if – else yapısından sonra program kaldığı yerden devam eder!
.
.
.
.
.

```

**Örnek (Yaşar, 2011, 69B):**

```

using System;
namespace or069b
{
class Program
{
static void Main(string[] args)
{
int a = -10;
if (a > 0)
Console.WriteLine("a değeri pozitif");
elseif (a == 0)
Console.WriteLine("a değeri sıfır");
else
Console.WriteLine("a değeri negatif");
Console.WriteLine("PROGRAMI SONLANDIRMAK İÇİN ENTER TUŞUNA BASINIZ..!"); //bu satır koşul
doğru da olsa, yanlış ta olsa çalışır!
Console.ReadLine();
}
}
}

```

**Örnek (Yaşar, 2011, 175):** A, B, C katsayıları dışarıdan girilen ikinci dereceden bir bilinmeyenli denklemin köklerini bulan program.

**İpucu:** Öncelikle 2. Dereceden bir bilinmeyenli denklemin ve köklerinin nasıl bulunduğu bilinmelidir.

İkinci dereceden bir bilinmeyenli denklem:

$$aX^2+bX+c = 0$$

a, b, c katsayıları verildiğinde bu denklemi sağlayan x değerleri aşağıdaki gibi hesaplanır:

$\Delta = b^2 - 4 \cdot a \cdot c$  formülü ile delta hesaplanır. Delta değerine göre köklerin gerçel olup olmadığına bakılır.

- Delta < 0 ise gerçel kök yoktur.
- Delta > 0 ise 2 adet farklı gerçel kök vardır.

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{ve} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

- Delta = 0 ise eşit iki gerçel kök vardır.

$$x_1 = x_2 = \frac{-b}{2 \cdot a}$$

using System;

namespace or175

```
{
    class Program
    {
        public static void Main(string[] args)
        {
            double x1, x2, delta;
            int a,b,c;
            Console.Write("a katsayısı: ");
            a=Convert.ToInt32(Console.ReadLine());
            Console.Write("b katsayısı: ");
            b=Convert.ToInt32(Console.ReadLine());
            Console.Write("c katsayısı: ");
            c=Convert.ToInt32(Console.ReadLine());
            delta=(double)(b*b)-(4*a*c);
            if (delta<0) Console.WriteLine("Denklemin Gerçel Kökleri Yoktur!");
            else if(delta==0)
            {
                x1=(double)-b/2*a;
                x2=x1;
                Console.WriteLine("x1=x2= "+x1);
            }
            else
            {
                x1=(-b-Math.Sqrt(delta))/(2*a);
                x2=(-b+Math.Sqrt(delta))/(2*a);
                Console.WriteLine("x1="+x1+" ve x2="+x2);
            }

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

## Şartların Birleştirilmesi

Şartların VE kelimesi ile birleştirilmesi için && operatörü, VEYA kelimesi ile birleştirilmesi için ise || operatörü kullanılır.

**NOT:** iki sayısal değer veya Tarih aralığı sorgulanacak ise aradaki operatör kesinlikle && olmalı. Bilgisayar ortamında tarih değerleri tamsayı olarak saklanır ve yeni tarih değeri her zaman eski tarih değerinden büyüktür. Tarihsel karşılaştırmalar buna göre yapılmalıdır.

Eğer(bugun>15.11.2011) ise ekranaYaz("bugün 15 Kasım 2011'den sonra!"); değil ise EkranaYaz("bugün 15 Kasım 2011 den önce!");

**NOT:**&& ve || operatörleri ile ikiden fazla koşulun da kontrolü yapılabilir. Ayrıca bu operatörler birlikte de kullanılabilir.

Örnek:

(koşul1&&koşul2&&koşul3&&..koşul\_N);//tüm koşullar birlikte sağlanırsa TRUE sonucu üretilir.

(koşul1||koşul2||koşul3||..koşul\_N);//koşullardan herhangi biri sağlanırsa TRUE sonucu üretilir.

Koşul1&&(koşul2||koşul3||(koşul4&&koşul5)&&koşul6);//karışık örnek

---

### Örnek (Yaşar, 2011, 73):

```
using System;
namespace or073
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 15;
            if (a >=10 && a <=50)
            {
                Console.WriteLine("a değeri 10 ve 50 aralığında!");
            }
            else
            {
                Console.WriteLine("a değeri 10 ve 50 aralığı dışında!");
            }
            Console.WriteLine("ÇIKIŞ İÇİN ENTER TUŞLAYINIZ..!");
            Console.ReadLine();
        }
    }
}
```



**Switch Yapısı**

İç içe çok sayıda if deyimi kullanmak yerine switch yapısı tercih edilebilir. Genellikle kullanıcıdan bir seçim yapması istendiğinde programdaki menü seçeneklerinin tercih edilmesi için kullanılır.

Bu yapı ile ilgili genel kurallar şöyle sıralanabilir:

- Sadece tam sayılar ve bazen char tipindeki değerler sorgulanabilir.
- Aralık belirlenemez.
- Hatalı kullanım olasılığı yüksektir. Default ve break sözcüklerinin kullanımı bazen unutulabilir.
- Kontrol edilen değer hiçbir seçenekte yok ise default sözcüğünden sonra yazılan işlemler icra edilir.
- Eğer break sözcüğü kullanılmaz ise program akışı sıradaki satırdan devam eder.

**Söz dizimi (syntax):**

```
Switch(Kontrol_Edilecek_Deger){
    Case Durum_Değeri_1: durum1 için yapılacak işlem. Birden fazla işlem var ise blok içinde yazılır.;break;
    Case Durum_Değeri_2: durum2 için yapılacak işlem. Birden fazla işlem var ise blok içinde yazılır.;break;
    .
    .
    .
    Case durum_değeri_N: durum_N için yapılacak işlem. Birden fazla işlem var ise blok içinde yazılır.;break;
    Default: kontrol edilen değer hiçbir durum değerine eşit değilse yapılacak işlem. Birden fazla işlem var ise blok içinde yazılır.;break;
```

**NOT:** default ve break sözcükleri kullanılmak zorunda değildir. Ancak break deyimi unutulursa takip eden case işlemleri de gerçekleştirilir. Break kelimesi case yapısından çıkmak için yazılır. Kontrol edilen switch kelimesi içindeki değer hiçbir case değerine eşit değil ise Default kelimesinden sonra yazılan program kodları işletilecektir.

**Örnek (Yaşar, 2011, 75):**

```
using System;
namespace or075
{
    class Program
    {
        static void Main(string[] args)
        {
            byte x = 2;
            switch (x)
            {
                case 1: Console.WriteLine("Pazartesi"); break;
                case 2: Console.WriteLine("Salı"); break;
                case 3: Console.WriteLine("Çarşamba"); break;
                case 4: Console.WriteLine("Perşembe"); break;
                case 5: Console.WriteLine("Cuma"); break;
                case 6: Console.WriteLine("Cumartesi"); break;
                case 7: Console.WriteLine("Pazar"); break;
                default: Console.WriteLine("Yanlış Giriş Yaptınız..!"); break;
            }
            Console.WriteLine("ÇIKIŞ İÇİN ENTER TUŞLA");
            Console.ReadLine();
        }
    }
}
```

**Örnek (Yaşar, 2011, 77):**

```

using System;
namespace or077
{
class Program
    {
static void Main(string[] args)
    {
char malzeme='Z';
byte kdv;
switch (malzeme)
    {
case 'A':
case 'B':
case 'C': kdv = 1; break;
case 'D':
case 'E':
case 'F': kdv = 8; break;
case 'G':
case 'H': kdv = 18; break;
default: kdv = 25; break;
    }
Console.WriteLine("ÜRÜN KATEGORİSİ=" + malzeme + " ve KDV'si= %" + kdv);
Console.WriteLine("ÇIKIŞ İÇİN ENTER TUŞLA");
Console.ReadLine();
    }
}

```

```

Geliri 2000 altında olanlara 400 ₺ burs, 2000-3000 olanlara 300 ₺ burs
if(gelir<2000)
    burs=400;
else if(gelir>=2000 && gelir <=3000)
    burs=300;
else
    E. yaz(Burs Yok)

```

## ?: Yapısı

if yapısı yerine kullanılabilir. Program kodunun okunabilirliğini arttırmak için tercih edilebilir. ?: operatörünün sonucu genellikle bir değişkene aktarılır!

Söz dizimi (syntax):

(koşul) ? (koşul doğru ise yapılacak işlem) : (koşul yanlış ise yapılacak işlem);

**Örnek (Yaşar, 2011, 78):**

```

using System;
namespace or078
{
class Program
    {
static void Main(string[] args)
    {
int a = 10, b;
    b=(a==10)?40:20;//a=10 ise b=40, değil ise b=20 yapar.
/* if(a==10)
b=40;
else
b=20;*/
/* switch(a)
    {
        * {
        * case 10: b=40; break;
        * default: b=20; break;
        * }
        * */
(a==10)?40:20;//Bu satırın tek başına bir anlamı yoktur! Üretilen değer kullanılmıyor!
Console.WriteLine("b="+b);
Console.WriteLine("ÇIKIŞ İÇİN ENTER TUŞUNA BASINIZ...!");
Console.ReadLine();
    }
}

```

Örnek (Yaşar, 2011, 78b):

```
/* Maaşı 1500 TL altında olanlara 500TL, 1500TL ve üzerinde olanlara 350TL ikramiye
ödenecek*/
using System;
namespace or078b
{
    class Program
    {
        static void Main(string[] args)
        {
            int maas = 1400, ikramiye;
            ikramiye = (maas < 1500) ? 500 : 350;
            Console.WriteLine("İKRAMİYE=" + ikramiye);
            Console.WriteLine("PROGRAMI SONLANDIRMAK İÇİN ENTER TUŞUNA BASINIZ..!"); //bu satır koşul
            doğru da olsa, yanlış ta olsa çalışır!
            Console.ReadLine();
        }
    }
}
```

---

Örnek (Yaşar, 2011, 78c):

```
using System;
namespace or078c
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 0;
            string s;
            s = (a < 0) ? "SAYI SIFIRDAN KÜÇÜK" : "SAYI SIFIR VEYA SIFIRDAN BÜYÜK";
            Console.WriteLine(s);
            Console.ReadLine();
        }
    }
}
```

---

Örnek (Yaşar, 2011, 78c1):

```
using System;
namespace or078c
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 1;
            //string s;
            Console.WriteLine(((a < 0) ? "SAYI SIFIRDAN KÜÇÜK" : "SAYI SIFIR VEYA SIFIRDAN BÜYÜK"));
            //?: operatörü iç içe de kullanılabilir.
            //Console.WriteLine((a < 0) ? "SAYI SIFIRDAN KÜÇÜK" : ((a == 0) ? "SAYI SIFIR" : "SAYI SIFIRDAN
            BÜYÜK"));

            Console.ReadLine();
        }
    }
}
```

---

## DÖNGÜLER (LOOPS)

Ders kitabı: Yaşar, ss: 80 - 96

Belirli işlemleri, belirli bir koşul veya koşullar sağlandığı sürece tekrarlamak için döngüler kullanılır. Döngüler yapılarına göre sabit ve şartlı olmak üzere ikiye ayrılır.

- Sabit Döngüler
  - For Döngüsü
  - ForEach Döngüsü
- Şartlı Döngüler
  - While döngüsü
  - Do-While döngüsü

### Sabit Döngüler

Döngü tekrar sayısı programcı tarafından baştan belirlenir.

#### For Döngüsü

Söz dizimi(syntax):

For(döngü\_değişkeni\_başlangıç\_değeri;koşul;döngü\_değişkeni\_değişim\_miktarı)

{ işlemler;

...

...

}

**Örnek (Yaşar, 2011, 82):**

*/\* 1'DEN 10'A KADAR OLAN TAMSAYILARI EKRANA YAZAR \*/*

using System;

namespace or082

{

class Program

{

public static void Main(string[] args)

{

int a;

for(a=1;a<=10;a++){

Console.WriteLine(a);

}

Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . .");

Console.ReadKey(true);

}

}

}

**Örnek (Yaşar, 2011, 82b):**

*/\* 90'DAN 100'E KADAR OLAN TAMSAYILARI EKRANA tersten ve ikişer YAZAR \*/*

using System;

namespace or082b

{

class Program

{

public static void Main(string[] args)

{

int a=100;

for (;;)

{

Console.WriteLine(a);

if (a == 90) break;

a -= 2;

}

Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . . ");

```

Console.ReadKey(true);
    }
}
}

```

**Örnek (Yaşar, 2011, 83):**

```

using System;
namespace or083
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a,b=0;
            for(a=1;a<=10;a++){
                b++;
            }
            Console.WriteLine("Döngü "+b+" kere çalışmıştır!");
            Console.WriteLine("Döngü değişkeninin son değeri a= "+a);
            Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . ");
            Console.ReadKey(true); } }

```

**Ödev:** Öyle bir program yazın ki kullanıcının klavyeden girdiği sayıları tolasın, kullanıcı klavyeden 83 girerse toplama işlemini bitirsin ve sonucu ekrana yazsın.

```

50
100
300
83
Toplam=450

```

**Ödev:** Öyle bir program yazın ki kullanıcıdan bir sayı istesin. Daha sonra bu sayı ile 1 arasındaki TEK veya ÇİFT tamsayıların toplamını hesaplayarak sonucu ekrana yazsın. Kullanıcıya TEK veya ÇİFT tamsayılar ile ilgili işlem yapılacağını sorsun?

Bir sayı girin:

TEK'leri mi ÇİFT'leri mi toplamak istersiniz?

**Ödev:** Öyle bir program yazın ki kullanıcıdan İKİ sayı istesin. Daha sonra bu İKİ arasındaki TEK veya ÇİFT tamsayıların toplamını hesaplayarak sonucu ekrana yazsın. Kullanıcıya TEK veya ÇİFT tamsayılar ile mi ilgili işlem yapılacağını sorsun?

İKİ sayı girin:

TEK'leri mi ÇİFT'leri mi toplamak istersiniz?

### **ForEach Döngüsü**

Bu döngü de for döngüsüne bezemekle beraber, döngü değişkeninin sıralı olmayan değerler almasına da izin verilir. Genellikle diziler ile birlikte kullanılır.

**Söz dizimi(syntax):**

ForEach(DöngüDeğişkeni in Dizi)

```

{...;
    işlemler;
....;
}

```

**NOT:** döngü değişkeni veri tipi ile dizi veri tipi aynı olmalıdır!

**Örnek (Yaşar, 2011, 85):**

```
using System;
namespace or085
{
    class Program
    {
        public static void Main(string[] args)
        {
            double []A={4,9,145,744,14,3.5};
            double toplam=0;
            foreach(double U in A){
                toplam+=U;
            }
            Console.WriteLine("Toplam = "+toplam);
            /* for (int sayac = 0; sayac < 6; sayac++)
            {
                toplam += A[sayac];
            }*/
            Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . ");
            Console.ReadKey(true);
        } } }
```

---

### Koşullu Döngüler

Belirli bir koşul sağlanana kadar veya sağlandığı sürece devam eden döngülerdir. Genellikle tekrar sayısı belli olmayan döngüler için kullanılırlar. Şartlı döngüler ikiye ayrılır:

- Koşulu başta verilen (While)
- Koşulu sonda verilen(do-while)

#### *Koşulu Başta Olan Döngü (While)*

Koşulun sağlanıp sağlanmadığına en başta bakılır ve koşul sağlanıyorsa döngü bloğu çalıştırılır, sağlanmıyorsa hiç çalıştırılmaz. Her bir tekrar sonunda koşul yeniden kontrol edilir.

Söz dizimi(syntax):

While(koşul)

```
{...;
İşlemler;
...;
}
```

**Not:** koşul içerisindeki değişkenin değeri ya döngü içerisinde otomatik olarak değiştirilerek, ya da elle değiştirilerek döngüden çıkış sağlanır.

---

**Örnek (Yaşar, 2011, 88):**

//Döngü doğal yolla sonlandırılacak!

using System;

namespace or088

```
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a=0;
            while(a<10){//döngü a değeri ondan küçük olduğu sürece çalışacak
                a++;
                Console.WriteLine(a);//ekrana yazılacak ilk değer 1 olacak!
            }
            Console.WriteLine("Döngü Dışı");
            Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Örnek (Yaşar, 2011, 89):**

//1-10 arasındaki tamsayıların toplamını hesaplayıp ekrana yazan program!

//Döngü el ile sonlandırılacak!

using System;

namespace or089

```
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a=0, b=0, toplam=0;
            while(b==0){
                a++;
                toplam=toplam+a;
                if(a==10){b=2;}
            }
            Console.WriteLine("Toplam = "+toplam);
            Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Örnek:** İkili sayı sistemindeki bir değeri onlu sayı sistemine dönüştüren program.

```
static void Main(string[] args)
{
    int sayi, ikiliDeger, onluDeger = 0, tabanDeger = 1, basamakDeger;
    sayi = 1010111001;
    ikiliDeger = sayi;

    while (sayi > 0)
    {
        basamakDeger = sayi % 10;
        onluDeger = onluDeger + basamakDeger * tabanDeger;
        sayi = sayi / 10;

        tabanDeger = tabanDeger * 2;
    }
    Console.WriteLine("İkili (Binary) Sayı: " + ikiliDeger);
    Console.WriteLine("\nOnlu (Decimal) Sayı: " + onluDeger);
    Console.ReadLine();
}
```

**Ödev:** Yukarıdaki örneği ikili sayı sistemindeki değeri klavyeden girilecek şekilde yeniden düzenleyiniz.



**Koşulu Sonda Olan Döngü (Do - While)**

Koşul döngü bloğunun en sonunda bulunur. Bu yüzden döngü bloğu en az bir kez çalışır.

Söz dizimi(syntax):

Do

{ ...;

İşlemler;

...;

} While(koşul);

**Not:** Do – While döngüsünde de koşul içerisindeki değişkenin değeri ya döngü içerisinde otomatik olarak değiştirilerek, ya da elle değiştirilerek döngüden çıkış sağlanır.

---

**Örnek (Yaşar, 2011, 91):**

//1-10 arasındaki sayıları ekrana yazan program!

//Döngü doğal olarak sonlandırılacak!

using System;

namespace or091

```
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a=0;
            do{
                a++;
                Console.WriteLine(a);
            }while(a<10);
            Console.WriteLine("Döngü Dışı...");
            Console.WriteLine("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . .");
            Console.ReadKey(true);    } } }
```

---

**Örnek (Yaşar, 2011, 92):**

//1-10 arasındaki tamsayıların toplamını hesaplayıp ekrana yazan program!

//Döngü el ile sonlandırılacak!

using System;

namespace or092

```
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a=0, b=0, toplam=0;
            do{
                a++;
                toplam=toplam+a;
                if(a==10){b=2;}
            }while(b==0);
            Console.WriteLine("Toplam = "+toplam);
            Console.WriteLine("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . .");
            Console.ReadKey(true);
        } } }
```

---

**ÖDEV:** Öyle bir program yazın ki, önce kullanıcıya kaç adet sayı girmek istediğini sorsun. Daha sonra kullanıcıdan sayı girmesini istesin. Kullanıcı fark etse de etmese de baştan girdiği adet kadar sayı tamamlandığında bu sayıların toplamını ve ortalamasını hesaplayarak ekrana yazsın.

**NOT:** Tüm döngüler iç içe olabilir. İçteki döngü sonlanmadan dıştaki döngü devam etmez. Yani döngüler birbirini kesmez.

### Döngü Denetimi

Döngülerin her zaman sonuna kadar devam etmesi gerekmez. Bazen de döngü bloğundaki tüm işlemler tamamlanmadan döngü bir sonraki tekrara başlamalıdır. Bunları sağlayan ki temel sözcük vardır: Break ve Continue.

### Break Sözcüğü

Döngü bloğu içerisinde istenen elde edildikten sonra döngünün sonlandırılması, diğer döngü aşamalarının (çevrimlerinin) gerçekleştirilmemesi amacıyla kullanılır.

Söz dizimi(syntax):

Break;

---

### Örnek (Yaşar, 2011, 94):

```
//b değerinin asal sayı olup olmadığını bulan program!
using System;
namespace or094
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a, b = 11;
            String s = "b Sayısı Asal Sayı";
            if (b == 1) { Console.WriteLine("ASAL DEĞİL!"); goto bitir;}
            for (a = 2; a <= b / 2; a++)
            {
                if (b % a == 0)
                {
                    s = "b Sayısı Asal Sayı Değil!";
                    break;
                }
            }
            Console.WriteLine(s);
            bitir: Console.Write("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . . ");
            Console.ReadKey(true);
        }
    }
}
```

**ÖDEV:** Yukarıdaki örneği b değeri kullanıcı tarafından girilecek şekilde düzenleyiniz.

**Continue Sözcüğü**

Döngü bloğundaki işlemlerin tamamlanmadan bir sonraki döngü çevrimine geçmek gerektiğinde kullanılır. Örneğin geliri sınıf ortalama gelirinin altında olan öğrencilere burs verilecek ise ve sınıfta 50 öğrenci var ise, geliri sınıf ortalamasının üstünde olan bir öğrenciye sıra geldiğinde onunla ilgili hiçbir işlem yapmadan sıradaki öğrenciye bakılmalıdır.

**Örnek (Yaşar, 2011, 96):**

//aldığı burs 500TL altında olanlara %10 ve +50 TL sabit artış yapan program!

//burs miktarları a dizi değişkeninde saklanıyor!

using System;

namespace or096

```
{
    class Program
    {
        public static void Main(string[] args)
        {
            double[] a={150,600,700,450,850,275,950,1500,210,368};
            int b;
            for (b=0;b<=9;b++){ //foreach(double d in a){
                if(a[b]>=500)
                { //burs miktarı 500 ve üzeri olanlara artış yapılmıyor!
                    Console.WriteLine(a[b]); //a[1]=600, a[2]=700, a[4]=850, a[6]=950, a[7]=1500,
                    continue; //döngünün aşağıdaki satırları çalışmadan yeni çevrime gider!
                }
                a[b]=(a[b]*0.1)+a[b];
                a[b]=a[b]+50; //a[0]=215, a[3]=545, a[5]=352,5, a[8]=281, a[9]=454,8
                Console.WriteLine(a[b]); //215,545,352.5,281,454.8
            }
            Console.WriteLine("DEVAM ETMEK İÇİN BİR TUŞA BASINIZ. . .");
            Console.ReadKey(true);
        }
    }
}
```

215
600
700
545
850
352,5
950
1500
281
454,8

**ÖDEV:** Örnek096'yı foreach ile yeniden yazınız.

**ÖDEV:** for/while/do-while ile yaptığınız tüm örnekleri bu kelimeleri kullanmadan goto-label kelimelerini kullanarak yapmaya çalışınız.

**DİZİLER ve MATRİSLER (Ders kitabı sf: 196)**

Bellekte ardışık olarak yer kaplayan veri kümesine dizi denmektedir. Aynı tipte çok sayıda değişken kullanmak yerine diziler kullanılabilir.

Değişken yalnızca bir bellek adresini gösterirken, dizi ise birden çok bellek adresini gösterir.

Bir dizinin elemanlarını gösteren indisler **[]** işaretleri arasında yazılır ve sıfırdan başlarlar.

Değişken gösterimi	X	Y	Z	T	K	Değişken veya dizinin her bir elemanı ne kadar byte'lık yer kaplıyorsa her bir bellek gözü de o kadar düşünülebilir.
Veri bellek gözleri	25	300	1209	0	45	
Dizi yapısı ile gösterim	A[0]	A[1]	A[2]	A[3]	A[4]	

Dizi başlangıç adresi.  
Örneğin 100.

Dizinin üçüncü elemanının (1209) başlangıç adresi. Her bir eleman 2 byte yer kaplıyorsa başlangıç adresinden  $2 \times 2 = 4$  byte uzaklıktaki veri.

Diziler tek boyutlu olabilecekleri gibi çok boyutlu da olabilirler. Dizinin herhangi bir elemanı kaç indis ile gösterilirse dizi o kadar boyutludur.  $A[X,Y] \rightarrow A$  dizisi iki boyutludur.

Örneğin iki boyutlu diziler resim işlemede kullanılabilirler. Bir resmin tüm piksel değerleri iki boyutlu bir dizinin içinde saklanabilir.

Yine örneğin satış elemanlarının bölgelere göre yaptıkları satışlar da iki boyutlu bir dizide saklanabilir. Eğer üç bölge ve üç satış elemanı var ise ve her bir satış elemanı her bölgede satış yapıyorsa, **satış elemanlarının satır başlıklarında, bölgelerin ise sütun başlıklarında** temsil edildiği göz önüne alındığında buna ilişkin matris şöyle düzenlenebilir:

A[ ]	0. sütun Manisa	1. sütun İzmir	2. sütun Aydın
0. satır Ali	780	790	800
1. satır Veli	200	300	600
2. satır Fatma	400	350	900

Her bir elemanı tamsayı olan A dizisinin yukarıdaki gibi tanımlandığı düşünülürse 2. Satış elemanının 2. Bölgede yaptığı satış miktarı 300 değeridir ve bilgisayarda bu değer **A[1,1]** ile ifade edilir. 2. Satış elemanının 3. Bölgede yaptığı satış ise **A[1,2]=600** ifadesi ile erişilebilir.

İki boyutlu dizilerde birinci indis satırlar, ikinci indis ise sütunları ifade eder. Satırlar yataydaki, sütunlar ise düşeydeki bloklardır.

Bilgisayardaki dizi kavramının matematikteki karşılığı **matrislerdir**. Matrisler daha çok resim işleme uygulamalarında kullanılırlar. Resmi belleğe almak, efektler vermek, döndürmek gibi işlemlerde matrisler kullanılır. Bunun dışında matematikçiler için lazım olan bir matrisin tersi, transpozu, matrislerin çarpımı gibi tüm problemlerin çözümünde kullanılırlar. Matrisler ayrıca lineer denklem sistemlerinin çözümünde de kullanılırlar.

Dizi tanımlama söz dizimi:

**1. Yol:** eleman sayısı belirli ve fakat elemanlar belirsiz. Köşeli parantez kesinlikle dizi adından önce olmalı.

Dizi\_Tipi [ ]Dizi\_Adi=new Dizi\_Tipi[Eleman\_Sayısı];

int [ ]A=new int[10];

double [ ]X=new double[100];

string [ ]S=new String[40];

char [ , , ]Harf=new char[3,4,5]; //3x4x5=60 elemanlı iki boyutlu char tipinde harf adında dizi.

int [ , ] b=new int[10,10]; //10x10=100 elemanlı iki boyutlu int tipinde b adında dizi.

**2. Yol:** Eleman sayısı ve eleman değerleri önceden yazılan durum. Dizinin bellek gözlerine başlangıç değerleri atanıp bu değer sayısı kadar eleman sayısının ima edildiği tanımlama şekli.

int [ ] D={1,2,3,4,5}; //int [ ] D=new int[5];

string [ ] isim={"Ali","Veli","Ahmet","Mehmet"};

char [ ] c={'a','b','c'};

int [ , ] K={{1,2,3},{4,5,6}}; //K[1,1]=5; //int [ , ] K=new int[2,3];

K[ ]	0	1	2
0	1	2	3
1	4	5	6

**Dizilerin Kullanımı**

Dizileri kullanırken dikkat edilecek hususlar şunlardır:

1. Diziler belleğe toplu bir şekilde yerleştiklerinden tanımlanan bir dizinin eleman sayısı programın bellekte kapladığı alana doğrudan etki edecektir. Dizi boyutunu belirlerken ince eleyip sık dokumak gerekmektedir.
2. Elemanları gösteren sayılar (indisler) sıfırdan başlayıp, eleman sayısının bir eksiğine kadar devam eder. Bu alt ve üst sınırlara azami dikkat edilmesi gerekmektedir.
3. Dizilere eleman atarken ve dizinin elemanlarını kullanırken dizi boyutu sayısınca iç içe for döngüsü kullanma gereği hemen akla gelmelidir. Örneğin iki boyutlu dizilere değer atamak için, iki for döngüsü iç içe kullanılmalıdır.

**Örnek (Yaşar, 2011, 200):**

```
using System;
namespace or200
{
    class Program
    {
        static int sayac,mayac;
        public static void Main(string[] args)
        {
            int [] A=new int[10];
            String [] S=new string[40];
            int[,] sayi={{1,2,3},{4,5,6}};
            int [,] R=new int[10,10];
            for(sayac=0;sayac<10;sayac++)
            {
                Console.Write("A dizisinin "+(sayac+1)+". elemanını giriniz:");//12
                A[sayac]=Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("A Dizisinin "+(sayac+1)+". Elemanı olarak "+A[sayac]+" girdiniz!");
            }
            for (sayac=0;sayac<2;sayac++) //satırları sayar
                for(mayac=0;mayac<3;mayac++) //sütunları sayar
                {
                    Console.Write("sayi dizisinin "+sayac+" . satır ve "+mayac+" . sütun değerini giriniz:");
                    sayi[sayac,mayac]=Convert.ToInt32(Console.ReadLine());
                }
            Console.WriteLine("\nSayı Dizisi Aşağıdaki Gibidir:");
            for (sayac=0;sayac<2;sayac++)
                for(int mayac=0;mayac<3;mayac++)
                {
                    Console.Write(sayi[sayac,mayac]+"\\t");
                    if (mayac==2) Console.WriteLine("\\n");
                }
            A[0]=123;
            A[9]=A[0]+199;//A[9]=322
            S[2]="Emin";
            S[0]="Celal Bayar Üniversitesi";
            S[1]="Kırkağaç Meslek Yüksekokulu";
            R[3,4]=55;
            Console.WriteLine("\\n\\n\\n"+A[9]);
            Console.WriteLine(S[0]);
            Console.WriteLine(S[1]);
            Console.WriteLine(S[2]);
            Console.WriteLine(R[3,4]);
            Console.WriteLine(sayi[1,2]);
            Console.ReadLine();    }    } }
```

A.Lenght  
(Fonksiyon sayısı)

**Dizilere Ait Metotlar****Dizinin Eleman Sayısı:**

Dizi\_adi.Length

**Diziyi Başka Bir Diziyeye Kopyalamak:**

Kaynak\_Dizi.CopyTo(Hedef\_Dizi,Hedef\_Dizinin\_Kopya\_Başlangıç\_indisi);

Burada kaynak dizinin tamamı hedef dizinin belirtilen başlangıç indisinden itibaren hedef diziyeye kopyalanır. Dikkat edilmesi gereken nokta hedef dizi en az kaynak dizi kadar eleman sayısına sahip olmalıdır ve aynı zamanda hedef dizinin belirtilen başlangıç indisinden itibaren kaynak dizi tamamen sığmalıdır.

**Örnek: dizi alt metotları**

```
using System;
namespace diz_alt_metot
{
    class Program
    {
        static int sayac,mayac;
        public static void Main(string[] args)
        {
            int [] A={1,2,3,4,5};
            Console.WriteLine("A[] dizisinin Eleman Sayısı: "+A.Length);//5
            int [] B=new int[10];
            A.CopyTo(B,5); //buradaki 5 değerini 6 veya 7 yaparak yeniden deneyiniz. Sonucu yorumlayınız.
            Console.WriteLine("B dizisinin 7. elemanı="+B[6]);
            Console.ReadLine();
        }
    }
}
```

**Örnek (Yaşar, 2011, 201):**

```
/* Elemanları dışarıdan girilen bir dizinin 3. elemanı ile 7. elemanını toplayıp sonucu ekrana yazan program */
using System;
namespace or201
{
    class Program
    {
        public static void Main(string[] args)
        {
            int [] B=new int[10];
            int a=0,top=0;
            foreach(int al in B)
            {
                Console.Write(a+". Elemanı Gir: ");
                B[a]=Int32.Parse(Console.ReadLine());
                a++;
            }
            top=B[2]+B[6];
            Console.WriteLine("Sonuç="+top);
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek (Yaşar, 2011, 202):**

// 5 elemanlı bir X dizisi içinde son rakamı sıfır olan kaç tane sayı olduğunu hesaplar

using System;

namespace or202

{

class Program

{

public static void Main(string[] args)

{

int [] B=new int[5];

int a=0,say=0;

foreach(int al in B)//for (a=0; a&lt;B.Length;a++)

{

Console.Write(a+" Elemanı Gir: ");

B[a]=Int32.Parse(Console.ReadLine());

if(B[a]%10==0) say++;

a++; //döngü for ile olsaydı a++ satırına gerek kalmayacaktı!

}

Console.WriteLine("Son Rakamı Sıfır Olan Sayı Adedi: "+say);

Console.Write("Press any key to continue . . . ");

Console.ReadKey(true);

}

}

}

**ÖDEV:** Yukarıdaki örneği son rakamı sıfır olan değerlerin toplamı ve ortalamasını da verecek şekilde yeniden yaz.

**Örnek (YAŞAR, 2011, 204):**

// 8 elemanlı ve elemanları dışarıdan girilen bir dizide yine dışarıdan girilen bir ismin ilk olarak kaçınıcı sırada bulunduğunu ve yoksa bulunmadığını belirtir

```
using System;
namespace or204
{
    class Program
    {
        public static void Main(string[] args)
        {
            SOR: string [] Y=new String[8];
            int a=0; string iletı="Aranan Değer Bulunamadı!";
            string hangi;
            ConsoleKeyInfo devam;
            Console.WriteLine("\nHangi ismi arıyorsunuz? ");
            hangi=Console.ReadLine();
            foreach(string s in Y)
            {
                Console.WriteLine(a+". İsmi Gir: ");
                Y[a]=Console.ReadLine();
                if(Y[a]==hangi) {iletı="İlk olarak baştan "+(a+1)+" . Sırada";break;}
                a++;
            }
            Console.WriteLine(iletı);
            SONMU: Console.WriteLine("\nBaşka Arama Yapacak Mısınız? (E/H)");devam=Console.ReadKey();
            if(devam.KeyChar=='E' || devam.KeyChar=='e') goto SOR;
            else if (devam.KeyChar=='H' || devam.KeyChar=='h') goto SON;
            else {Console.WriteLine("\nEvet için E, hayır için H tuşlayınız..!"); goto SONMU;}
            SON: Console.WriteLine("\nPress any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek (YAŞAR, 2011, 205):**

//10 elemanlı ve elemanları 10-50 arasında rastgele tamsayılardan oluşan bir dizinin elemanlarını alt alta yazdırır

```
using System;
namespace or205
{
    class Program
    {
        public static void Main(string[] args)
        {
            Random rastgele=new Random();//Random sınıfından rastgele isimli yeni bir nesne bildiriliyor.
            int a;
            int[] B= new int[10];
            for(a=0;a<B.Length;a++){
                B[a]=rastgele.Next(10,51);//10 – 50 arasında rastgele sayı üretilir ve B[a] ya atanır.
                Console.WriteLine(a+". Sayı = "+B[a]);
            }
            Console.WriteLine("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```



**Örnek (YAŞAR, 2011, 207):** Eleman sayısı kullanıcı tarafından belirleniyor!

// Öğrenci sayısı kadar elemana sahip diziye değer alır ve ilk değeri ekrana yazar

```
using System;
namespace or207
{
    class Program
    {
        public static void Main(string[] args)
        {
            int b,c;
            Console.Write("Girilecek Öğrenci Sayısı: ");//27
            b=Int32.Parse(Console.ReadLine());
            String [] A=new String[b];
            for(c=0;c<=A.Length-1;c++)
            {Console.Write(c+". Öğrenci Adı: ");
              A[c]=Console.ReadLine();}
            Console.Write("Girilen İlk Öğrenci : "+A[0]);
            Console.Write("\nPress any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Örnek (YAŞAR, 2011, 208):**

// 10 Elemanlı ve elemanları dışarıdan girilen bir dizinin 100-200 arasında olmayan elemanlarını %10 arttırır

```
using System;
namespace or208
{
    class Program
    {
        public static void Main(string[] args)
        {
            int c;
            double [] A= new Double[10];
            for (c=0; c<A.Length;c++)//A dizisinin tüm elemanlarına sırayla klavyeden değer alınıyor.
            {
                Console.Write(c+". Sayı : ");
                A[c]=Int32.Parse(Console.ReadLine());
            }
            for(c=0;c<=A.Length-1;c++){//A dizisinin 100 – 200 arasında olmayan elemanları %10 arttırılıyor.
                if (A[c]<100 || A[c]>200) //if( !(A[c]>=100 && A[c]<=200))
                {
                    A[c]=A[c]*1.10;
                }
                Console.WriteLine(c+". Sayı "+A[c]);
            }
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Örnek (YAŞAR, 2011, 209): (Yer değiştirme (takas) algoritması )**

//Elemanları dışarıdan girilen 10 elemanlı A dizisinin elemanlarını tersten yerleştirir.

using System;

namespace or209

```

{
    class Program
    {
        public static void Main(string[] args)
        {
            int c, gecici;
            int [] A=new int[10];
            for(c=0;c<=A.Length-1;c++){// Eleman girişi
                Console.Write(c+" Sayı: ");
                A[c]=int32.Parse(Console.ReadLine());
            }
            for(c=0;c<=(A.Length-2)/2;c++){//TersÇevir
                gecici=A[(A.Length-1)-c];
                A[(A.Length-1)-c]=A[c];
                A[c]=gecici;
                Console.WriteLine("A["+c+"]="+A[c]);
            }
            for(c=0;c<=A.Length-1;c++){ //Ekran yazdırma
                Console.WriteLine(A[c]);
            }
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}

```

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
	1	2	3	4	5	6	7	8	9	10
	10	2	3	4	5	6	7	8	9	1
	10	9	3	4	5	6	7	8	2	1
	10	9	8	4	5	6	7	3	2	1
	10	9	8	7	5	6	4	3	2	1
	10	9	8	7	6	5	4	3	2	1

**Örnek (YAŞAR, 2011, 211) (En büyük bulma algoritması!)**

10 Elemanlı A dizisinde isimler, B dizisinde de bilgisayar sınav notları vardır. A'nın sıfırıncı elemanındaki ismin bilgisayar notu B dizisinin sıfırıncı elemanındadır. En yüksek notu alan kişinin adını ekrana yazdırınız.

/\* B dizisinde en büyük notun sırası A dizisindeki isme karşılık geleceğinden amaç

\* B dizisindeki en yüksek notun sırasını öğrenmektir. \*/

using System;

namespace or211

```

{
    class Program
    {
        public static void Main(string[] args)
        {
            int c, EnBuyuk=0, yer=0;
            String [] A={"Emin", "Emir", "İclal", "Bekir", "Cem", "Arda", "Kaan", "Metin", "Can", "Ramiz"};
            int [] B={56,78,95,23,95,87,61,77,45,33};
            for (c=0;c<=B.Length-1;c++){
                if(B[c]>= EnBuyuk){
                    EnBuyuk=B[c];
                    yer=c; }
            }
            Console.WriteLine("En Yüksek Notu Alan Kişi: "+A[yer]);
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}

```

**ÖDEV:** en yüksek notu alan birden fazla kişi varsa onların da isimlerini yazdırınız.

**Örnek (YAŞAR, 2011, 213): (Sıralama algoritması)**

// 10 Elemanlı bir dizinin elemanlarını küçükten büyüğe doğru sıralayınız.

```

using System;
namespace or213
{
    class Program
    {
        public static void Main(string[] args)
        {
            int i,j,gecici,EnKucukYer;
            int []A={78, 56, 23, 95 ,68,87,61,77,45,33};
            for(i=0; i<=A.Length-1;i++)
            {
                EnKucukYer=i;//1-1. EnKucukYer=0, EnKucukYer=1
                for(j=i+1;j<=A.Length-1;j++){//2-1. J=1, j=2, j=3, j=4, j=5, j=6, j=7, j=8, j=9—2-2 j=2
                    if(A[j]<A[EnKucukYer]) EnKucukYer=j;//2-1 enkucukyer=1, enkucukyer=2,
                }
                gecici=A[i];//1.1. gecici=78
                A[i]=A[EnKucukYer];//1-1 A[0]=23
                A[EnKucukYer]=gecici;//1-1 A[2]=78 //{23,56,78,95,68,87,61,77,45,33}
                Console.WriteLine(A[i]);// 23
            }
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        } } }

```

0	1	2	3	4	5	6	7	8	9
78	56	23	95	68	87	61	77	45	33
23	56	78	95	68	87	61	77	45	33
23	33	78	95	68	87	61	77	45	56
23	33	45	95	68	87	61	77	78	56
23	33	45	56	68	87	61	77	78	95
23	33	45	56	61	87	68	77	78	95
23	33	45	56	61	68	87	77	78	95
23	33	45	56	61	68	77	87	78	95
23	33	45	56	61	68	77	78	87	95

**ÖDEV:** Programı, diziyi büyükten küçüğe yerleştirecek şekilde yeniden yazınız.**Sıralama algoritmalarını toplu bir şekilde izlemek için:** [www.sorting-algorithms.com](http://www.sorting-algorithms.com)<https://www.toptal.com/developers/sorting-algorithms>**Örnek (Yaşar, 2011, 214):**

// 1-99 arasındaki sayıları yazı ile yazdırır.

```

using System;
namespace or214
{
    class Program
    {
        public static void Main(string[] args)
        {
            byte a;
            string s=" ";
            string [] Birler={"Bir","İki","Üç","Dört","Beş","Altı","Yedi","Sekiz","Dokuz"};
            string [] Onlar={"On","Yirmi","Otuz","Kırk","Elli","Altmış","Yetmiş","Seksen","Doksan"};
            Console.Write("Bir Sayı Giriniz: ");//72 – 80 – 6
            a=Byte.Parse(Console.ReadLine());
            if(a/10>=1){ //Sayı iki veya daha fazla basamaklı mı?
                s=Onlar[(a/10)-1];s= yetmiş – seksen -
            }
            if (a%10!=0){ //Sayının son basamağı sıfır değilse yaz
                s=s+Birler[(a%10)-1];s= yetmiş+iki – “ altı”

            }

            Console.WriteLine(s);//yetmişiki – seksen – “ altı”
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        } } }

```

**ÖDEV:** Programda önce girilen sayının en çok iki basamaklı ve 1 – 99 arasında pozitif sayı olduğunu kontrol ettiriniz. Eğer girilen sayı gerekli şartı sağlamıyor ise programın “Uygun değer girmediniz” mesajı ile sonlanmasını sağlayınız.

**ÖDEV:** Yukarıdaki programı 3 basamaklı sayıları da ekrana yazıyla yazacak şekilde düzenleyiniz.

---

**Örnek (YAŞAR, 2011, 216):**

// 10 Elemanlı bir A dizisi içerisindeki sayılardan tek olanları B dizisinin üst kısmına, çift olanları alt kısmına yerleştirir.

using System;

namespace or216

```
{
    class Program
    {
        public static void Main(string[] args)
        {
            int a, t=0, c=9;
            int []A={45,26,66,12,13,48,84,53,69,91};
            int []B=new int[10];
            for (a=0;a<=9;a++){
                if(A[a]%2==0){//çift mi diye bakılıyor
                    B[c]=A[a];
                    c--;}
                if(A[a]%2==1){//tek mi diye bakılıyor
                    B[t]=A[a];
                    t++;}
            }
            for(a=0;a<=9;a++){
                Console.WriteLine(B[a]);
            }
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Örnek (YAŞAR, 2011, 218):**

7 öğretmenin adı "isim" dizisinde ve 7 şehir adı "il" dizisinde bulunmaktadır. Bu 7 öğretmeni rastgele her bir şehre bir öğretmen olmak üzere atayan programı yazınız.

```
using System;
namespace or218
{
    class Program
    {
        public static void Main(string[] args)
        {
            int sayi,a;
            string[]il={"Muş","Ağrı","Bolu","Mersin","Ankara","Sivas","Kayseri"};
            string[]isim={"Asuman","İclal","Emin","Bekir","Ahmet","Can","Esmâ"};
            bool[]D=new bool[7];
            for(a=0;a<=6;a++) Console.WriteLine("Bool D dizisinin "+a+" . elemanı= "+D[a]);
            Random rastgele=new Random();
            for(a=0;a<=6;a++){
                do{ //bu döngü ve d[sayi]=true satırı ile bir şehre birden fazla öğretmen atanması engelleniyor.
                    sayi=rastgele.Next(0,7);//sıfır dahil, 7 dahil değil.
                }while(D[sayi]==true);
                D[sayi]=true;
                Console.WriteLine(isim[a]+"\\t= "+il[sayi]);
            }

            for(a=0;a<=6;a++) Console.WriteLine("Bool D dizisinin "+a+" . elemanı= "+D[a]);
            Console.WriteLine("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

Next(6) = 0dan 6 ya kadar

**ÖDEV:** Yukarıdaki programı öğretmenlerin de sırayla değil, rastgele seçileceği şekilde yeniden düzenleyiniz.

**Örnek (YAŞAR, 2011, 220):**

Elemanları dışarıdan girilen 10 elemanlı bir B dizisinin ilk elemanına değer olarak en yakın ikinci elemanı ekrana yazdıran program.

**İpucu:** En yakın elemanı bulmak için ilk eleman ile diğer tüm elemanlar sıra ile birbirinden çıkarılır ve sonucun mutlak değeri alınarak en küçük sonucun bulunduğu noktadaki indis değeri en yakın elemanın indis değeri olarak değerlendirilir.

```
using System;
namespace or220
{
    class Program
    {
        public static void Main(string[] args)
        {
            int sayi, a, e=1;
            int []B={36,78,18,41,165,75,55,40,125,91};
            for(a=2;a<B.Length;a++)
            {
                if(Math.Abs(B[0]-B[a])<Math.Abs(B[0]-B[e])) e=a;
            }
            Console.WriteLine("İlk Sayıya En Yakın Sayı = "+B[e]);
            Console.WriteLine("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Ödev:** Yukarıdaki programı, dizi değerleri dışarıdan girilecek şekilde düzenleyiniz.

**Örnek (YAŞAR, 2011, 221):** 1 – 10 arasında 1000 adet tamsayı üreterek bu sayıların hangisinden kaç adet üretildiğini ekrana yazan program.

**İpucu:** 1000 çevrimlik bir döngü kurduktan sonra her çevrimde üretilen rastgele sayının bir eksiği dizi indis değeri olarak kullanılarak o dizi elemanının değeri bir artırılır. Sonuçta dizinin her bir elemanının son değeri üretilen sayı adedini verir.

```
using System;
namespace or221
{
    class Program
    {
        public static void Main(string[] args)
        {
            int sayi, a, toplam=0;
            int []B=new int[10]; //üretilen sayıların adedi bu dizide tutulacak!
            for(a=0;a<=9;a++) Console.WriteLine("B["+a+"] = "+B[a]);
            Random rastgele=new Random();
            for(a=1;a<=1000;a++) // Bin adet rastgele sayı üretiliyor
            {
                sayi=rastgele.Next(1,11);
                B[sayi-1]++; //üretilen sayıya göre adedi bir artırılıyor.
            }
            for(a=0;a<=9;a++) //hangi sayıdan kaç adet üretildiği ekrana yazılıyor!
            {
                Console.Write("\n"+(a+1)+"den\t= "+B[a]+" adet");
                toplam += B[a];
            }
            Console.WriteLine("\n\nÜretilen Toplam Sayı Adedi= " + toplam);
            Console.Write("\nPress any key to continue . . .");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek (YAŞAR, 2011, 223):** EBUBEKİR isimindeki karakterleri rastgele olarak ekrana yazan program.

```
using System;
namespace or223
{
    class Program
    {
        public static void Main(string[] args)
        {
            int sayi,a;
            char[] isim={'E','B','U','B','E','K','İ','R'};
            bool []D=new bool[8];
            Random rastgele=new Random();
            for(a=0;a<=7;a++)
            {
                do{
                    sayi=rastgele.Next(8); //sıfır ile yedi arasında rastgele sayı üretiliyor.
                }while(D[sayi]==true);
                D[sayi]=true;
                Console.Write(isim[sayi]); //EBEK
            }
            Console.WriteLine("\n\nPress any key to continue . . .");
            Console.ReadKey(true);
        }
    }
}
```

BOOL [D=NEW BOOL[8]							
D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]
TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE

**Örnek (YAŞAR, 2011, 223) Ödev:** Yukarıdaki programı, karıştırılacak isim klavyeden girilecek şekilde yeniden düzenleyiniz.

**İpucu:** Bütün stringler aslında birer dizidir.

**Çözüm:**

```
using System;
namespace or223odev
{
    class Program
    {
        public static void Main(string[] args)
        {
            string isim;
            int sayi,a,uzunluk;

            Console.Write("Karıştırılacak ismi giriniz:");
            isim=Console.ReadLine();
            Console.WriteLine("Karıştırılacak isim "+(uzunluk=isim.Length)+" karakter");

            bool []D=new bool[uzunluk];
            Random rastgele=new Random();

            for(a=0;a<uzunluk;a++)
            {
                do{
                    sayi=rastgele.Next(uzunluk); //sıfır ile uzunluk arasında sayı üretiliyor.
                }while(D[sayi]==true);
                D[sayi]=true;
                Console.Write(isim[sayi]);
            }
            Console.Write("\n\nPress any key to continue . . ");
            Console.ReadKey(true);    } } }
```

**Array Sınıfı Özellik ve Metotları**

Her dizi MS .NET Framework içindeki **array** sınıfının (**class**) bir nesnesidir. Bu sayede diziler üzerinde pratik bazı işlemler yapılabilir.

Metot ve özellik	Açıklama
Dizi_Adı.Length	Dizinin eleman sayısı
Dizi_Adı.Rank	Dizinin boyutu
Array.Sort(Dizi_Adı)	Diziyi sırala (küçükten büyüğe)
Array.Reverse(Dizi_Adı)	Dizinin elemanlarını ters çevir/sırala
Array.Clear(Dizi_Adı,a,b)	Dizinin elemanlarını a indisinden başlayarak b adet sil.
Dizi_Adı.SetValue(x,y)	y. elemana x değerini ata
Array.BinarySearch(Dizi_Adı,anahtar)	Sıralı bir dizide anahtar değeri ara. Eğer anahtar değer bulunursa indis numarasını, bulunamazsa negatif değer döndürür (verir).
Array.IndexOf(Dizi_adı, anahtar);	Belirtilen dizide anahtar değeri arar. Eğer bulunursa indis numarasını, bulunamazsa -1 değerini döndürür.
String_Adı.Trim()	Stringin sol ve sağındaki boşlukları siler. Aradaki boşluklara dokunmaz.

**Örnek:** Array sınıfı özellik ve metotları.

```
using System;
namespace arrayclass{
    class Program {
        public static void Main(string[] args) {
            string[] isimler = { "Ali", "Veli", "Murat", "Mert", "Eren", "Onur", "Efe"};
            Console.WriteLine("isimler dizisi " + isimler.Length + " adet elemana sahiptir");
            Console.WriteLine("isimler dizisi " + isimler.Rank + " boyutludur");
            Array.Sort(isimler);
            Console.WriteLine("isimler dizisi elemanları küçükten büyüğe (a-z) sıralandı:");
            foreach (string txt in isimler) Console.WriteLine(txt);
            Array.Reverse(isimler);
            Console.WriteLine("isimler dizisi elemanları tersten yerleştirildi:");
            foreach (string txt in isimler) Console.WriteLine(txt);
            Array.Clear(isimler, 2, 3);
            Console.WriteLine("isimler dizisinin 2., 3., ve 4. indisindeki elemanlar silindi!");
            foreach (string txt in isimler) Console.WriteLine(txt);
            isimler.SetValue("Fikret", 2); isimler.SetValue("Ömer", 3); isimler.SetValue("Melih", 4);
            Console.WriteLine("ikinci indis değeri olarak \"Fikret\" değeri atandı..!");
            foreach (string txt in isimler) Console.WriteLine(txt);
            Array.Sort(isimler);
            Console.WriteLine("Yeni değerler eklenince isimler dizisi yeniden sıralandı..!");
            foreach (string txt in isimler) Console.WriteLine(txt);
            Console.WriteLine("Kime bakmıştınız? ");
            string kim = Console.ReadLine().Trim();
            if (Array.BinarySearch(isimler, kim) < 0)
                Console.WriteLine("BinarySearch ile aradığınız değer bulunamadı...!");
            else
                Console.WriteLine("BinarySearch ile aradığınız değer " + Array.BinarySearch(isimler, kim) + " sıralı indistedir");
            if (Array.IndexOf(isimler, kim) == -1)
                Console.WriteLine("Array.IndexOf ile aradığınız değer bulunamadı...!");
            else
                Console.WriteLine("Array.IndexOf ile aradığınız değer " + Array.IndexOf(isimler, kim) + " nolu indistedir..!");
            Console.ReadLine(); } } }
```



**Örnek (Yaşar, 2011, 224):** Bir otomobil fabrikasının ürettiği Molo, Metta ve Massat modellerinin sedan, station ve hatchback tipleri bulunmaktadır. Ayrıca her model ve tipin kırmızı, beyaz ve mavi renkleri de bulunmaktadır. Bu yapı için uygun bir dizi tanımlayarak her bir model, tip ve renge ait 0 – 10 arasında rastgele adet miktarı üreterek en yüksek adede sahip arabanın modelini, tipini ve rengini ekrana yazdırınız.

**İpucu:** Örneğe göre model, tip ve renk olmak üzere her bir model için toplam 9 ayrı çeşit otomobil vardır. 3 boyutlu ve her bir boyutu 3 elemanlı Araba isminde bir dizi tanımlayarak araba çeşitleri bellekte tutulabilir. Dizi **model, tip ve renk** olmak üzere üç boyutlu olacaktır. Her bir boyutun da üç farklı değeri olacaktır. Dizi üç boyutlu olduğu için diziye eleman atarken ve elemanlarını işlerken iç içe üç adet for döngüsü kullanılmalıdır. Araba[model][tip][renk]

using System;

namespace or224

{class Program

{public static void Main(string[] args)

{

int m,t,r,enbuyuk=-1500;

String s="";

int[,,,]Araba=new int[3,3,3];//tüm boyutlar int o yüzden sitringleri tutacak bir dizi daha gerek

String []isim={"Molo", "Metta", "Massat", "Sedan", "Station", "Hatchback", "Kırmızı", "Beyaz", "Mavi"};

Random rastgele=new Random();

for(m=0;m<=2;m++)

{

for(t=0;t<=2;t++)

{

for(r=0;r<=2;r++)

{

1	2	3	4	5	6	7	8	9
A[0,0,0] 7	A[0,0,1] 6	A[0,0,2] 3	A[0,1,0] 4	A[0,1,1] 5	A[0,1,2] 9	A[0,2,0] 7	A[0,2,1] 2	A[0,2,2] 8
A[1,0,0]	A[1,0,1]	A[1,0,2]	A[1,1,0]	A[1,1,1]	A[1,1,2]	A[1,2,0]	A[1,2,1]	A[1,2,2]
A[2,0,0]	A[2,0,1]	A[2,0,2]	A[2,1,0]	A[2,1,1]	A[2,1,2]	A[2,2,0]	A[2,2,1]	A[2,2,2]

Araba[m,t,r]=rastgele.Next(11); } }

for(m=0;m<=2;m++) //en çok hangi üründen üretildiğini bulur!

{

for(t=0;t<=2;t++)

{

for(r=0;r<=2;r++)

{

if(Araba[m,t,r]>enbuyuk)

{

enbuyuk=Araba[m,t,r];//s=Molo Station Mavi

s=isim[m+0]+" "+isim[t+3]+" "+isim[r+6]+" "+enbuyuk+" Adet";

//s=model[m]+" "+tip[t]+" "+renk[r]+" "+enbuyuk+" adet üretilmiştir";

} }

Console.WriteLine("En yüksek adete sahip araba: ");

Console.WriteLine(s);

Console.WriteLine(" Press any key to continue . . . ");

Console.ReadKey(true); }

**Ödev:** Yukarıdaki programı, en yüksek adete sahip birden fazla ürün olduğunda bunların tamamını da ekrana yazacak şekilde yeniden düzenleyiniz. **İpucu:** önce en çok üretim adedi değer olarak üçlü for döngü grubunun en içteki döngüsünde if ile bulunur. Daha sonra yeni bir üçlü for döngüsü grubu içinde en yüksek üretim adedi tüm üretim adetleriyle karşılaştırılır. Eşit olan yerde if kapsamından çıkmadan isimler dizisinden m, t, r değişkenlerinin işaret ettiği isimler s stringinde birleştirilerek veya doğrudan Console.WriteLine() metodu içinde birleştirilerek ekrana yazılır.

**Örnek (YAŞAR, 2011, 226):** Elemanları dışarıdan girilen  $n \times n$  iki matrisin eşit olup olmadığını bulan program.

**İpucu:** iki  $n \times n$  matrisin eşit olabilmesi için sırası ile tüm elemanlarının değerlerinin eşit olması gerekmektedir.

```
using System;
namespace or226
{
    class Program
    {
        public static void Main(string[] args)
        {
            int n, satir, sutun;
            string s="A ve B matrisleri Eşittir";
            Console.Write("n*n için n değerini giriniz:");
            n=Int32.Parse(Console.ReadLine());
            int [,]a=new int[n,n];
            int [,]b=new int[n,n];
            for(satir=0;satir<n;satir++) //A matrisi giriliyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write("A Matrisinin "+satir+" "+sutun+" Sayı=");
                    a[satir,sutun]=Int32.Parse(Console.ReadLine());
                }
            }

            for(satir=0;satir<n;satir++) //B matrisi giriliyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write("B Matrisinin "+satir+" "+sutun+" Sayı=");
                    b[satir,sutun]=Int32.Parse(Console.ReadLine());
                }
            }

            for(satir=0;satir<n;satir++) //Matrisler eşit mi diye bakılıyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    if(a[satir,sutun]!=b[satir,sutun]) {s="A ve B Matrisleri Eşit Değildir!";break;}
                }
            }

            Console.WriteLine(s);
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek (YAŞAR, 2011, 228):**  $n \times n$  boyutunda birim matris elde ederek ekrana yazdıran program.

**İpucu :**  $n \times n$  birim matrisin özelliği, sol üstten sağ alta uzanan köşegen üzerindeki tüm elemanların değerlerinin 1 olmasıdır. Örneğin  $3 \times 3$  bir matrisin birim matrisi aşağıdaki gibidir:

1	0	0
0	1	0
0	0	1

Şekilden de görüldüğü gibi satır ve sütun numarası eşit olan elemanların değeri 1, diğerlerinin değerleri ise sıfır olacaktır. Program algoritması buna göre kurulmalıdır.

```
using System;
namespace or228
{
    class Program
    {
        public static void Main(string[] args)
        {
            int n, satir, sutun;
            Console.Write("n*n için n değerini giriniz:");
            n=Int32.Parse(Console.ReadLine());
            int [,]a=new int[n,n];
            for(satir=0;satir<n;satir++)
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    if(satir==sutun) a[satir,sutun]=1;
                    else a[satir,sutun]=0;
                    Console.Write(a[satir,sutun]+"\\t");
                }
                Console.Write("\\n");
            }
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek (YAŞAR, 2011, 229):**  $n \times n$  tipinde iki matrisin toplamını bulan program.

**İpucu:** Matrislerin toplanabilmeleri için aynı satır ve sütun sayılarına sahip olmaları gereklidir. Örnekte satır ve sütun sayıları eşit olan matristen söz edildiği için sorun yoktur. Önce 3 adet dizi tanımlanmalıdır. Tahmin edilebileceği gibi birinci dizide a matrisi, ikinci dizide b matrisi tutulurken, üçüncü dizide ise toplam değerler tutulacaktır. İki boyutlu dizilerde ilk indis değerinin satırı, ikinci indis değerinin ise sütunu ifade ettiği unutulmamalıdır.

```
using System;
namespace or229
{
    class Program
    {
        public static void Main(string[] args)
        {
            int n, satir, sutun;
            Console.Write("n*n için n değerini giriniz:");
            n=Int32.Parse(Console.ReadLine());
            int [,] a=new int[n,n];
            int [,] b=new int[n,n];
            int [,] toplam=new int[n,n];
            for(satir=0;satir<n;satir++) // A matrisinin değerleri sıra ile giriliyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write("A Matrisinin "+satir+" "+sutun+" Sayı=");
                    a[satir,sutun]=Int32.Parse(Console.ReadLine());
                }
            }
            Console.WriteLine("A Matrisi:");
            for(satir=0;satir<n;satir++) //A matrisi ekrana matris gibi satır sütun düzeninde yazdırılıyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write(a[satir,sutun]+"\\t");
                }
                Console.WriteLine("\\n");
            }

            for(satir=0;satir<n;satir++) // B matrisinin değerleri sıra ile giriliyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write("B Matrisinin "+satir+" "+sutun+" Sayı=");
                    b[satir,sutun]=Int32.Parse(Console.ReadLine());
                }
            }
            Console.WriteLine("B Matrisi:");
            for(satir=0;satir<n;satir++) //B matrisi ekrana matris gibi satır sütun düzeninde yazdırılıyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write(b[satir,sutun]+"\\t");
                }
                Console.WriteLine("\\n");
            }
        }
    }
}
```

```

Console.WriteLine("Toplam Matrisi :");
for(satir=0;satir<n;satir++)
{
    for(sutun=0;sutun<n;sutun++)
    {
        Console.Write((toplam[satir,sutun]=a[satir,sutun]+b[satir,sutun])+"\t");
    }
    Console.Write("\n");
}
Console.Write("Press any key to continue . . . ");
Console.ReadKey(true);
}
}
}

```

**Örnek (YAŞAR, 2011, 232):** Elemanları dışarıdan girilen  $n \times n$  boyutunda iki matrisin çarpımını ekrana yazan program.

**İpucu:** Matrisler çarpılırken ilk matrisin satırı ile ikinci matrisin sütunları çarpılmaktadır. Elemanların çarpımları toplanarak çarpım matrisinin bir elemanı elde edilmektedir. Önce matrislerin elemanlarını dışarıdan istenerek A ve B matrisleri elde edilir. Daha sonra da iç içe üç döngü ile matrislerin çarpımı olan çarpım matrisi elde edilir.

$$A = \begin{pmatrix} a(0,0) & a(0,1) \\ a(1,0) & a(1,1) \end{pmatrix} \qquad B = \begin{pmatrix} b(0,0) & b(0,1) \\ b(1,0) & b(1,1) \end{pmatrix}$$

$$\text{ÇARPIM} = \begin{pmatrix} a(0,0)*b(0,0)+a(0,1)*b(1,0) & a(0,0)*b(0,1)+a(0,1)*b(1,1) \\ a(1,0)*b(0,0)+a(1,1)*b(1,0) & a(1,0)*b(0,1)+a(1,1)*b(1,1) \end{pmatrix}$$

```

using System;
namespace or232
{
    class Program
    {
        public static void Main(string[] args)
        {
            int n, satir, sutun, t;
            Console.Write("n*n için n değerini giriniz=");
            n=Convert.ToInt32(Console.ReadLine());
            int [,]a=new int[n,n];
            int [,]b=new int[n,n];
            int [,]carpim=new int[n,n];
            for(satir=0;satir<n;satir++) // A matrisinin değerleri sıra ile giriliyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write("A Matrisinin "+satir+" "+sutun+" Sayı=");
                    a[satir,sutun]=Int32.Parse(Console.ReadLine());
                }
            }
            Console.WriteLine("A Matrisi:");
            for(satir=0;satir<n;satir++) //A dizisi ekrana matris gibi satır sütun düzeninde yazdırılıyor
            {
                for(sutun=0;sutun<n;sutun++)

```

```

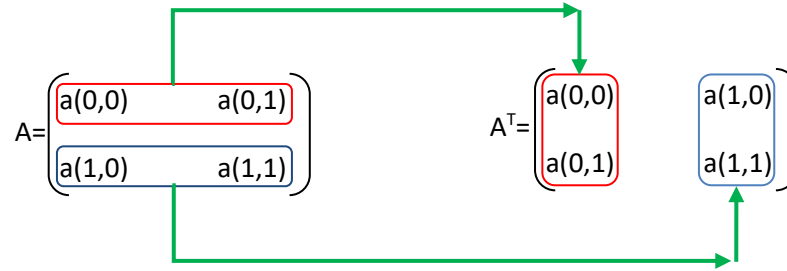
    {
        Console.Write(a[satir,sutun]+"\\t");
    }
    Console.Write("\\n");
}

for(satir=0;satir<n;satir++) // B matrisinin deęerleri sıra ile giriliyor
{
    for(sutun=0;sutun<n;sutun++)
    {
        Console.Write("B Matrisinin "+satir+" "+sutun+" . Sayı=");
        b[satir,sutun]=Int32.Parse(Console.ReadLine());
    }
}
Console.WriteLine("B Matrisi:");
for(satir=0;satir<n;satir++) //B dizisi ekrana matris gibi satır sütun düzeninde yazdırılıyor
{
    for(sutun=0;sutun<n;sutun++)
    {
        Console.Write(b[satir,sutun]+"\\t");
    }
    Console.Write("\\n");
}
Console.WriteLine("Çarpım Matrisi :");
for(satir=0;satir<n;satir++)
{
    for(sutun=0;sutun<n;sutun++)
    {
        carpim[satir,sutun]=0;
        for(t=0;t<n;t++)
        {
            carpim[satir,sutun]=carpim[satir,sutun]+(a[satir,t]*b[t,sutun]);
        }
    }
}
for(satir=0;satir<n;satir++)//çarpım dizisi ekrana matris gibi satır sütun düzeninde yazdırılıyor.
{
    for(sutun=0;sutun<n;sutun++)
    {
        Console.Write(carpim[satir,sutun]+"\\t");
    }
    Console.WriteLine();
}
Console.Write("\\nDevam etmek için bir tuşa basınız . . . ");
Console.ReadKey(true);
}
}
}

```

**Örnek (YAŞAR, 2011, 235):** A matrisinin transpozisini alınız.

**İpucu:** Transpoze işlemini T harfi ifade eder. Bir matrisin transpozesi satırları ile sütunları yer değiştirilerek elde edilir. Bu yapı aşağıda gösterilmiştir. A matrisi kaynak matris, B matrisi de A'nın transpozesi olmak üzere kodlamalar oluşturulmalıdır.  $B[SUTUN,SATIR]=A[SATIR,SUTUN]$



```
using System;
namespace or235
{
    class Program
    {
        public static void Main(string[] args)
        {
            int n, satir, sutun;
            Console.WriteLine("n*n için n değerini giriniz=");
            n=Convert.ToInt32(Console.ReadLine());
            int [,]a=new int[n,n];
            int [,]b=new int[n,n];

            for(satir=0;satir<n;satir++) // A matrisinin değerleri sıra ile giriliyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.WriteLine("A Matrisinin "+satir+" "+sutun+" Sayı=");
                    a[satir,sutun]=Int32.Parse(Console.ReadLine());
                }
            }
            Console.WriteLine("A Matrisi:");
            for(satir=0;satir<n;satir++) //A dizisi ekrana matris gibi satır sütun düzeninde yazdırılıyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write(a[satir,sutun]+"\\t");
                }
                Console.WriteLine("\\n");
            }

            for(satir=0;satir<n;satir++) //Yer değiştirme işlemi yapılıyor.
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    b[sutun,satir]=a[satir,sutun];
                }
            }
            Console.WriteLine("\\n\\nA Matrisinin Transpozesi:");
            for(satir=0;satir<n;satir++)//yer değiştirilmiş matris ekrana yazılıyor
            {
```

```

    for(sutun=0;sutun<n;sutun++)
    {
        Console.Write(b[satir,sutun]+"\\t");
    }
    Console.WriteLine();
}
Console.Write("\\nDevam etmek için bir tuşa basınız . . . ");
Console.ReadKey(true);
}
}
}

```

**Örnek (YAŞAR, 2011, 238):**  $n \times n$  boyutundaki A matrisinin simetrik olup olmadığını bulan program:

**İpucu:** bir matrisin simetrik olabilmesi için o matrisin bir kare matris olması ve aynı zamanda da transpozese eşit olması gerekir. Ayrıca matrisin transpozese bulmadan önce simetrik olup olmadığına bakmak için asıl köşegenin altındaki ve üstündeki karşılıklı elemanların eşit olması gerekmektedir.

4	2	8
2	3	6
8	6	7

```

using System;
namespace or238
{
    class Program
    {
        public static void Main(string[] args)
        {
            int n, satir, sutun;
            string s="Simetrik";
            Console.Write("n*n için n değerini giriniz=");
            n=Convert.ToInt32(Console.ReadLine());
            int [,]a=new int[n,n];
            for(satir=0;satir<n;satir++) // A matrisinin değerleri sıra ile giriliyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write("A Matrisinin "+satir+" "+sutun+" Sayı=");
                    a[satir,sutun]=Int32.Parse(Console.ReadLine());
                }
            }
            Console.WriteLine("\\nA Matrisi:");
            for(satir=0;satir<n;satir++) //A dizisi ekrana matris gibi satır sütun düzeninde yazdırılıyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.Write(a[satir,sutun]+"\\t");
                }
                Console.WriteLine("\\n");
            }
        }
    }
}

```



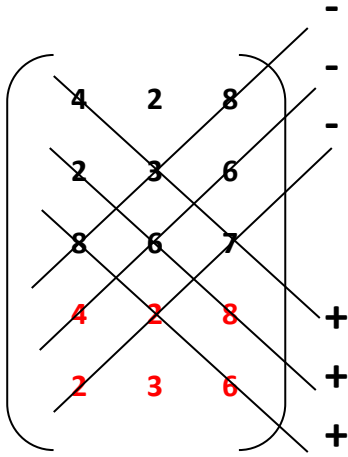
```
for(satir=0;satir<n;satir++) //
{
    for(sutun=0;sutun<n;sutun++)
    {

        if(satir==sutun) continue;
        if(a[satir,sutun]!=a[sutun,satir])
        {
            s="Simetrik değil";
            break;
        }
    }
}
Console.WriteLine("\n"+s);
Console.Write("\nDevam etmek için bir tuşa basınız . . ");
Console.ReadKey(true);
}
}
```

---

**Örnek (YAŞAR, 2011, 239):** 3\*3, X matrisinin determinantını Sarus kuralına göre bulunuz.

**İpucu:** Sarus kuralına göre determinant hesabı sadece 3\*3 matrislerde geçerlidir. Buna göre ilk iki satır matrisin altına tekrar eklenir. Daha sonra sol üst köşeden sağ alt köşeye doğru inen toplam üç köşegen üzerindeki elemanlar her bir köşegen için birbiri ile çarpılarak toplanır. Daha sonra sol alt köşeden sağ üst köşeye doğru toplam üç köşegen üzerindeki elemanlar her bir köşegen için birbiri ile çarpılarak toplanır. Elde edilen birinci toplamdan ikinci toplam çıkarılır ve elde edilen sonuç ilgili matrisin determinant değeridir.



$$A = a_{00} * a_{11} * a_{22}$$

$$B = a_{10} * a_{21} * a_{02}$$

$$C = a_{20} * a_{01} * a_{12}$$

$$D = a_{20} * a_{11} * a_{02}$$

$$E = a_{00} * a_{21} * a_{12}$$

$$F = a_{10} * a_{01} * a_{22}$$

$$\text{Det } X = (A+B+C) - (D+E+F)$$

```
using System;
namespace or239
{
    class Program
    {
        public static void Main(string[] args)
        {
            int n, satir, sutun, det;

            Console.WriteLine("n*n için n değerini giriniz=");
            n=Convert.ToInt32(Console.ReadLine());
            int [,]a=new int[n,n];
            for(satir=0;satir<n;satir++) // A matrisinin değerleri sıra ile giriliyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.WriteLine("A Matrisinin "+satir+" "+sutun+" Sayı=");
                    a[satir,sutun]=Int32.Parse(Console.ReadLine());
                }
            }
            Console.WriteLine("\nA Matrisi:");
            for(satir=0;satir<n;satir++) //A matrisi ekrana matris gibi satır sütun düzeninde yazdırılıyor
            {
                for(sutun=0;sutun<n;sutun++)
                {
                    Console.WriteLine(a[satir,sutun]+"\\t");
                }
                Console.WriteLine("\\n");
            }

            det=a[0,0]*a[1,1]*a[2,2]+a[1,0]*a[2,1]*a[0,2]+a[2,0]*a[0,1]*a[1,2]
            -a[2,0]*a[1,1]*a[0,2]-a[1,2]*a[2,1]*a[0,0]-a[2,2]*a[0,1]*a[1,0];
            Console.WriteLine("\\n"+det);
        }
    }
}
```

```
Console.Write("\nDevam etmek için bir tuşa basınız . . .");
```

```
Console.ReadKey(true);
```

```
}
```

```
}
```

```
}
```

**Örnek (Yaşar, 2011, 241):**  $n \times n$ ,  $X$  matrisinin determinantını bulunuz.

**İpucu:** Sarus kuralı sadece  $3 \times 3$  matrisler için uygulanırken daha küçük veya daha fazla boyutlu matrisler için kullanılamamaktadır. Bir matrisin determinantının alınabilmesi için kare matris olması gereklidir. Kare matrisler ise satır ve sütun sayıları eşit olan matrislerdir.  $n \times n$  matrisin determinantını hesaplamak için aşağıdaki kurallar uygulanır:

1. İstenen bir satır veya sütun tamamen seçilir.
2. Bu satır ve sütunun tüm elemanları artı işaretinden başlayarak bir artı bir eksi yapılır.
3. Seçilen satır veya sütunun her bir elemanını bulunduğu satır veya sütun dışındaki kalan elemanların determinantları hesaplanır ve sonuç seçilen elemanla işaretli olarak çarpılır. Aşağıda seçilen satırın ilk elemanı için (1) bu yapı gösterilmiştir.
4. Tüm çarpım sonuçları toplanarak determinant bulunur.

$$X = \begin{bmatrix} 1 & 3 & 2 \\ 4 & 1 & 3 \\ 2 & 5 & 2 \end{bmatrix} \xrightarrow{\text{İlk satır seçildi ve işaretler değiştirildi}} X = \begin{bmatrix} +1 & -3 & +2 \\ 4 & 1 & 3 \\ 2 & 5 & 2 \end{bmatrix}$$

$$\det X = +1 \begin{vmatrix} 1 & 3 \\ 5 & 2 \end{vmatrix} - 3 \begin{vmatrix} 4 & 3 \\ 2 & 2 \end{vmatrix} + 2 \begin{vmatrix} 4 & 1 \\ 2 & 5 \end{vmatrix}$$

Yukarıdaki metoda göre örneğin  $4 \times 4$ 'lük bir matris için istenen satır ve sütun seçildiği halde  $3 \times 3$ 'lük bir alt matris kalmaktadır.  $3 \times 3$ 'lük matrisi de bu şekilde çözerek  $2 \times 2$ 'lik forma dönüşene kadar devam edilmelidir.

```

using System;
namespace or241
{
    class Program
    {
        public int determinant(int[,] mat)
        {
            int sonuc=0;
            if(mat.Length==1)
            {sonuc=mat[0,0];
            return sonuc;
            }
            if(mat.Length==2)
            {sonuc=mat[0,0]*mat[1,1]-mat[0,1]*mat[1,0];
            return sonuc;
            }
            int n=(int)Math.Sqrt(mat.Length);
            for(int i=0; i<n;i++)
            {
                int[,] temp=new int[n-1,n-1];
                for(int j=1;j<n;j++)
                {
                    for(int k=0;k<n;k++)
                    {
                        if(k<i)
                        {
                            temp[j-1,k]=mat[j,k];
                        }
                        else if(k>i)
                        {
                            temp[j-1,k-1]=mat[j,k];
                        }
                    }
                }
                sonuc+=mat[0,i]*(int)Math.Pow(-1,i)*determinant(temp);
            }
            return sonuc;
        }
    }

    public static void Main(string[] args)
    {
        int n,sa,su;
        Console.Write("n değerini giriniz: ");
        n=Int32.Parse(Console.ReadLine());
        int [,] a=new int[n,n];
        for(sa=0;sa<=n-1;sa++)
        {
            for(su=0;su<=n-1;su++)
            {
                Console.Write("Matrisin "+sa+" "+su+" Elemanı: ");
                a[sa,su]=Int32.Parse(Console.ReadLine());
            }
        }
        Program hesap=new Program();
        int c=hesap.determinant(a);
    }
}

```

```
Console.WriteLine("Girilen Matrisin Determinantı: "+c);
```

```
Console.Write("Press any key to continue . . . ");
```

```
Console.ReadKey(true);
```

```
}
```

```
}
```

```
}
```

## String (Sözcük) İşlemleri

String S="İstanbul", YS="Sivas";

İşlev	Metot	Kullanım Biçimi	Çıktı
Uzunluğu (karakter Sayısı)	Length	S.Length	8
Büyük harfe dönüştürme	ToUpper()	S.ToUpper()	İSTANBUL
Küçük harfe dönüştürme	ToLower()	S.ToLower()	istanbul
Parça Kopyalama – 1 (a. indisten itibaren sonuna kadar)	Substring(a)	S.Substring(2)	tanbul
Parça kopyalama – 2 (a. indisten itibaren b tane)	Substring(a,b)	S.Substring(2,4)	tanb
Parça silme. <u>Remove(a)</u> , a. indisten başlayarak sonuna kadar siler. <u>Remove(a,b)</u> a. indisten başlayarak b tane siler.	Remove(a,b)	S.Remove(2,4)	İstan <b>bul</b>
String değiştirme. Replace(d,e) d stringini e ile değiştirir.	Replace(d,e)	YS.Replace("S","k")	kivas

### Örnek (YAŞAR, 2011, 248):

```
using System;
```

```
namespace or248
```

```
{
```

```
class Program
```

```
{
```

```
public static void Main(string[] args)
```

```
{int uzunluk;
```

```
string S="İstanbul";
```

```
Console.WriteLine(S.Length);//8
```

```
Console.WriteLine(S.ToUpper());//İSTANBUL
```

```
Console.WriteLine(S.Substring(3));//anbul
```

```
Console.WriteLine(S.Substring(3,4));//anbu
```

```
Console.WriteLine(S.Replace("s","k"));//İktanbul
```

```
Console.WriteLine(S.Remove(2,3)); //İsbul
```

```
Console.Write("Press any key to continue . . . ");
```

```
Console.ReadKey(true);
```

```
}
```

```
}
```

```
}
```

**Örnek (YAŞAR, 2011, 249):** Dışarıdan girilen bir cümleyi tersten yazdırınız.

```
using System;
namespace or249
{
    class Program
    {
        public static void Main(string[] args)
        {
            string s;
            Console.Write("Bir cümle giriniz:");//ali
            s=Console.ReadLine();//s=ali
            int uzunluk;
            uzunluk=s.Length;//uzunluk=3
            for(int a=uzunluk-1;a>=0;a--)
                Console.Write(s[a]);
            Console.WriteLine("Press any key to continue . . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Örnek (YAŞAR, 2011, 250):** Dışarıdan girilen bir kelimenin uzunluğunu Length metodunu kullanmadan bulan program.

```
using System;
namespace or250
{
    class Program
    {
        public static void Main(string[] args)
        {
            string s; int a;
            Console.Write("Bir cümle giriniz:");//ali
            s=Console.ReadLine();//s=ali
            s=s+"\0";//s stringinin sonuna NULL karakter ekleniyor. S=aliNULL
            for(a=0;;a++)
            {
                if(s[a]=='\0') break;
            }
            Console.WriteLine("Girilen cümlenin uzunluğu: "+a+" karakter");

            Console.WriteLine("Press any key to continue . . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Örnek (YAŞAR, 2011, 251):** Dışarıdan girilen bir cümleyi ekrana her kelime bir satırda olacak şekilde yazan program.

```
using System;
namespace or251
{
    class Program
    {
        public static void Main(string[] args)
        {
            string s; int a;
            Console.Write("Bir cümle giriniz:");//ali gel
            s=Console.ReadLine();
            int uzunluk=s.Length;// uzunluk=7;
            for(a=0;a<uzunluk;a++)
            {
                if(s[a]==' ') Console.WriteLine();
                else Console.Write(s[a]);//ali ENTER gel
            }
            Console.WriteLine("\n\nPress any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek (YAŞAR, 2011, 253):** bir cümle içerisinde istenen bir kelimeyi arayıp kaçınıcı karakterden sonra başladığını bulan program.

**İpucu :** Girilen cümle içerisinde aranan kadar kelimelerin baştan sona kadar aranan kelimeye eşit olup olmadığına bakmak gerekmektedir. Örneğin “Ali eve erken gel” cümlesinde “eve” kelimesini aramak için eve kelimesi 3 harfli olduğu için cümlemizi baştan başlayarak 3’erli gruplar halinde kopyalayıp aranan kelimeye eşit olup olmadığına bakmamız gerekmektedir. Eşit olduğu yerde, sonuç değişkenine bulunduğu yerin numarasını ekleriz.

```
using System;
namespace or253
{
    class Program
    {
        public static void Main(string[] args)
        {
            string ys, s, aranan, sonuc = "Bulunamadı";
            int a;
            Console.Write("Bir cümle giriniz:");//ali eve erken gel → 17 karakter
            s = Console.ReadLine();
            Console.Write("Aranan Kelime:");//eve → 3 karakter
            aranan = Console.ReadLine();
            for (a = 0; a <= s.Length - aranan.Length; a++)
            {
                if (s.Substring(a, aranan.Length) == aranan)
                { sonuc = ("Bulunduğu yer: " + (a+1)); Console.WriteLine(sonuc); }
                //else Console.WriteLine("Bulunamadı");
            }
            if (sonuc == "Bulunamadı") Console.WriteLine(sonuc);
            Console.WriteLine("\n\nPress any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek (YAŞAR, 2011, 254):** Girilen bir cümle içindeki tüm boşlukları atarak kelimeleri bitişik yazdıran program.

**İpucu:** Girilen cümle içinde tek tek karakterlere bakıp eğer boşluk değilse bir başka (ys) string değişkenle toplayıp, boşluk ise işlem yapmadan ihmal ederek, diğer karakterlere cümlemin sonuna kadar bakmak gerekmektedir.

```
using System;
namespace or254
{
    class Program
    {
        public static void Main(string[] args)
        {
            string ys="";
            int a;
            Console.WriteLine("Bir cümle giriniz:");//ali gel
            s=Console.ReadLine();
            for(a=0;a<s.Length;a++)//s.Length=7
            {
                if(s[a]!=' ')
                    ys=ys+s[a];//a, al, ali, alig, alige, aligel
            }
            Console.WriteLine(ys);
            Console.WriteLine("\n\nPress any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Örnek (YAŞAR, 2011, 255):** Dışarıdan girilen bir cümlemin bir kelimesini büyük, bir kelimesini küçük olarak yazdıran program.

**İpucu:** Cümle içerisinde kelimeleri ayıran karakter boşluk olduğu için, her bir boşlukta mod değiştirerek karakterleri tek tek moda göre büyük ya da küçük yazmak gerekir.

```
using System;
namespace or255
{
    class Program
    {
        public static void Main(string[] args)
        {
            string ys="",s,k="";
            bool mod=false;
            int a;
            Console.WriteLine("Bir cümle giriniz:");//ali EVE erken GEL
            s=Console.ReadLine();
            for(a=0;a<s.Length;a++)//s.Length=17
            {
                k="";
                if(s[a]==' ')
                {
                    mod=!mod;//TRUE
                    ys=ys+" ";
                    continue;
                }
                k=k+s[a];//char tipini stringe dönüştürmek için kullanıldı k=a, l, i, E, V, E
                if (mod)
                    ys=ys+k.ToLower();//"ALİ "+e, ALİ ev, ALİ eve,
                else
                    ys=ys+k.ToUpper();//A, AL, ALİ, ALİ ,
            }
        }
    }
}
```



```

Console.WriteLine(ys);
Console.WriteLine("\n\nPress any key to continue . . . ");
Console.ReadKey(true);
    }
}
}

```

**Örnek (YAŞAR, 2011, 257):** Dışarıdan girilen tek sayıda harfe sahip bir kelimeyi aşağıdaki biçimde satır satır yazdıran prgram.

		M		
	H	M	E	
A	H	M	E	T

**İpucu:** Şekle dikkatle bakılırsa önce boşluk sonra karakterler konularak yapılması gerekmektedir. Burada satır sayısı ve ortadaki harf kelime uzunluğuna göre bulunmaktadır. En önemli nokta on1 değişkeni satırdaki harflerin başlangıcını, on1+a değeri de sonunu gösterir. Her satırda boşluklar bir azalmaktadır (bb değişkeni). Harf sayısı da artmaktadır. (on1 ile on1+a arası)

```

using System;
namespace or257
{
    class Program
    {
        public static void Main(string[] args)
        {
            string s;
            int bb, m,n,ss,uzunluk,a=0,on1;
            Console.Write("Bir cümle giriniz:");//AHMET
            s=Console.ReadLine();
            uzunluk=s.Length;//uzunluk=5
            bb=(uzunluk-1)/2;//bb=2
            ss=bb+1;//ss=3
            on1=ss-1;//on1=2
            for(m=1;m<=ss;m++)
            {
                for(n=1;n<=bb;n++)
                {
                    Console.Write(" ");
                }
                for(n=on1;n<=on1+a;n++)
                {
                    Console.Write(s[n]);
                }
                bb=bb-1; on1=on1-1;a=a+2;
                Console.WriteLine();
            }
            Console.WriteLine("\n\nPress any key to continue . . . ");
            Console.ReadKey(true);
        } }
}

```

**Örnek (YAŞAR, 2011, 259):** Dışarıdan girilen kelimenin küçük harflerini büyük yazan program. (C# için ToUpper kullanmadan)

**İpucu:** Küçük harfler ASCII kodu olarak 97'den, büyük harfler ise 65'ten başlar. Küçük harfin ASCII kodu büyük harfinkinden 32 büyüktür. Buna göre bir karakterin önce ASCII kodunu bulup, daha sonra ASCII kodu eğer 97'den büyük ise o karakterin küçük harf olduğuna karar verip, daha sonra ASCII kodundan 32 çıkarmak gerekir ki büyük harfe dönüşsün. En son olarak ta ASCII kodu yeniden karaktere dönüştürmek gerekecektir.

```
using System;
namespace or259
{
    class Program
    {
        public static void Main(string[] args)
        {
            string s, ys="";
            int a;
            byte c;
            Console.Write("Bir cümle giriniz:");
            s=Console.ReadLine();
            for(a=0;a<s.Length;a++)
            {
                c=(byte)s[a]; //s stringinin sıradaki karakterinin ASCII kodu elde ediliyor!
                if(c>=97 && c<=122) //küçük harf ise
                    ys=ys+(char)(c-32); //ASCII kodu tekrar karaktere dönüştürülüyor!
                else ys=ys+s[a];
            }
            Console.WriteLine(ys);
            Console.WriteLine("\n\nPress any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek (YAŞAR, 2011, 260):** dışarıdan girilen tek sayılı satır kadar, aşağıdaki şekli satır satır oluşturan program.

Satır Sayısı: 11	Satır sayısı: 3
*	*
*	*
*	*
*	
*	
*	
*	
*	
*	

**İpucu:** Her satırdaki boşluk sayıları sıfırdan başlayıp orta noktaya kadar artmakta, sonrasında ise azalmaktadır. Her satırda önce boşluklar sonra ise yıldız karakteri yazılmaktadır. Orta nokta (satır sayısı+1)/2 formülü ile hesaplanmaktadır.

```

using System;
namespace or260
{
    class Program
    {
        public static void Main(string[] args)
        {
            int bb=0, m,n,ss;
            Console.Write("Satır Sayısı:");//3
            ss=Int32.Parse(Console.ReadLine());
            for(m=1;m<=ss;m++)
            {
                for(n=1;n<=bb;n++)
                Console.Write(" ");
                Console.WriteLine("*");
                if(m>=((ss+1)/2)) //orta noktaya ulaşp ulaşmadığı kontrol ediliyor.
                    bb=bb-1;//orta noktaya ulaştıysa boşuk sayısı azaltılıyor
                else
                    bb=bb+1;//orta noktaya ulaşmadıysa boşluk sayısı artırılıyor
            }
            Console.WriteLine("\n\nPress any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}

```

## Koleksiyonlar

Klasik dizileri dizi boyutunu bildirdikten sonra daraltmak veya genişletmek mümkün değildir. Ayrıca dizi boyutu bildirildikten sonra içine değer atanmasa bile bellekte ayrılan yeri yine de kaplarlar.

ArrayList sınıfı ise böyle değildir ve System.Collections sınıf kütüphanesini kullanmaktadır.

- ArrayList her türlü nesneyi kabul eder.
- ArrayList içindeki veriler object tipinde olduğundan, nesne kullanılmak istendiğinde tip dönüşümü gerekebilir.
- Collection.Generic sınıfını kullanan GenericList<> sınıfının ilkel halidir.

using System; **//(Uzunköprü)**

using System.Collections;

namespace \_7\_11\_ArrayList

```
{
    class Program
    {
        static void Main(string[] args)
        {
            ArrayList listedizi = new ArrayList();//listedizi adında ArrayList sınıfından bir nesne tanımlanıyor
            listedizi.Add("Celal Bayar Üniversitesi");//listenin SONUNA object tipinde değer ekleniyor
            listedizi.AddRange("Harflerine Ayır".ToCharArray());//string harf harf ayrılıyor
            listedizi.Insert(16, "MCBÜ MYO");//16. indise MCBÜ MYO değeri ekleniyor. 15. indis yoksa hata oluşur
            listedizi.InsertRange(16, "Merhaba".ToCharArray());//16. indise Merhaba'yı harf harf ekler
            Console.WriteLine("Kapasite: " + listedizi.Capacity);/* Dizinin kapasitesini verir. kapasite 2^n artar. Eleman sayısı<=Kapasite ise Kapasite 2'nin üslerinden biri olur. Örneğin 5 elemanlı dizi için kapasite 5<=X ise X=2^3=8 olur.*/
            listedizi.TrimToSize();//listenin üzerindeki kapasite ile eleman sayısı eşitlenir.
            Console.WriteLine("Yeni kapasite: " + listedizi.Capacity);
            Console.WriteLine("Kaç Eleman var: " + listedizi.Count);//listedeki eleman sayısını verir.
            listedizi.Remove("MCBÜ MYO");//object tipindeki liste elemanı diziden kaldırılır
            listedizi.RemoveAt(11);//11 indisindeki eleman diziden kaldırılır
            listedizi.RemoveRange(0, 1);//0 indisinden itibaren 1 eleman diziden kaldırılır
            Console.WriteLine(listedizi.Contains("Ali") == false ? "Aranan değer yok" : "Değer var");//Dizide "Ali" aranır. Bulursa true, bulamazsa false döner.
            Console.WriteLine(listedizi.IndexOf("Veli") == -1 ? "Aranan değer yok" : "Değer var");//Dizide "Veli" değerini arar. bulursa indis numarasını, bulamazsa -1 döndürür.
            Console.WriteLine(listedizi.LastIndexOf(35) == -1 ? "Aranan değer yok" : "Değer var");//Dizide 35 değeri birden fazla varsa son 35'in indis numarasını verir. Bulamazsa -1 verir.

            for (int i = 0; i < listedizi.Count; i++)//liste dizi elemanları sırayla ekrana yazdırılıyor
            {
                Console.WriteLine(i + ". eleman=" + listedizi[i]);
            }

            Console.WriteLine("\n\nSıralanmış dizi elemanları:");
            listedizi.Sort();//dizi elemanları küçükten büyüğe sıralanıyor.
            for (int i = 0; i < listedizi.Count; i++)//liste dizi elemanları sırayla ekrana yazdırılıyor
            {
                Console.WriteLine(i + ". eleman=" + listedizi[i]);
            }
            listedizi.Reverse();//dizi elemanları ters çevriliyor

            Console.WriteLine("\n\nTers çevrilmiş dizi elemanları:");
            for (int i = 0; i < listedizi.Count; i++)//liste dizi elemanları sırayla ekrana yazdırılıyor
            {
                Console.WriteLine(i + ". eleman=" + listedizi[i]);
            }
        }
    }
}
```

```

listedizi.ToArray();//listeyi Object[] Array tipine çevirir.
Console.WriteLine(listedizi.ToString());//listeyi string tipine çevirir. Tüm liste string tipine çevirilemezse sadece
System.Collections.ArrayList isim uzayı döner.
listedizi.Clear();//listedeki tüm elemanları siler
Console.ReadLine();
}
}
}

```

**Örnek (Uzunköprü):** Kullanıcının girdiği bir sayıya kadar olan sayıların, tek ve çift olanlarını ayrı ayrı dizilerde saklayıp, ekrana yazan C# konsol uygulaması.

```

using System;
using System.Collections;
namespace _7_12_ArrayListTekCift
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Tek ve Çift Rakamlar, hangi sayıya kadar listelenecek:");
            int kacaKadar = Convert.ToInt32(Console.ReadLine());
            ArrayList tekSayilar = new ArrayList();
            ArrayList ciftSayilar = new ArrayList();
            for (int i = 1; i < kacaKadar; i++)
            {
                if (i % 2 == 0) ciftSayilar.Add(i);
                if (i % 2 != 0) tekSayilar.Add(i);
            }
            Console.WriteLine("Tek Sayılar:");
            for (int i = 0; i < tekSayilar.Count; i++)
                Console.WriteLine(tekSayilar[i]);
            Console.WriteLine("Çift Sayılar:");
            for (int i = 0; i < ciftSayilar.Count; i++)
                Console.WriteLine(ciftSayilar[i]);
            Console.ReadKey();
        }
    }
}

```

---

**Örnek (Uzunköprü):** Bir öğrencinin bir dersten aldığı tüm sınav notlarını saklayan ve her sınavı eşit ağırlıklandırarak not ortalaması 50 ve üzerindeyse geçti, değilse kaldı yazdıran program.

```
using System;
using System.Collections;
namespace _7_13_ArrayListNotlar
{
class Program
{
static void Main(string[] args)
{
int sayac = 0;
double ortalama = 0;
Console.Write("Ortalamasına bakmak istediğiniz kaç not var:");
int kacNot = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("0 ile 100 arasında bir rakam yazınız.");
ArrayList notlar = new ArrayList();
for (int i = 0; i < kacNot; i++)
{
Console.Write("Aldığınız {0}. notu yazınız: ", i + 1);
notlar.Add(Console.ReadLine());
}
for (int j = 0; j < notlar.Count; j++)
{
sayac += Convert.ToInt32(notlar[j]);
ortalama = (double)sayac / (notlar.Count);
Console.WriteLine("Not Ortalamanız:" + ortalama);
Console.WriteLine(ortalama >= 50 ? "Geçtiniz" : "Kaldınız");
Console.ReadLine();
}
}
}
```

**Örnek (Uzunköprü):** Diziye eklenen değerlerin toplamının, aritmetik ortalamasının, enbüyük ve en küçük değerinin bulunması.

```
using System;
using System.Collections;
namespace _7_14_ArrayListMaxMinAvg{
class Program {
static void Main(string[] args) {
Console.Write("Kaç Sayı Girilecek: ");
int boyut = Convert.ToInt32(Console.ReadLine());
double[] dizi = new double[boyut];
double toplam = 0, max, min;
for (int i = 0; i < boyut; i++)
{
Console.Write("{0}.Sayıyı Giriniz:", i + 1);
dizi[i] = Convert.ToDouble(Console.ReadLine());
}
max = dizi[0];
min = dizi[0];
for (int j = 0; j < boyut; j++)
{
min = min < dizi[j] ? min : dizi[j];
max = max > dizi[j] ? max : dizi[j];
toplam += dizi[j];
}
Console.WriteLine
("Toplam: {0}, Ortalama: {1}, Max: {2}, Min: {3}",
toplam, toplam / dizi.Length, max, min);
Console.ReadKey();
}}}
```

**Örnek (Uzunköprü):** Diziye eklenen değerlerin toplamının, aritmetik ortalamasının, enbüyük ve en küçük değerinin **Linq** sınıf kütüphanesi ile bulunması. Bu örnekte **using System.Linq;** satırı sayesinde kodlar daha kısa olacaktır.

```
using System;
using System.Collections;
using System.Linq;//ArrayList nesnesindeki Max, Min, Sum, Average metotlarını kullanmak için
namespace _7_14_ArrayListMaxMinAvgKisa
{
    class Program
    {
        static void Main(string[] args)
        {
            int diziBoyut;
            Console.Write("Dizi boyutunuz nedir: ");
            diziBoyut = Convert.ToInt32(Console.ReadLine());
            double[] sayilar = new double[diziBoyut];
            for (int i = 0; i < diziBoyut; i++)
            {
                Console.Write("{0}.Sayıyı Giriniz: ", i + 1);
                sayilar[i] = Convert.ToDouble(Console.ReadLine());
            }
            Console.Write("Toplam = {0} Ortalama={1} Max = {2} Min = {3} ",
sayilar.Sum(), sayilar.Average(), sayilar.Max(), sayilar.Min());
            Console.ReadKey();
        }
    }
}
```

#### *ArrayList dezavantajları:*

- ArrayList'e bir eleman eklediğimizde bu eleman object tipinde değilse ArrayList object tipinde elemanlar tuttuğu için arkaplanda önce bu eleman objeye çevrilir öyle ArrayList'e eklenir. Bu performans sorununa neden olacaktır.
- Tersini içinde aynı durum söz konusudur. ArrayList'ten bir eleman çektiğimizde object tipinde bize dönecektir. Fakat biz bu elemanı int tipinde istiyorsak bunu int'e çevirmemiz gerekecektir. Aynı şekilde buda performans sorununa neden olacaktır.
- foreach ile ArrayList'imiz üzerinde döndüğümüzü düşünelim bize gelen her nesne object olduğu için gerçek halinin ne olduğunu bilemeyiz. Gelen her elemanı int tipine çevirsek belki arada bir string tipinde "Ali" gelecektir bu durumda programımız hata alacaktır. Yani non-generic koleksiyonlar **type-safe** değildir.

**Generic List**

Bu dezavantajlardan kurtulmak için **System.Collections.Generic**; sınıf kütüphanesinde yer alan List sınıfından yararlanılabilir.

Söz dizimi şöyledir:

```
List<VeriTipi> ListeAdi = new List<VeriTipi>();
```

Burada VeriTipi yerine T harfi kullanılırsa veri tipi otomatik seçilir.

Örneğin değişik veri tiplerindeki iki değişkenin değerini takas etmek için aşağıdaki gibi bir program yazılabilir:

```
using System;
```

```
using System.Collections.Generic;
```

```
namespace CSD
```

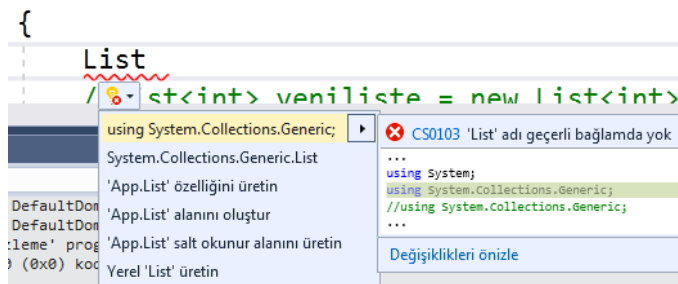
```
{
    class App
    {
        static void takas<T>(ref T a, ref T b)//Takas metodu!
        {
            T temp;//boş kova/takas değişkeni
            temp = a;
            a = b;
            b = temp;
        }
        public static void Main()
        {
            //int ile takas
            int a = 1;
            int b = 2;
            takas<int>(ref a, ref b);//takas metodu çağrılıyor ve ona a,b değerleri veriliyor!
            Console.WriteLine("a=" + a + " ve b=" + b);

            //string ile takas
            string s1 = "Ahmet";
            string s2 = "Mehmet";
            takas<string>(ref s1, ref s2);
            Console.WriteLine("s1=" + s1 + " ve s2=" + s2);
            Console.ReadLine();
        }
    }
}
```

**Not:** C#’ta hangi using kelimesi ile sınıf kütüphanesinin eklenmesi gerektiğini öğrenmek için kullanılmak istenen sınıf yazıldıktan sonra CTRL+Nokta işareti basılır. Örneğin List kullanmak istiyorsak ve sadece using System; yazmışsak

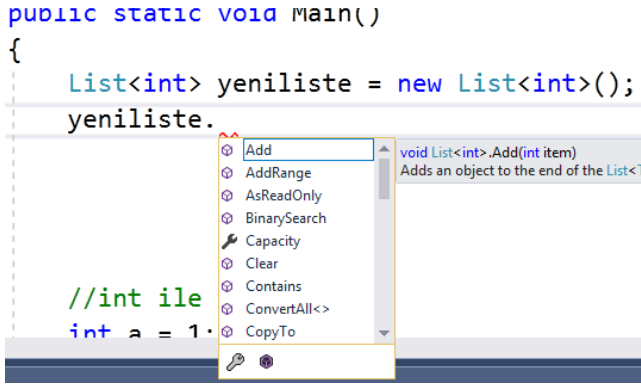
List yazdıktan sonra **CTRL+.** Basarsak bize tavsiyelerde bulunulacaktır.

```
public static void Main()
```





Bir generic liste tanımladıktan sonra liste adını yazıp nokta karakterine basınca seçenekler ortaya çıkacaktır.



- Generic list sınıfı normalde sadece belirtilen tipte eleman alan bir sınıftır. Collection.Generic sınıfını kullanır. Non-Generic Array sınıfının gelişmiş halidir.
- Generic list sınıfı Collections.Generic sınıfını kullanırlar. Kod bloklarının en üst satırında **using System.Collections.Generic;** kodu kullanılmalıdır.

Collections Sınıf Kütüphanesi	Collections.Generic Sınıf Kütüphanesi
ArrayList (object tipinde veri alır)	List<> (belirtilen tipte veri alır)
HashTable (Object tipinde Key, Object tipinde value alır)	Dictionary<> (tipi bildirilen key, tipi bildirilen value alır)
SortedList (Object tipinde eleman kuyruğa eklenir, ilk girilen eleman ilk çıkar)	SortedList<> (tipi bildirilen key'e göre verileri sıralar, tipi bildirilen value alır)
Queue (object tipinde eleman kuyruğa eklenir, ilk giren eleman ilk çıkar)	Queue<> (Bildirilen tipte eleman kuyruğa eklenir, ilk giren eleman ilk çıkar)
Stack (Object tipinde eleman kuyruğa eklenir, son giren eleman ilk çıkar)	Stack<> (Bildirilen tipte eleman kuyruğa eklenir, son giren eleman ilk çıkar)

```
using System; //Uzunköprü
using System.Collections.Generic; //Generic koleksiyonlar bu sınıfı kullanır.
using System.Linq; //myList.ToArray<string>() bu kütüphaneyi kullanır.

namespace _7_19_GenericList
{
    class Program
    {
        static void Main(string[] args)
        {
            List<string> myList = new List<string>(); //Generic List tipinde <yalnız string veri kabul eden>
            myList adında nesne bildiriliyor
            myList.Add("Matematik"); //myList isimli Generic liste sonuna string "Matematik" değeri ekleniyor.
            myList.Add("Fizik");
            myList.Add("İngilizce");
            myList.Add("Kimya");
            myList.Add("Biyoloji");
            myList.Add("İngilizce");
            myList.Add("Coğrafya");
            myList.Add("Kimya");
            myList.Add("Programlama Temelleri");
            foreach (string JenerikList in myList)
            {
                Console.WriteLine(JenerikList);
            }
            myList.Insert(0, "Türkçe"); //Listenin 0. indisine eleman ekleniyor. Insert ile araya eleman
            ekleme.
            Console.WriteLine("\nTürkçe eklenince yeni liste:");
            foreach (string JenerikList in myList)
            {

```

```

Console.WriteLine(JenerikList);
}
Console.WriteLine("Kapasite:" + myList.Capacity); //listenin kapasitesi döner. Kapasite 2^n artar.
myList.TrimExcess(); //listenin eleman sayısı ile kapasiteyi eşitler. Fazla kapasite atılır.
Console.WriteLine("TrimExcess sonrası kapasite:" + myList.Capacity);
Console.WriteLine("Count:" + myList.Count); //Count ile listenin eleman sayısı elde edilir.
//myList.Clear(); //--> .Clear ile listenin tüm elemanları temizlenir.
myList.Remove("İngilizce"); //Remove ile belirtilen eleman silinir. Baştan ilk eleman
silinecektir!
    Console.WriteLine("\nİngilizce silinince yeni liste: (NOT: ilk İngilizce silindi...!!!");
foreach (string JenerikList in myList)
{
    Console.WriteLine(JenerikList);
}
myList.RemoveAt(4); //RemoveAt ile belirtilen indisteki eleman silinir.
myList.RemoveRange(0, 2); //Listenin 0. indisinden itibaren 2 eleman silinir.
    Console.WriteLine("\n0. indisten itibaren 2 eleman (Türkçe-Matematik) silinince yeni
liste:");
foreach (string JenerikList in myList)
{
    Console.WriteLine(JenerikList);
}
    Console.WriteLine("\nContains Kimya için:" + myList.Contains("Kimya")); //Kimya değeri
listede aranır. Bulursa TRUE, bulamazsa FALSE döner.
    Console.WriteLine("\nIndexOf Kimya için:" + myList.IndexOf("Kimya")); //Aranan değeri listede
bulursa ilk elemanın indis numarasını verir. Bulamazsa -1 verir.
    Console.WriteLine("\nLastIndexOf Kimya için:" + myList.LastIndexOf("Kimya")); //Aranan değeri
listede bulursa son değer indis numarasını verir.
    myList.Sort(); //Listeyi küçükten büyüğe (A'dan Z'ye) sıralar.
    Console.WriteLine("\nSıralanınca yeni liste:");
foreach (string JenerikList in myList)
{
    Console.WriteLine(JenerikList);
}
//myList.Add("Coğrafya"); //sıralı listenin sonuna Coğrafya değeri ekleniyor.
myList.Reverse(); //Listeyi sondan başa yeniden oluşturur.
    Console.WriteLine("\nSondan başa doğru yerleştirince yeni liste:");
foreach (string JenerikList in myList) //
{
    Console.WriteLine(JenerikList);
}
string[] dersler = { "Veri Tabanı", "Web Tasarımının Temelleri", "Grafik ve Animasyon" };
myList.AddRange(dersler.ToArray<string>()); //AddRange ile listeye ICollection tipinde
eleman ekleniyor.
    Console.WriteLine("\nAddRange ile listeye değer eklenince yeni liste:");
foreach (string JenerikList in myList) //
{
    Console.WriteLine(JenerikList);
}
string[] ArrayDizi = myList.GetRange(2, 3).ToArray<string>(); //ToArray<string> ile ise generic liste
Array diziyeye çeviriliyor. Listenin tamamını diziyeye dönüştürmek için "GetRange(2, 3)." silinir.
    Console.WriteLine("\nDiziyeye dönüşünce yeni liste:"); //ve GetRange ile listenin 2.
indisinden itibaren toplam 3 değer ArrayDizi'ye aktarılıyor.
foreach (string NormalDizi in ArrayDizi)
{
    Console.WriteLine(NormalDizi);
}
List<string> myList02 = new List<string>();
myList02.Add("Üniversite Yaşam Kültürü");
myList02.AddRange(myList.ToArray<string>()); //Bir listeye bir başka liste eklenebilir.
Önceki veriler korunur.
    Console.WriteLine("\nMyList02'ye MyList eklendi ve yeni MyList02:");
foreach (string JenerikListe in myList02)
{
    Console.WriteLine(JenerikListe);
}

```

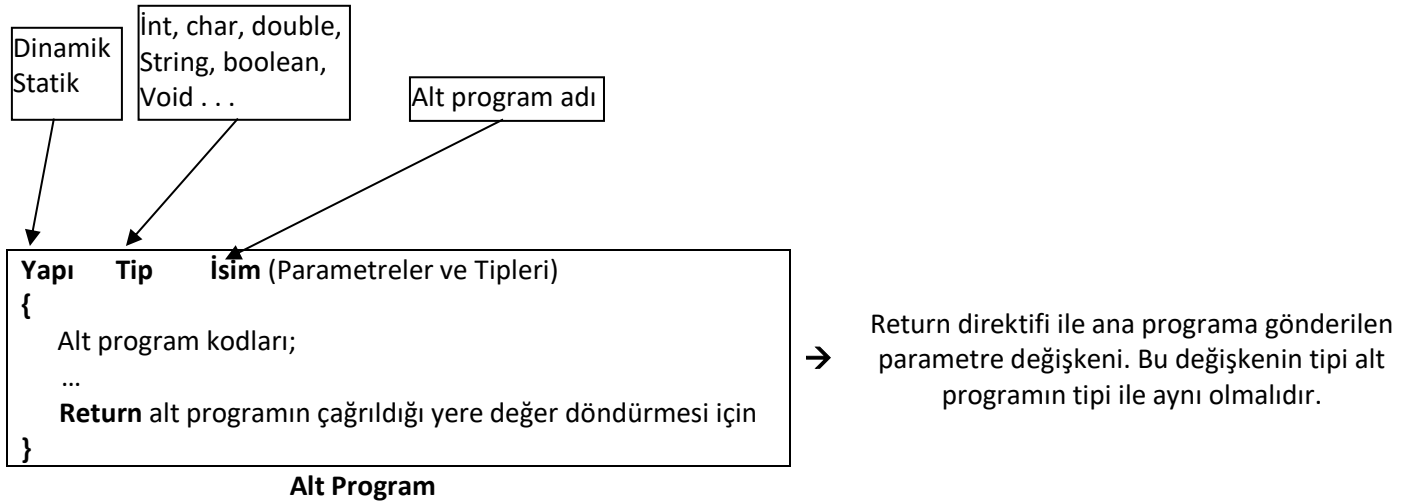
```

    }
    myList02 = myList; //bu satırda myList olduğu myList02'ye kopyalanır. myList02'nin önceki
    değerleri silinir. (Üniversite Yaşam Kültürü)
    Console.WriteLine("\nmyList, myList02'ye kopyalanınca yeni MyList02:");
    foreach (string JenerikListe in myList02)
    {
        Console.WriteLine(JenerikListe);
    }
    Console.ReadKey();
}
}
}

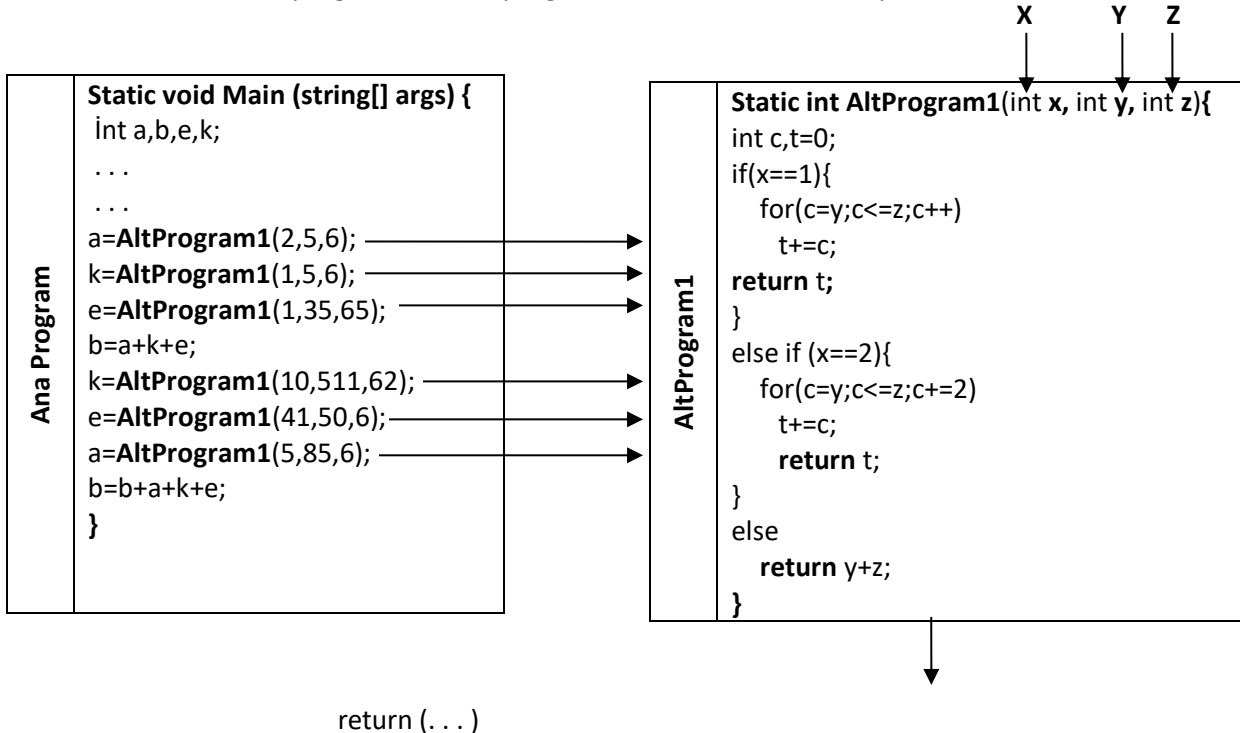
```

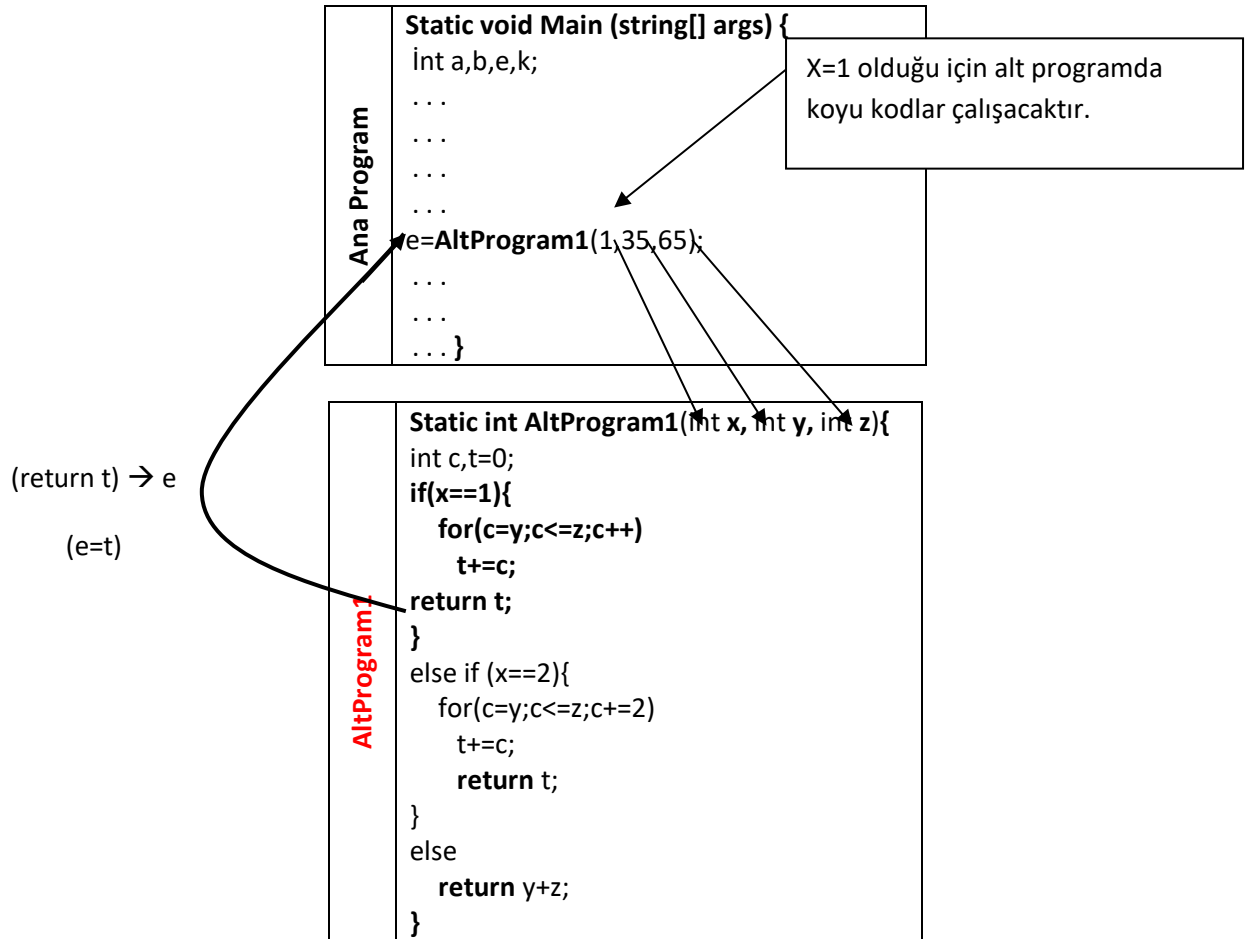
## ALT PROGRAMLAR/FONKSİYONLAR/METOTLAR (SUBROUTINES)

Ana program tarafından çalıştırılan küçük program bloklarıdır. Uzun ve çok kullanılan kodlar bir isim altında bir araya getirilir ve bir isimle çağrılır. Böylece program sade ve anlaşılır hale gelerek programcının kodlara olan hâkimiyetini artırır. Alt programlar hep aynı sonuçları üretebildiği gibi aldığı parametreler ile çalışarak her seferinde farklı sonuçlar da üretebilirler.



Normalde bir C# konsol programında ana program **main** adında ve **void** tipinde birer alt programdır.





### Statik Alt Programlar

Programın çalıştırılmasıyla otomatik olarak oluşturulan alt programlardır. Bu alt programlar oluşturulmadan doğrudan kullanılabilirler. Program içerisinde kullanılmazken de bellekte yer kaplarlar.

**Uygulama (Yaşar, 2011, 266) :**

```
using System;
namespace or266
{
    class Program
    {
        static int toplama(int a, int b) //a ve b değerlerini toplayan ve sonucu dışarı veren toplama adında fonksiyon
        { return (a+b); }
        static void Main(string[] args)
        { int x=10, y=20, z=100,sonuc;
          Console.WriteLine("iki değer sabit ve toplama fonksiyonu ile toplanıyor! X+Y=10+20="+toplama(x,y));
          Console.Write("Birinci Değeri Giriniz: "); //25
          x=Convert.ToInt32(Console.ReadLine());
          Console.Write("İkinci Değeri Giriniz: "); //75
          y=Convert.ToInt32(Console.ReadLine());
          Console.WriteLine("Dışarıdan girilen iki değerin toplamı: X+Y="+x+"+"+y+"="+toplama(x,y));
          sonuc=4+toplama(x,y)*2;
          Console.WriteLine("toplamanın iki katının dört fazlası="+sonuc);
          toplama(x,y);
          Console.Write("Press any key to continue . . . ");
          Console.ReadKey(true);
        } }
    }
```

**Örnek (Yaşar, 2011, 267):**

Dışarıdan girilen sayının faktöriyelini hesaplayan static alt programı oluşturunuz.

**İpucu:** Bir sayının faktöriyeli 1'den kendisine kadarki sayıların çarpımına eşittir. Faktöriyelin sonucu tam sayı olacağı için fonksiyonun geriye döndüreceği değer tamsayıdır. Fonksiyonun giriş parametresi faktöriyeli alınacak olan sayıdır.

```
using System;
namespace or267
{
    class Program
    {
        static int faktoriyel(int a) //a değerinin faktöriyelini alan ve sonucu dışarı veren faktöriyel adında fonksiyon
        {
            int fak=1, b;
            for(b=1;b<=a;b++)
                fak*=b;//fak=fak*b, 1, 2, 6, 24
            return(fak);
        }
        static void Main(string[] args)
        {
            int x,y;
            Console.Write("Faktöriyeli alınacak sayı: ");
            x=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine(x+" Faktöriyel = "+faktoriyel(x));
            y=faktoriyel(x); Console.WriteLine(x+" Faktöriyel="+y);
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek (Yaşar, 2011, 268):** Dışarıdan girilen sayının tek olup olmadığını bulan statik alt program.

**İpucu:** Bir sayı ya tek, ya da çifttir. Bu şekilde iki olasılıklı veriler boolean tanımlanabilir. Bu yüzden tanımlanacak olan alt programın geriye dönüş tipi boolean olabilir. Örneğin sayı tek ise TRUE değeri, çift ise FALSE değeri fonksiyon dışına gönderilebilir. Sayının tek olup olmadığı alt program içinde ekrana yazdırılabilir. Fakat bu durum fonksiyonun kullanım esnekliğini olumsuz yönde etkiler. Ya fonksiyondan dönen tek ya da çift bilgisi ekrana yazdırılmak istenmiyorsa?

```
using System;
namespace or268
{
    class Program
    {
        static bool tek(int a)//sayı tek ise TRUE, çift ise FALSE değerini döndürür
        {
            if (a%2==1) return true;
            else return false;
        }
        static void Main(string[] args)
        {
            int x;
            Console.Write("Sayı: ");
            x=Convert.ToInt32(Console.ReadLine());
            if(tek(x)==true) Console.WriteLine("Girdiğiniz Sayı Tek !");
            else Console.WriteLine("Girdiğiniz Sayı Çift !");
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);    } } }
```

**Örnek (Yaşar, 2011, 269):** Dışarıdan girilen satır sayısı kadar aşağıdaki şekli yapan alt programı oluşturunuz.

**İpucu:** Alt program ekran çıktısı yapacağı için geriye bir değer döndürmesine gerek yoktur. Bu yüzden fonksiyonun tipi **void** olarak belirlenmelidir.



```
using System;
namespace or269
{
    class Program
    {
        static void yildiz(int a)//girilen satır sayısı kadar her satırda bir yıldız koyar
        {
            int b,c,bb=0;
            for(b=1;b<=a;b++)
            {
                for(c=1;c<=bb;c++)
                    Console.Write(" ");
                Console.WriteLine("*");
                bb++; //1, 2, 3
            }
        }
        static void Main(string[] args)
        {
            int ss;
            Console.Write("Satır Sayısı: ");
            ss=Convert.ToInt32(Console.ReadLine());
            yildiz(ss);
            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

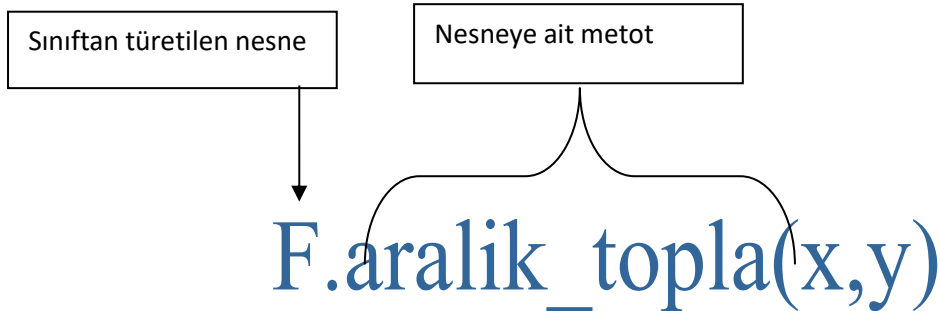
**Örnek:** Ekranda saati gösteren static alt program.

```
using System;
namespace _8_1_MetotSaatKac
{
class Program
{
static void saatKac()
{
while (!Console.KeyAvailable)//bir tuşa basılmadığı sürece döngü çalışır
{
Console.BackgroundColor = ConsoleColor.Red;//arka plan kırmızı
Console.ForegroundColor = ConsoleColor.White;//yazılar beyaz
Console.Clear();//konsol ekranını temizledik.
Console.WriteLine(DateTime.Now.ToLongTimeString());//DateTime sınıfından hazır saati
System.Threading.Thread.Sleep(1000); //programı 1000 milisaniye bekletir. 1000 ms=1
saniye
Console.BackgroundColor = ConsoleColor.White;//arka plan beyaz
Console.ForegroundColor = ConsoleColor.Black;//yazılar siyah
}
Console.ReadKey();
}
static void Main(string[] args)
{
saatKac();
Console.WriteLine("Ana program kaldığı yerden devam ediyor..!");
Console.ReadLine();
}
}
}
```

---

### Static Olmayan Alt Programlar

Programın çalıştırılmasıyla otomatik olarak oluşturulmazlar. Bu tür alt programlar gerektiğinde oluşturulur ve kullanılırlar. Statik olanlara göre hafıza kullanımları bu yüzden daha verimlidir. Dinamik fonksiyonlar oluşturulurken içinde bulunulan sınıftan new direktifi ile bir nesne türetilir ve bu nesneye ait alt program (metot) çağırılarak kullanılır. Aşağıda sınıftan oluşturulan nesne adı F olduğu için F nesnesinin **aralik\_topla** metoduna erişmek için aralarına aitlik bildiren nokta (•) işareti konulur.



**Örnek (Yaşar, 2011, 271):** iki sayı aralığını toplayan dinamik bir alt program.

```
using System;
namespace or271
{
    class Program
    {
        int aralik_topla(int a, int b)
        {
            int toplam=0, k;
            for(k=a;k<=b;k++)
                toplam+=k;
            return(toplam);
        }
        static void Main(string[] args)
        {
            int x=10, y=20;
            Program F=new Program();//dinamik alt program çağırabilmek için Program sınıfında bir nesne tanımlanıyor
            Console.WriteLine("Sabit Değerler (10-20) Arası Toplamı="+F.aralik_topla(x,y));

            Console.Write("Birinci Değeri Giriniz: ");
            x=Convert.ToInt32(Console.ReadLine());
            Console.Write("İkinci Değeri Giriniz: ");
            y=Convert.ToInt32(Console.ReadLine());
            if(x<y) Console.WriteLine("Girdiğiniz "+x+" ve "+y+" değerleri arası toplamı = "+F.aralik_topla(x,y));
            else Console.WriteLine("Girdiğiniz "+x+" ve "+y+" değerleri arası toplamı = "+F.aralik_topla(y,x));
            Console.Write("Press any key to continue . . .");
            Console.ReadKey(true);
        }
    }
}
```



**Örnek (Yaşar, 2011, 272):** Dışarıdan girilen bir kelimenin istenen bir karakterinden başlayıp istenen karakterine kadarını kopyalayan dinamik alt programı substring metodunu kullanmadan oluşturunuz.

**İpucu:** alt programa gönderilecek üç adet parametre var. Bunlardan ilki girilen string kelime. Diğer başlangıç noktası, diğeri ise bitiş noktasıdır. Fonksiyonun geriye döndüreceği değer string tipinde olacaktır. Aşağıda tanımlı parca\_al metodunda ilk parametre string, diğerleri ise tamsayı tipindedir. Metoda göre parametre gönderirken sıralı bir şekilde tanımlandığı sırada olması gerekmektedir.

Kelime: Ebubekir  
Başlangıç Noktası: 2  
Bitiş Noktası: 4  
ube

```
using System;
namespace or272
{
    class Program
    {
        string parca_al(string kelime, int baslangic, int bitis)//
        {
            int a;
            string ys="";
            for(a=baslangic;a<=bitis;a++)//a=2, 3,4, 5, 6 döngü biter
                ys+=kelime[a];//ys=r, rh, rhab
            return(ys);
        }
        static void Main(string[] args)
        {
            int x,y;
            string s;
            Console.Write("Kelimeyi Giriniz: ");//merhaba
            s=Console.ReadLine();
            Console.Write("Başlangıç Noktası: ");//2
            x=Convert.ToInt32(Console.ReadLine());//x=2
            Console.Write("Bitiş Noktası: ");//5
            y=Convert.ToInt32(Console.ReadLine());//y=5
            Program F=new Program();
            Console.WriteLine(F.parca_al(s,x,y));//rhab

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

**Örnek 273:** Dışarıdan girilen bir kelimenin istenen karakterlerini istenen karakterlerle değiştiren dinamik yapıda alt programı replace metodu kullanmadan oluşturunuz.

**İpucu:** Alt programa uygulanacak değişken parametreler üç adettir. Bu parametrelerden birisi string tipinde kelime, diğeri değiştirilecek karakteri belirleyen parametre, sonuncusu ise yerine gelecek parametredir. Son iki parametre char tipinde olmak zorundadır. Metodun geriye döndürdüğü parametre ise string tipinde olmalıdır.

Kelime: Dalama  
Değiştirilecek Karakter: a  
Yeni Karakter: i  
Dilimi

```
using System;
namespace or273
{
    class Program
    {
        string Degistir(string kelime, char a, char b)
        {
            int r;
            string ys="";
            for(r=0;r<kelime.Length;r++)
                if(kelime[r]==a)
                    ys=ys+b;//ys= k, kiva+k=kivak
                else ys=ys+kelime[r];//ys=k+i, ki+v, kiv+a,
            return(ys);
        }
        static void Main(string[] args)
        {
            char x,y;
            byte t;
            string s;
            Console.Write("Kelimeyi Giriniz: ");//sivas
            s=Console.ReadLine();
            Console.Write("Değiştirilecek Karakter: ");//s
            x=Convert.ToChar(Console.ReadLine());//x=s
            Console.Write("Yeni Karakter: ");//k
            y=Convert.ToChar(Console.ReadLine());//y=k
            Program altprogram=new Program();
            Console.WriteLine(altprogram.Degistir(s,x,y));

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

### Alt Programlara Parametre Olarak Dizileri Uygulamak

Bir diziyi parametre olarak alt programa geçirip istenen işlemler yaptırılabilir. Alt programlara sıralı olarak birden fazla parametre geçişi bu şekilde sağlanabilir. Aşağıda statik ve dinamik yapıda iki adet alt program örneği ile bu konu incelenmektedir.

**Örnek (Yaşar, 2011, 275):** Tamsayılardan oluşan bir dizinin en küçük elemanını bulan statik bir yapıda alt program.

**İpucu:** Alt programa giriş olarak istenen boyutta bir dizi gönderilebilir. Dizin elemanları tamsayı olduğu için alt programın geriye döndürdüğü değerin tipi de tamsayı olmalıdır.

```
using System;
namespace or275
{
    class Program
    {
        public static int kucuk( int [] sayilar)
        {
            int e=sayilar[0],k;
            for(k=1;k<sayilar.Length;++k)
            {
                if(e>sayilar[k])
                    e=sayilar[k];
            }
            return e;
        }
        static void Main(string[] args)
        {
            int [] x={15,30,2,55,61,8,35,105,6,5};
            Console.WriteLine("En küçük sayı= "+kucuk(x));

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Örnek (Yaşar, 2011, 276):** Kelimelerden oluşan bir dizinin içerisindeki en uzun kelimeyi bulan dinamik alt program.

**İpucu:** Alt program giriş ve çıkış parametreleri string olacaktır. Stringlerin uzunluğuna bakılarak en uzun kelime bulunup ana programa dönmek gerekmektedir. Eğer istenirse kodlar kelimeler dışarıdan istenecek şekilde düzenlenebilir.

```
using System;
namespace or276
{
    class Program
    {
        public string Uzun_Isim( string[] isimler)//ali, özlem, veli, ufuk
        {
            int enuzun=isimler[0].Length; //enuzun=3,
            int k;
            string uzun="";
            for(k=1;k<isimler.Length;k++)//k=1, 2, 3, 4
            {
                if(enuzun<isimler[k].Length)
                {
                    enuzun=isimler[k].Length;//enuzun=5,
                    uzun=isimler[k];//uzun="özlem";
                }
            }
            return uzun;
        }
        static void Main(string[] args)
        {
            string [] x={"Asuman","Ahmet","Mehmet","Tekin","Emin","Can","İclal","Muzaffer"};
            Program altprogram=new Program();
            Console.WriteLine("En uzun isim= "+altprogram.Uzun_Isim(x));

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

---

**Özyinelemeli – Recursive (kendi kendini çağırın) Alt Programlar**

Yazılan alt program kodları içerisinde aynı alt program kendi içerisinde çağırılıyorsa bu tür alt programlara öz yinelemeli – recursive (kendi kendini çağırın) alt programlar denir.

Bazı durumlar vardır ki For ve While döngüleri dışında başka bir yolla daha kısa kod kullanarak, daha yüksek performans alarak problemler çözülebilir. Bu yollardan biri de öz yinelemeli fonksiyonlardır.

Öz yinelemeli fonksiyonlar bir kontrol ile sınırlandırılmazsa sonsuza kadar kendi içinde çalışır.

**Örnek ozyinelemeli01:** Girilen bir sayının faktöriyelini öz yinelemeli fonksiyon ile bulan program.

**using** System;

**namespace** ozyinele01

```
{
    class program
    {
        public static void Main()
        {
            int sayi;
            Console.Write("Faktöriyeli Alınacak Sayı Nedir? "); //4
            sayi=Convert.ToInt32(Console.ReadLine()); //sayi=4
            Console.WriteLine(sayi+"!="+faktoriyelHesapla(sayi)); //4!=24
            Console.ReadKey(true);
        }
        public static double faktoriyelHesapla(int sayi)
        {
            if (sayi == 1 || sayi == 0)
                return 1;
            else
                return sayi * faktoriyelHesapla(sayi - 1);
        }
    }
}
```

$  \begin{aligned}  &4 * \text{faktoriyelHesapla}(3) \\  &= 4 * 3 * \text{faktoriyelHesapla}(2) \\  &= 4 * 3 * 2 * \text{faktoriyelHesapla}(1) \\  &= 4 * 3 * 2 * 1 = 24  \end{aligned}  $
--

**Örnek ozyinelemeli02:** Baştan kaç fibonacci sayısı görmek istediğini kullanıcıya soran ve o kadar sayıda fibonacci sayısını ekrana yazan program.

**İpucu:** fibonacci sayıları 1 ve 2 ile başlayan ve kendinden önceki iki fibonacci sayısının toplamı ile elde edilen pozitif tamsayılardır. Sıradaki fibonacci sayısı X ile gösterilirse  $X = \text{FibonacciSayısı}(X-1) + \text{FibonacciSayısı}(X-2)$ .

using System;

namespace ozyinele02

```
{
    class Program
    {
        public static void Main()
        {
            int sayi;
            Console.Write("Baştan Kaç Adet Fibonacci Sayısı Görmek İstersiniz? "); //4
            sayi = Convert.ToInt16(Console.ReadLine());
            for(int i=1; i<=sayi; i++) //i=1-2-3-4
                Console.WriteLine(fibonacciHesapla(i));
            Console.ReadKey(true);
        }
        public static int fibonacciHesapla(int sayi)
        {
            if (sayi == 1 || sayi == 0)
                return 1;
            return fibonacciHesapla(sayi-1) + fibonacciHesapla(sayi - 2);
        }
    }
}
```

i= 1 iken ekran return 1 → 1  
i=2 iken fibonacciHesapla(2-1)+fibonacciHesapla(2-2)=1+1= 2  
i=3 iken fibonacciHesapla(3-1)+fibonacciHesapla(3-2)  
= fibonacciHesapla(2)+1  
= (fibonacciHesapla(2-1)+fibonacciHesapla(2-2))+1  
= (1+1)+1=3  
i=4 iken fibonacciHesapla(4-1)+fibonacciHesapla(4-2)  
= (fibonacciHesapla(3)+fibonacciHesapla(2))  
= ((fibonacciHesapla(2)+fibonacciHesapla(1))+1)+ (1+1)  
= 1 + 1 + 1 + 1 + 1 = 5

## HATA YAKALAMA

Program çalışırken kullanıcı kaynaklı veya programcı tarafından tedbir alınmayan durumlarda hatalar ortaya çıktığında programın nasıl davranacağı belirlenebilir. C# programlama dilinde bu durumlarda TRY – CATCH – FINALLY veya TRY – PARSE yöntemleri ile tedbir alınabilir.

TRY – CATCH – FINALLY ile tedbir almadan önce olabilecek hatalar kodlama ile giderilmeye çalışılır. Eğer farklı bir algoritma veya kodlama ile ortaya çıkan hatalar giderilemiyorsa bu durumda TRY – CATCH – FINALLY ile tedbir alınabilir.

Beklenmedik durum=Exception

### TRY – CATCH – FINALLY Yöntemi

TRY – CATCH – FINALLY söz dizimi aşağıdaki gibidir:

try //dene

```
{
    Hataya sebep olan kodlar;
}
catch (System.Exception hata) //oluşacak hata türü ile bilgiler hata isimli nesnede saklanacak.
{
    Hata yakalama kodları;//exception oluştuğunda ne yapılacağı bu kapsam içinde yazılır.
    MessageBox.Show(hata.Message);//hata mesajı ekranda bir kutuda gösteriliyor.
    Console.WriteLine(hata.Message);//hata mesajı konsol ekranda görüntülenir.
    /* üst satırda Message yerine
    * helplink yazılırsa hata ile ilgili yardım dosyasına ulaşılır.
    * source yazılırsa hatanın kaynağı, oluştuğu sınıf bildirilir.
    * stacktrace yazılırsa hata hakkında ayrıntılı bilgi verilir.
    * targetsite yazılırsa hataya neden olan kod bloğu hakkında bilgi verilir. */
    //MessageBox.Show yerine Console.WriteLine kullanılabilir.
    MessageBox.Show(hata.GetType().ToString());//* burada bir mesaj kutusu içinde ne tür bir sistem hatası
    * üretildiği söylenir. */
    MessageBox.Show("Bir hata oluştu, çünkü şundan, bundan, ondan olabilir :-)");//* sistemin ürettiği hata mesajı
    *yerine kendi hata mesajınızı da catch bloğu içinde yazabilirsiniz!*/
}
finally
{
    Hata olsa da, olmasa da çalışması istenen kodlar;
}
```

Oluşacak hata türü önceden biliniyorsa catch parantezi başka türlü düzenlenebilir. Örneğin bir dosya okunması gerekirken okunacak dosya adı hatalı veya bakılan yerden o dosya yoksa aşağıdaki gibi bir catch bloğu düzenlenebilir:

```
catch (System.IO.FileNotFoundException)//
{
    MessageBox.Show("Dosya adı yanlış veya belirtilen dosya belirtilen konumda yok! Dosya bulunamadı");
}

catch (System.DivideByZeroException)
{
    MessageBox.Show("Sıfıra bölme hatası! Payda kısmı sıfır olmasın lütfen..!");
}
```

Eğer isterseniz bir try bloğu için birden fazla catch bloğu yazabilirsiniz:

```
try
{
rtb.LoadFile("C:\veriler.dat");
}
Catch()
{
MessageBox.Show("Bir hata oluştu!");
}
catch (System.IO.FileNotFoundException)
{
MessageBox.Show("Dosya bulunamadı!");
}
catch(.....)
{
MessageBox.Show("Başka bir şey de olmuş olabilir!");
}
```

### Try – Catch Örnek01:

```
static void Main(string[] args)
{
int sayi;
Console.Write("Bir sayı giriniz:");//beş
try
{
sayi = Convert.ToInt32(Console.ReadLine());
Console.Write("girilen sayının karesi="+sayi*sayi);
}
catch (Exception hata)
{

Console.Write("oluşan hata mesajı= "+hata.Message);
}
finally
{
Console.WriteLine("\nFinally kapsamına yazılan satırlar hata oluşsa da, oluşmasa da çalışırlar!");
}
Console.ReadLine();
}
```

Yukarıdaki örnekte finally bloğu hiç kullanılmasa da program catch bloğundan sonraki normal akışına devam eder.

Peki öyleyse finally bloğu nerelerde işe yarar? Return kullanılan fonksiyonlarda, return nerede kullanılırsa kullanılsın mutlaka çalışması istenen satırlar varsa bu satırlar finally kapsamı içinde kullanılır. Bunun için aşağıdaki örneği inceleyelim:



**Try – Catch Örnek02:**

```

static void kareal()
{
    int sayi;
    Console.Write("Bir sayı giriniz:");
    try
    {
        sayi = Convert.ToInt32(Console.ReadLine());
        Console.Write("girilen sayının karesi=" + sayi * sayi);
    }
    catch (Exception hata)
    {
        Console.Write("oluşan hata mesajı= " + hata.Message);
    }
    finally
    {
        Console.WriteLine("\nFinally kapsamına yazılan satırlar hata oluşsa da, oluşmasa da çalışırlar!\nreturn olsa bile..!");
    }
    Console.WriteLine("kareal metodu (fonksiyonu) hala çalışıyor..! return kullanılmadığı için bu satır hep çalışır. Finally yine işe yaramadı:");
}
static void Main(string[] args)
{
    kareal();
    Console.WriteLine("Şimdi ana program çalışıyor..!");
    Console.ReadLine();
}

```

---

Yukarıdaki örnekte return kullanılmadığı için finally bloğunun yine etkisi olmadı!

Şimdi aynı programı, kareal metodu içindeki try ve catch bloklarının son satırına return komutu yazarak deneyelim.

**Try – Catch Örnek03:**

```

static void kareal()
{
    int sayi;
    Console.Write("Bir sayı giriniz:");
    try
    {
        sayi = Convert.ToInt32(Console.ReadLine());
        Console.Write("girilen sayının karesi=" + sayi * sayi);
        return;
    }
    catch (Exception hata)
    {
        Console.Write("oluşan hata mesajı= " + hata.Message);
        return;
    }
    finally
    {
        Console.WriteLine("\nFinally kapsamına yazılan satırlar hata oluşsa da, oluşmasa da çalışırlar!\nreturn olsa bile..!");
    }
    Console.WriteLine("fonksiyon return ile sonlandırılmışsa sadece finally kapsamı içindeki satırlar çalışır. Bu satır hata olsa da, olmasa da hiç çalışmayacaktır!");
}
static void Main(string[] args)
{
    kareal();
    Console.WriteLine("Şimdi ana program çalışıyor..!");
    Console.ReadLine();
}

```

## TryParse Yöntemi

Bu metod tip dönüşümü başarılı ise TRUE değeri döndürür ve **out** kelimesinden sonra gelen değişkene sonucu yazar, başarısız ise FALSE değeri döndürür ve **out** kelimesinden sonra gelen değişkene sıfır yazar. Böylece bir if ile hata denetimi yapılabilir.

Neyedönüştürecek.TryParse(neyidönüştürecek, **out** neredetutacak);

### TryParse Örnek 01:

```
static void Main(string[] args)
{
    int sayi;
    yeniden:Console.Write("bir sayı gir:");//iki
    if (Int32.TryParse(Console.ReadLine(), out sayi))
    {
        Console.WriteLine("Dönüştürme başarılı!");
        Console.WriteLine("girilen sayının karesi=" + sayi * sayi);
    }
    else
    {
        Console.WriteLine("dönüştürme yapılamadı!");
        Console.WriteLine("hatalı giriş!"); goto yeniden;
    }
    Console.ReadLine();
}
```

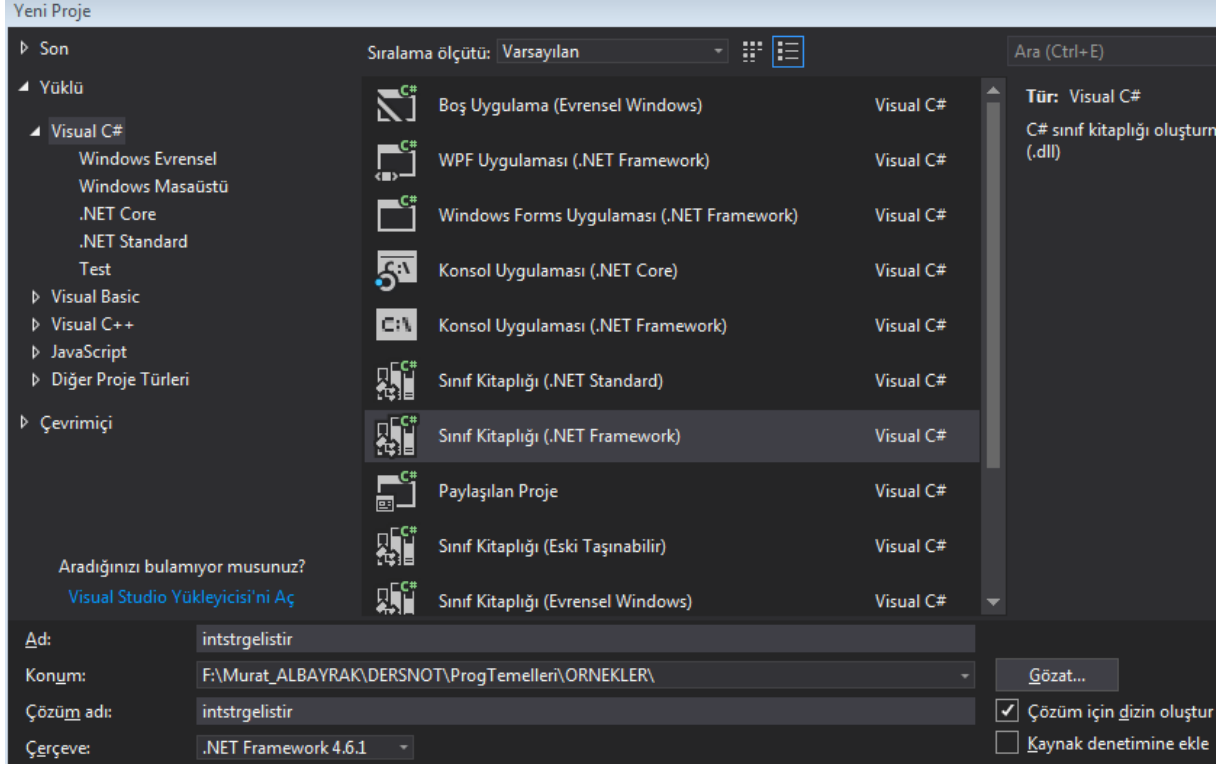
---

## DLL METOTLARI

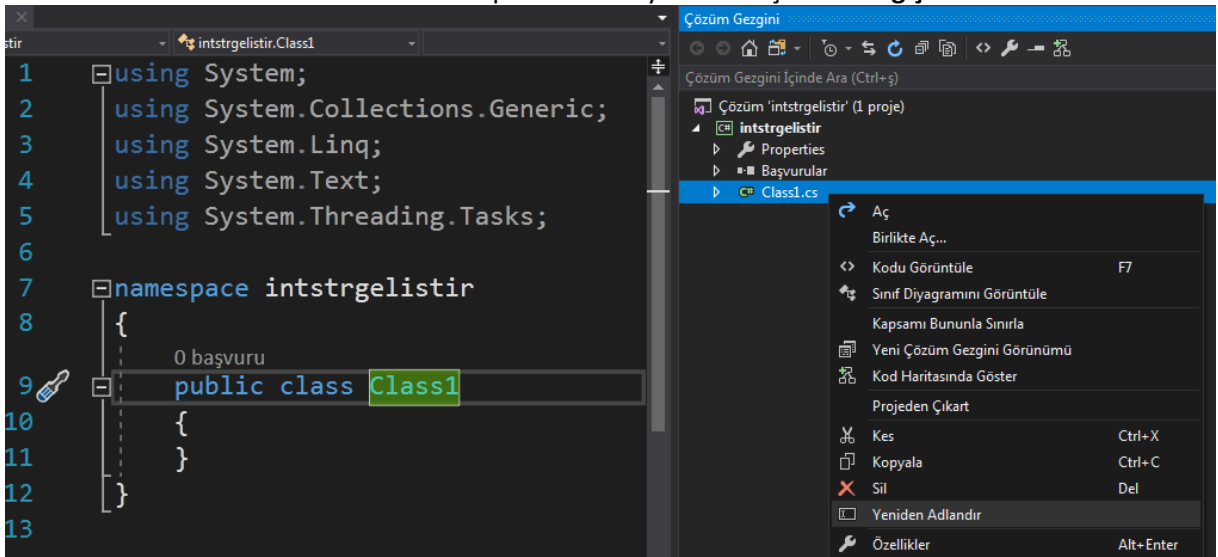
Belirli işlemleri yapan kodlar sınıf kütüphanesi (**class library**) olarak kaydedildikleri takdirde diğer projelerde ayrıntıya girmeden pratik bir şekilde kullanılabilirler. Class library projeleri derlendikleri takdirde **.exe** uzantılı yerine **.DLL (Dynamic Link Library)** uzantılı dosya/dosyalar oluşur. DLL uzantılı dosyalar ise programların dinamik olarak bağlantı kurarak kullanabileceği **metotlar** içerir.

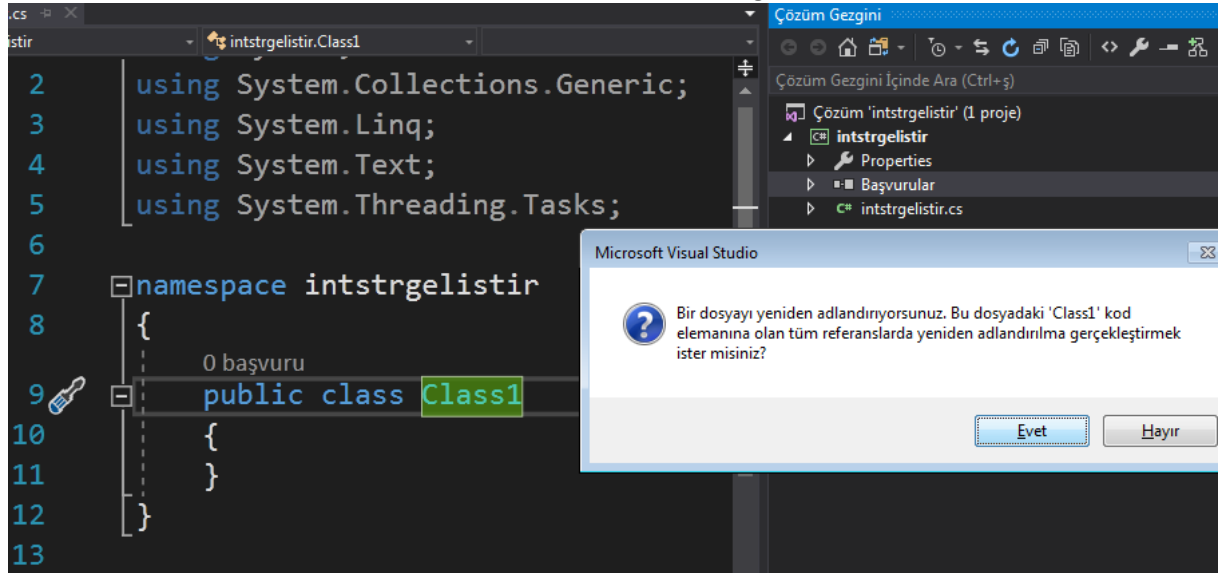
**Örneğin**, int ve string veri tiplerini geliştiren bir metot yazalım. Bu metot bir değerin sayısal olup olmadığını, eğer sayısal ise tek mi çift mi olduğunu tespit etsin. Bunun için;

- 1) Visual Studio'da **Dosya/Yeni/Proje/Sınıf Kütüphanesi** (File/New/Project/Class Library) seçilerek sınıf kütüphanesi eklenir.

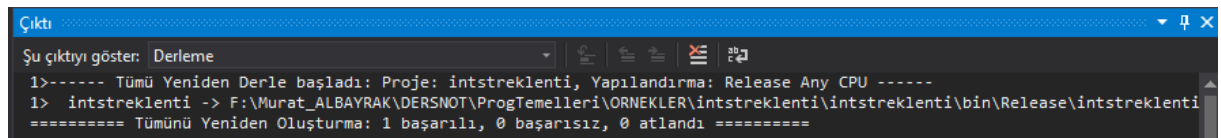
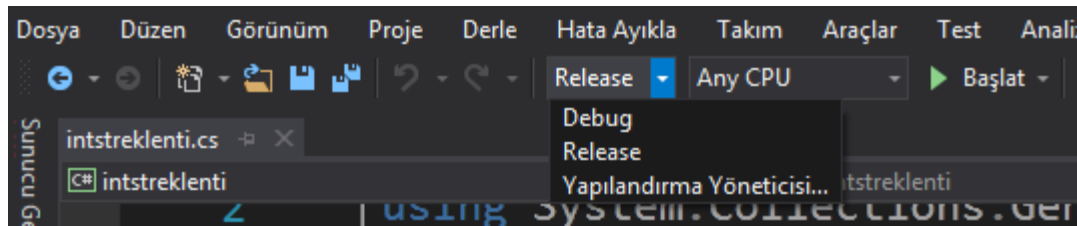


- 2) Gelen ekranda **Class 1** sınıfının adı namespace adı ile aynı olacak şekilde **değiştirilir**.

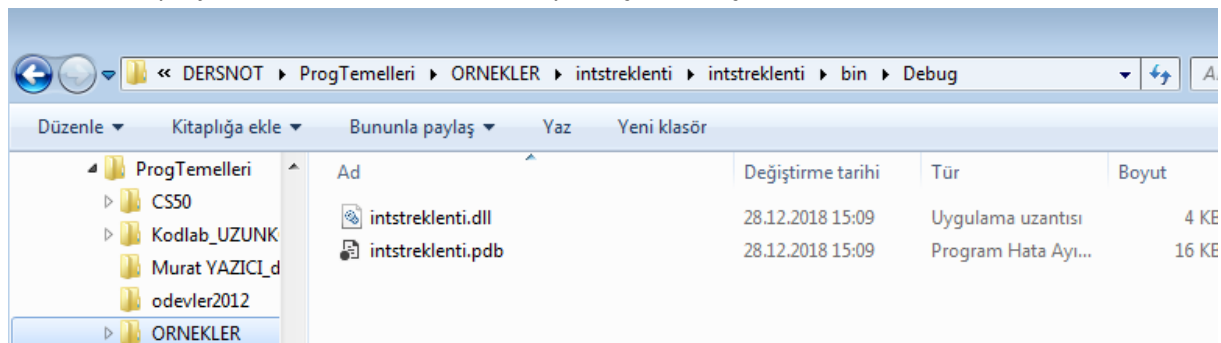




- 4) Çözüm yapılandırması **Release** moduna alınır ve **CTRL+Shift+B** ile veya **Shift+F6** ile veya **Derle/Yapılandır** **intstreklenti (Debug/Build intstreklenti)** ile derleme yapılır.



Bu durumda proje klasöründe DLL uzantılı dosya oluşturulmuş olacaktır.



- 5) class içine metotlar yazılır. DLL dosyası içinde sadece class'lar (sınıflar) olabilir. Main metodu olmaz. Metotlar ve bunların içinde bulunduğu sınıflar **static** olmalıdır.

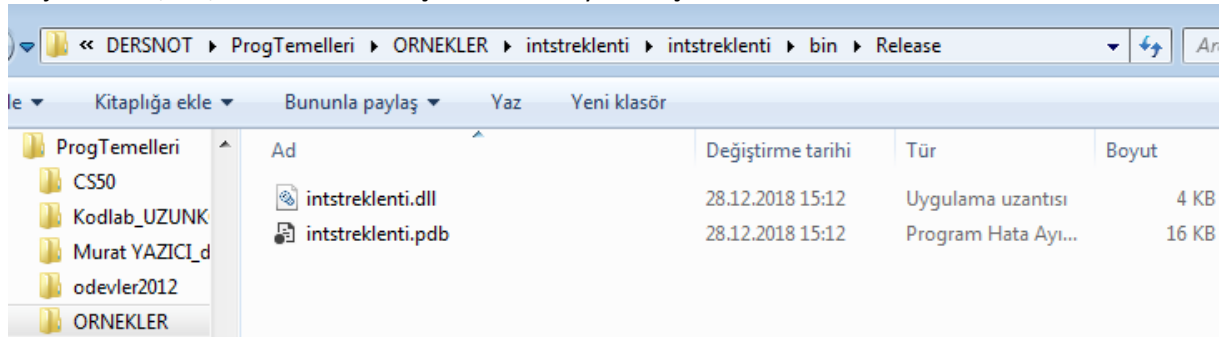
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

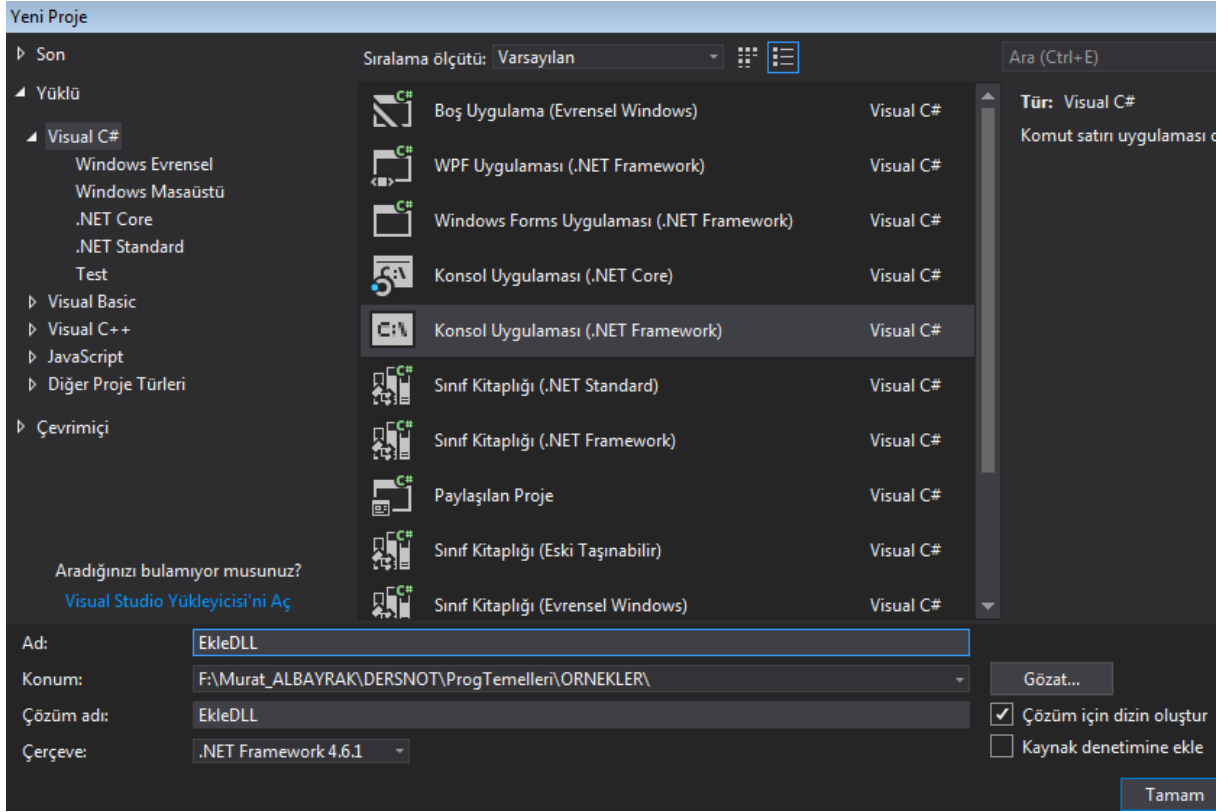
namespace intstrgelistir
{
    public static class intstrgelistir
    {
        public static bool ciftmi(this int sayi)
        {
            return sayi % 2 == 0;
        }
        public static bool sayisalmi(this string sayi)
        {
            double sayisal = 0;
            return double.TryParse(sayi, out sayisal);
        }
    }
}

```

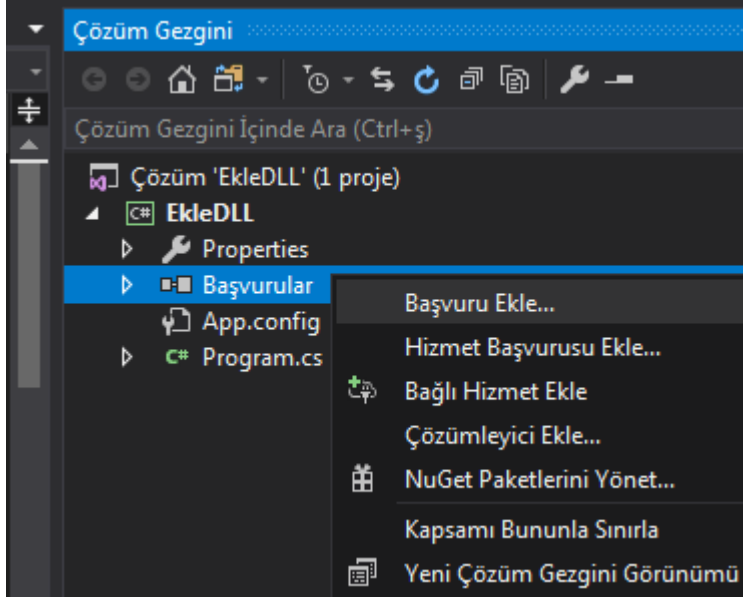
- 6) TEKRAR Çözüm yapılandırması **Release** moduna alınır ve **CTRL+Shift+B** ile veya **Shift+F6** ile veya **Derle/Yapılandır intstreklenti (Debug/Build intstreklenti)** ile derleme yapılır. Proje klasörü/Bin/Release klasörü içinde DLL dosyası oluşturulacaktır.



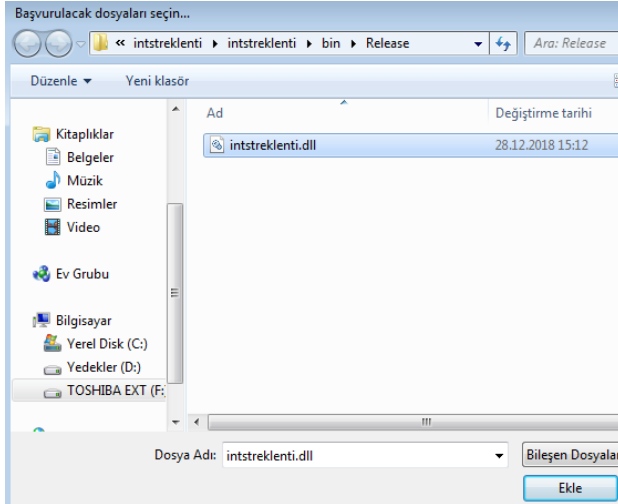
7) Artık C# projesi açıldığında ona özel DLL dosyası eklenebilir.



8) Daha sonra gelen ekranda **Çözüm Gezgini/Başvurular** sağ tık **başvuru ekle...**

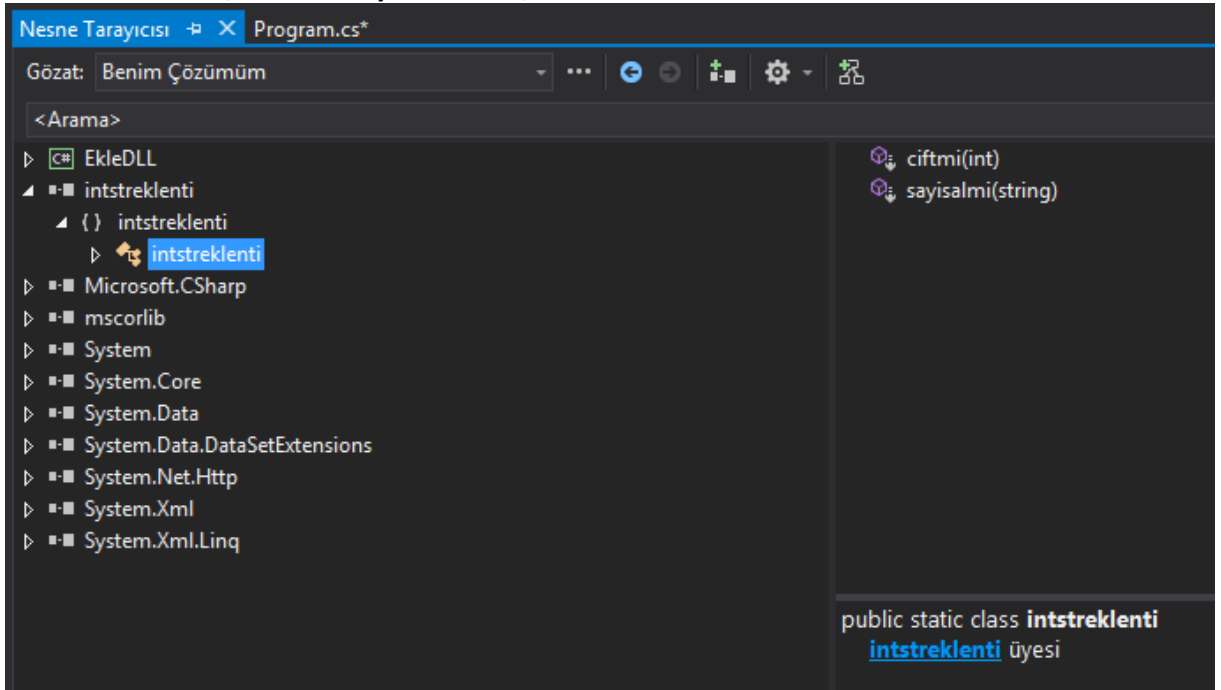


- 9) Daha sonra gelen ekranda **Gözet...** tıklanır ve DLL dosyasının bulunduğu klasörden ilgili DLL dosyası seçilerek eklenir.



- 10) Daha önce oluşan DLL dosyası güncel proje içine de kopyalanabilirdi.

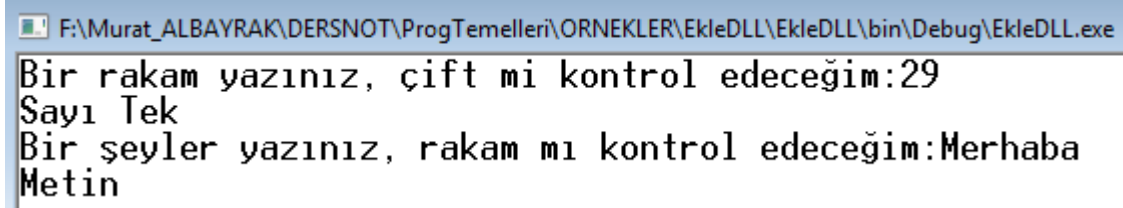
- 11) İstenirse **Görünüm/Nesne Tarayıcı CTRL+W,J** ile eklenen DLL izlenebilir.



12) Using kelimesi ile DLL eklentisi projeye dâhil edilir ve kodlar aşağıdaki gibi olabilir:

```
using System;
using intstrekleri;
namespace EkleDLL
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Bir rakam yazınız, çift mi kontrol edeceğim:");
            int sayi = int.Parse(Console.ReadLine());
            Console.WriteLine(sayi.ciftmi() ? "Sayı Çift" : "Sayı Tek");
            Console.WriteLine("Bir şeyler yazınız, rakam mı kontrol edeceğim:");
            string yazi = Console.ReadLine();
            Console.WriteLine(yazi.sayisalimi() ? "Rakam" : "Metin");
            Console.Read();
        }
    }
}
```

13) Projeyi F5 ile test ettiğimizde örneğin aşağıdaki gibi bir çıktı görünecektir:



```
F:\Murat_ALBAYRAK\DERSNOT\ProgTemelleri\ORNEKLER\EkleDLL\EkleDLL\bin\Debug\EkleDLL.exe
Bir rakam yazınız, çift mi kontrol edeceğim:29
Sayı Tek
Bir şeyler yazınız, rakam mı kontrol edeceğim:Merhaba
Metin
```



## KAYNAKÇA

Ebubekir Yaşar. **Algortima ve Programlamaya Giriş**. Ekin Yayınevi.

Süleyman UZUNKÖPRÜ. **Algoritmalar**. Kodlab Yayınları.

Emre Ayrılmaz. **C# 3.0 ile Programlama Temelleri**. Bilge Adam Yayınları, İstanbul:2008

Mehmet Aktaş, EBİL Eğitim ve Yazılım Ltd. Şti.

Mustafa Aksu, **Algoritma ve Programlamaya Giriş Ders Notları**.<[www.hakankör.com.tr](http://www.hakankör.com.tr)>

[www.gokhandokuyucu.com](http://www.gokhandokuyucu.com) <Eylül 2011>

MEGEP, **Programlama Temelleri Modülü**. Ankara:2007

MEGEP, **Kodlamaya Hazırlık Modülü**, 482BK0123, Ankara:2011

Onur Yalazı. **Programlama Temelleri-1**. <[www.ceturk.com](http://www.ceturk.com)>, Eylül 2011

Şeref Akyüz. **C#'taRecursive Fonksiyonlar**.< <http://www.serefakyuz.com/2011/04/csharp-ta-recursive-oz-yinelemeli-fonksiyonlar.html>>, Eylül 2011

<http://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/csharp/ch05.pdf>

<http://oguzalpundegerli.blogspot.com/2012/04/float-double-decimal-arasndaki-temel.html>

<https://lbozan.blogspot.com>